МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НТУ «Дніпровська політехніка»

Інститут електроенергетики

Факультет інформаційної технології

Кафедра САіУ

ЗВІТ

з практичної роботи

дисципліни «Веб-застосунки з Java/Spring»

Виконав: ст. гр. 122м-24-1

Байдашний Єгор Станіславович

Перевірив: доц. Мінєєв О.С

Дніпро

2025

**Завдання**

Розробити програмне забезпечення мовою Java з використанням фреймворку Spring Boot. Взаємодію між клієнтом і сервером реалізувати за допомогою REST API. Тематика та функціональне призначення додатка можуть бути довільними. Наприклад, це може бути сервіс пошуку квартир без участі ріелторів у місті Дніпро на платформі OLX. Основні вимоги до додатка:

* Реалізація можливості експорту даних у форматі Excel (наприклад, звіт або статистика).
* Отримання та обробка інформації з Інтернету (наприклад, з інтернет-магазинів або платформ оголошень).
* Використання відкритого REST API для отримання певної інформації (наприклад, курсу долара з ПриватБанку).
* Наявність фронтенд-компонента (щонайменше використання шаблонізатора з мінімальним рівнем складності).
* Інтеграція з базою даних (реляційною або нереляційною). У разі використання реляційної бази даних слід застосовувати H2. Скрипт для створення необхідних таблиць та інших об'єктів бази даних необхідно додати до проєкту у файлі db.sql.

**Результат**

Створений застосунок автоматизує процес збору вакансій із ресурсу work.ua, здійснюючи пошук на основі заданих ключових слів. Окрім цього, він отримує актуальний курс долара через відкритий API ПриватБанку. Отримані дані про вакансії зберігаються у відповідній таблиці бази даних. Для виведення інформації про вакансії та поточний валютний курс використовується шаблонізатор Thymeleaf.

Після запуску застосунку, додаток може бути доступний за адресою http://localhost:8080, і має такі ресурси:

* GET / – повертає представлення переліку товарів ;
* POST /import – імпортує даних, та виконує редирект на головну сторінку;
* POST /delete – виконує видалення ресурсу та редирект;
* GET /api/job – повертає перелік вакансій в форматі json;
* POST /api/job – виконує додавання вакансій;
* POST /api/job /delete – виконує видалення ресурсу;
* GET /api/job /export – повертає дані сформованого файлу .xlsx в вигляді масиву байтів.

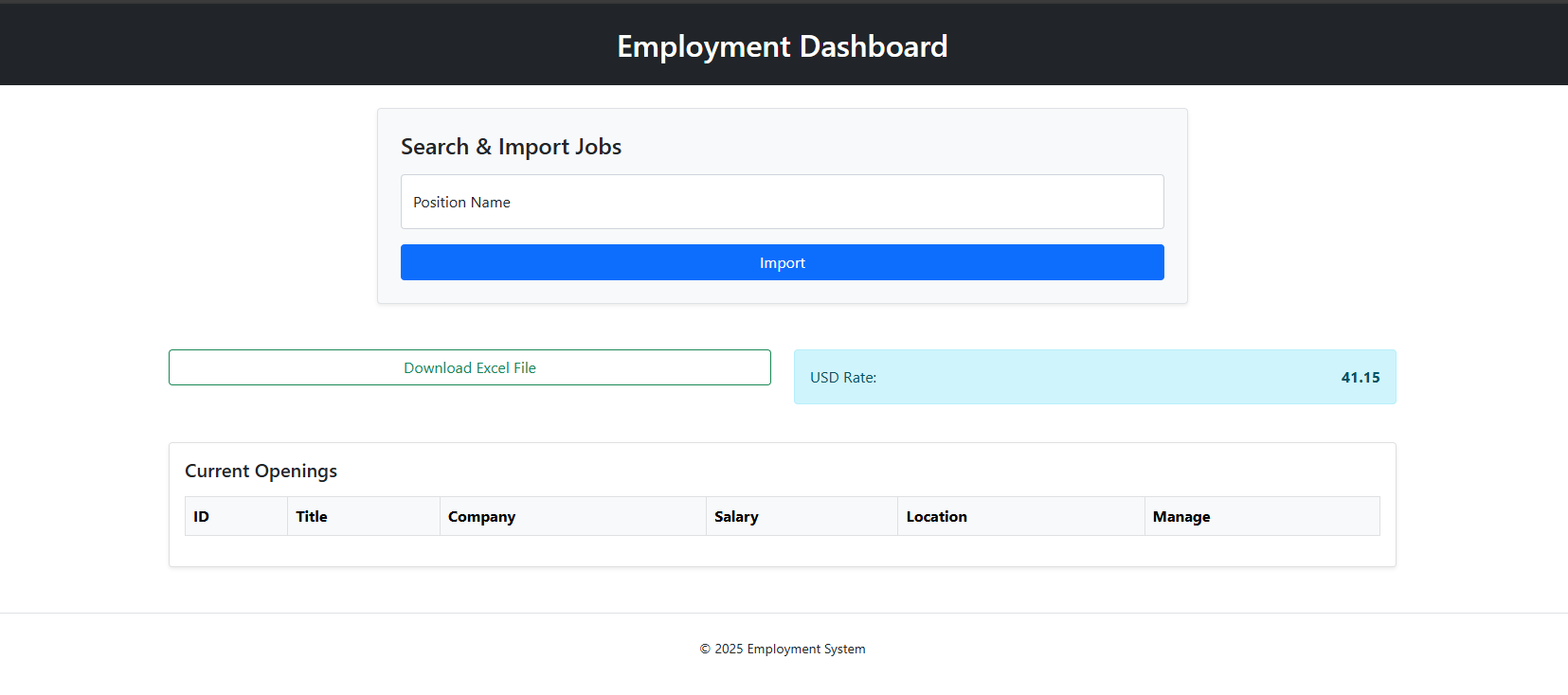


Рис. 2. Стартова сторінки застосунку

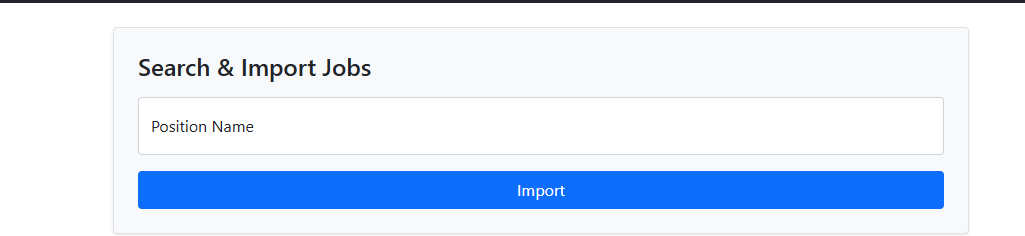


Рис. 3. Заповнення списку знайдених вакансій

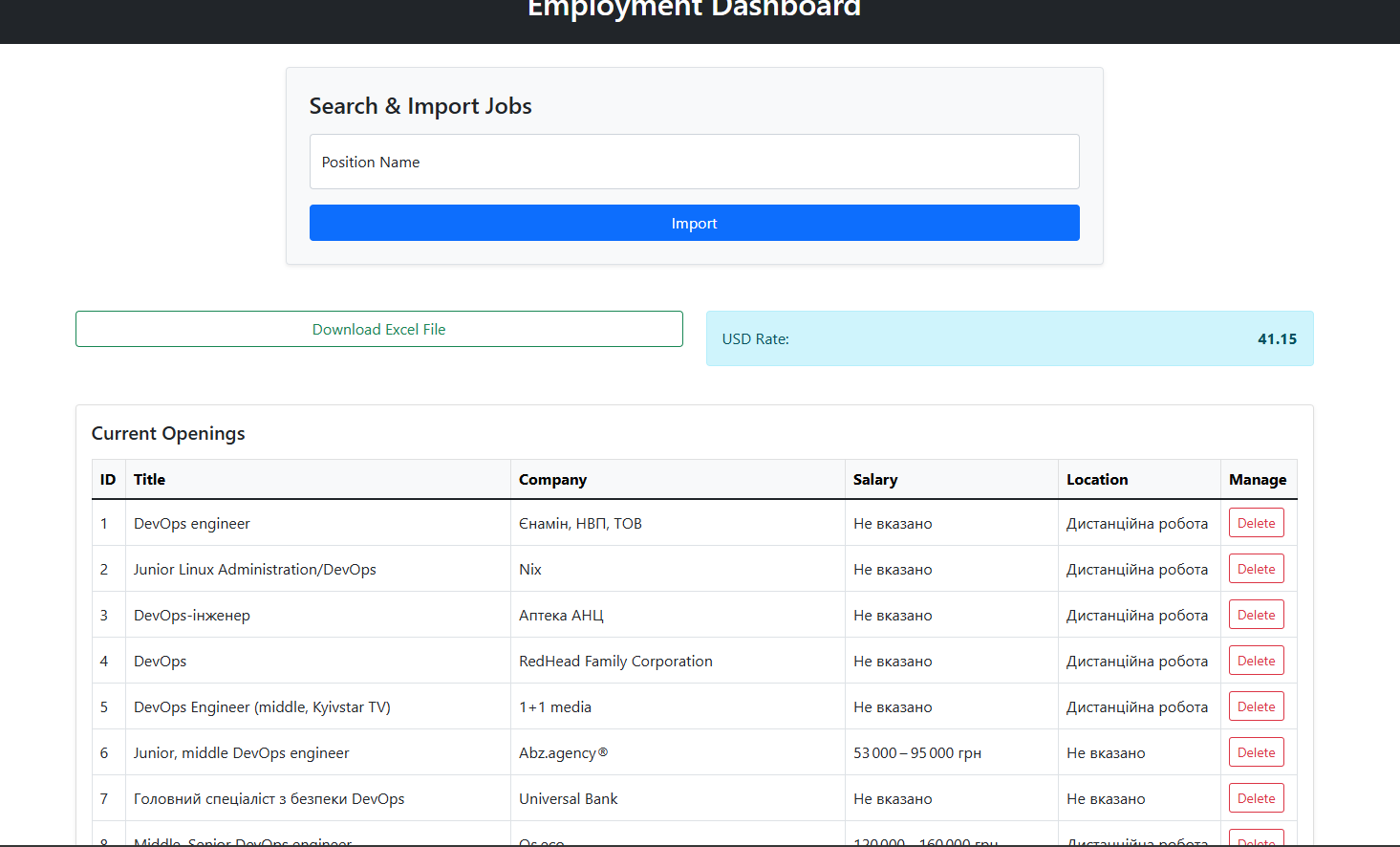


Рис. 4. Знайдені вакансії

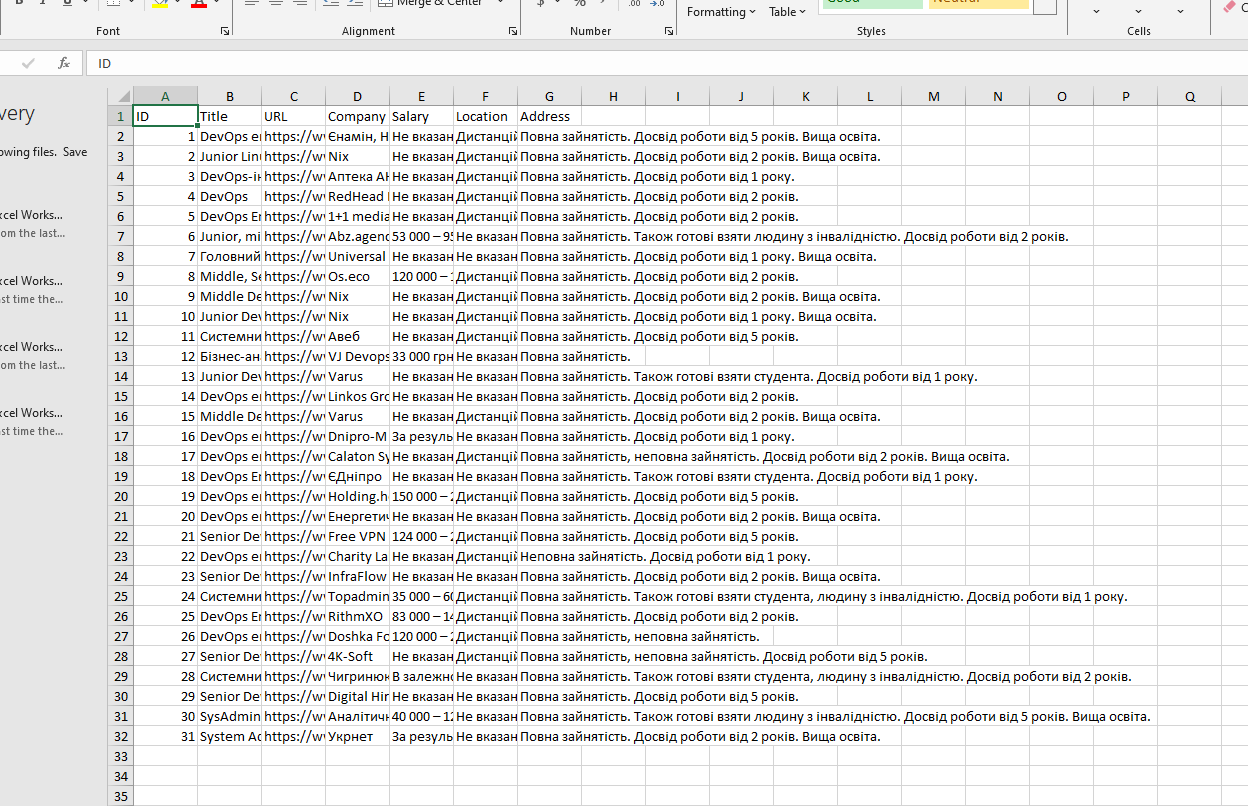


Рис. 5. Експортовані дані

**Код програми**

**Файл «JobController.java»**

package com.example.demo.controller;

import java.io.IOException;

import java.util.List;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.entity.Vacancy;

import com.example.demo.repository.IVacancyRepository;

import com.example.demo.service.ExcelExportService;

import com.example.demo.service.VacancyParserService;

@RestController

@RequestMapping("/api/vacancies")

public class JobController {

private final VacancyParserService vacancyParserService;

private final IVacancyRepository vacancyRepository;

private final ExcelExportService excelExportService;

public VacancyController(VacancyParserService vacancyParserService, IVacancyRepository vacancyRepository, ExcelExportService excelExportService) {

this.vacancyParserService = vacancyParserService;

this.vacancyRepository = vacancyRepository;

this.excelExportService = excelExportService;

}

@GetMapping

public List<Vacancy> getVacancies(@RequestParam String job) throws IOException {

return vacancyParserService.parseVacancies(job);

}

@PostMapping

public ResponseEntity<List<Vacancy>> addVacancies(@RequestParam String job) throws IOException {

List<Vacancy> vacancies = vacancyParserService.parseVacancies(job);

if (vacancies.isEmpty()) {

return ResponseEntity.notFound().build();

}

vacancyRepository.saveAll(vacancies);

return ResponseEntity.status(HttpStatus.CREATED).body(vacancies);

}

@PostMapping("/delete")

public ResponseEntity<String> deleteVacancy(@RequestParam Long id) {

if (vacancyRepository.existsById(id)) {

vacancyRepository.deleteById(id);

return ResponseEntity.ok("Vacancy with ID " + id + " deleted successfully.");

}

return ResponseEntity.status(HttpStatus.NOT\_FOUND)

.body("Vacancy with ID " + id + " not found.");

}

@GetMapping("/export")

public ResponseEntity<byte[]> exportVacanciesToExcel() throws IOException {

List<Vacancy> vacancies = vacancyRepository.findAll();

byte[] excelFile = excelExportService.exportVacanciesToExcel(vacancies);

HttpHeaders headers = new HttpHeaders();

headers.add("Content-Disposition", "attachment; filename=vacancies.xlsx");

return new ResponseEntity<>(excelFile, headers, HttpStatus.OK);

}

}

**Файл «JobPageController.java»**

*package com.example.demo.controller;*

*import java.io.IOException;*

*import java.util.List;*

*import org.springframework.stereotype.Controller;*

*import org.springframework.ui.Model;*

*import org.springframework.web.bind.annotation.GetMapping;*

*import org.springframework.web.bind.annotation.PostMapping;*

*import org.springframework.web.bind.annotation.RequestParam;*

*import com.example.demo.entity.Vacancy;*

*import com.example.demo.repository.IVacancyRepository;*

*import com.example.demo.service.PrivateRateFetcher;*

*import com.example.demo.service.VacancyParserService;*

*@Controller*

*public class VacancyPageController {*

*private final VacancyParserService vacancyParserService;*

*private final IVacancyRepository vacancyRepository;*

*private final PrivateRateFetcher rateFetcher;*

*public VacancyPageController(VacancyParserService vacancyParserService, IVacancyRepository vacancyRepository, PrivateRateFetcher rateFetcher) {*

*this.vacancyParserService = vacancyParserService;*

*this.vacancyRepository = vacancyRepository;*

*this.rateFetcher = rateFetcher;*

*}*

*@GetMapping*

*public String showVacanciesPage(Model model) {*

*List<Vacancy> vacancies = vacancyRepository.findAll();*

*model.addAttribute("vacancies", vacancies);*

*model.addAttribute("dollarRate", rateFetcher.getDollarRate());*

*return "vacancies";*

*}*

*@PostMapping("/import")*

*public String importVacancies(@RequestParam String job) throws IOException {*

*List<Vacancy> vacancies = vacancyParserService.parseVacancies(job);*

*if (!vacancies.isEmpty()) {*

*vacancyRepository.saveAll(vacancies);*

*}*

*return "redirect:/";*

*}*

*@PostMapping("/delete")*

*public String deleteVacancy(@RequestParam Long id) {*

*if (vacancyRepository.existsById(id)) {*

*vacancyRepository.deleteById(id);*

*}*

*return "redirect:/";*

*}*

*}*

**Файл «Vacancy.java»**

package com.example.demo.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "vacancy")

public class Vacancy {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String url;

private String company;

private String salary;

private String location;

private String address;

public Vacancy() {}

public Vacancy(String title, String url, String company, String salary, String location, String address) {

this.title = title;

this.url = url;

this.company = company;

this.salary = salary;

this.location = location;

this.address = address;

}

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getUrl() { return url; }

public void setUrl(String url) { this.url = url;}

public String getCompany() { return company; }

public void setCompany(String company) { this.company = company; }

public String getSalary() { return salary; }

public void setSalary(String salary) { this.salary = salary; }

public String getLocation() { return location; }

public void setLocation(String location) { this.location = location; }

public String getAddress() { return address; }

public void setAddress(String address) { this.address = address; }

}

**Файл «IVacancyRepository.java»**

package com.example.demo.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import com.example.demo.entity.\*;

import jakarta.transaction.Transactional;

@Repository

public interface IVacancyRepository extends JpaRepository<Vacancy, Long>{

}

**Файл «ExcelExportService.java»**

package com.example.demo.service;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.util.List;

import org.apache.poi.xssf.usermodel.XSSFCell;

import org.apache.poi.xssf.usermodel.XSSFRow;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import org.springframework.stereotype.Service;

import com.example.demo.entity.Vacancy;

@Service

public class ExcelExportService {

public byte[] exportVacanciesToExcel(List<Vacancy> vacancies) throws IOException {

XSSFWorkbook workbook = new XSSFWorkbook();

XSSFSheet sheet = workbook.createSheet("Vacancies");

XSSFRow headerRow = sheet.createRow(0);

String[] columnNames = {"ID", "Title", "URL", "Company", "Salary", "Location", "Address"};

for (int i = 0; i < columnNames.length; i++) {

XSSFCell cell = headerRow.createCell(i);

cell.setCellValue(columnNames[i]);

}

int rowNum = 1;

for (Vacancy vacancy : vacancies) {

XSSFRow row = sheet.createRow(rowNum++);

row.createCell(0).setCellValue(vacancy.getId());

row.createCell(1).setCellValue(vacancy.getTitle());

row.createCell(2).setCellValue(vacancy.getUrl());

row.createCell(3).setCellValue(vacancy.getCompany());

row.createCell(4).setCellValue(vacancy.getSalary());

row.createCell(5).setCellValue(vacancy.getLocation());

row.createCell(6).setCellValue(vacancy.getAddress());

}

ByteArrayOutputStream baos = new ByteArrayOutputStream();

workbook.write(baos);

baos.close();

return baos.toByteArray();

}

}

**Файл «PrivateRateFetcher.java»**

package com.example.demo.service;

import org.json.JSONArray;

import org.springframework.stereotype.Component;

import org.springframework.web.client.RestTemplate;

@Component

public class PrivateRateFetcher {

private static final String API\_URL = "https://api.privatbank.ua/p24api/pubinfo?json&exchange&coursid=5";

private final RestTemplate template;

public PrivateRateFetcher() {

this.template = new RestTemplate();

}

public float getDollarRate() {

String response = fetchExchangeRate();

if (response == null) {

return -1;

}

return getUsdBuyRateFromResponse(response);

}

private String fetchExchangeRate() {

try {

return template.getForObject(API\_URL, String.class);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

private float getUsdBuyRateFromResponse(String response) {

try {

JSONArray jsonArray = new JSONArray(response);

String buyRate = jsonArray.getJSONObject(1).getString("buy");

return Float.parseFloat(buyRate);

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

}

**Файл «VacancyDetailsParser.java»**

package com.example.demo.service;

import java.io.IOException;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.springframework.stereotype.Component;

import com.example.demo.entity.Vacancy;

@Component

public class VacancyDetailsParser {

public Vacancy parseVacancy(String jobUrl) throws IOException {

Document jobDoc = Jsoup.connect(jobUrl).get();

String title = jobDoc.selectFirst("h1").text();

String company = getElementText(jobDoc, "span[title='Дані про компанію'] + a");

String salary = getElementText(jobDoc, "span[title='Зарплата'] + span");

String location = getElementText(jobDoc, "li:has(span[title='Місце роботи'])");

String address = getElementText(jobDoc, "li:has(span[title='Умови й вимоги'])");

return new Vacancy(title, jobUrl, company, salary, location, address);

}

private String getElementText(Document doc, String selector) {

Element element = doc.selectFirst(selector);

return (element != null) ? element.text() : "Не вказано";

}

}

**Файл «VacancyParser.java»**

package com.example.demo.service;

import java.util.List;

public interface VacancyParser {

public List<String> getVacancyLinks(String jobName);

}

**Файл «VacancyParserService.java»**

package com.example.demo.service;

import java.io.IOException;

import java.util.List;

import java.util.stream.Collectors;

import org.springframework.stereotype.Service;

import com.example.demo.entity.Vacancy;

@Service

public class VacancyParserService {

private final VacancyParser listParser;

private final VacancyDetailsParser detailsParser;

public VacancyParserService(VacancyParser listParser, VacancyDetailsParser detailsParser) {

this.listParser = listParser;

this.detailsParser = detailsParser;

}

public List<Vacancy> parseVacancies(String jobName) throws IOException {

List<String> vacancyLinks = listParser.getVacancyLinks(jobName);

return vacancyLinks.stream()

.map(link -> {

try {

return detailsParser.parseVacancy(link);

} catch (IOException e) {

e.printStackTrace();

return null;

}

})

.filter(vacancy -> vacancy != null)

.collect(Collectors.toList());

}

}

**Файл «VacancyUrlBuilder.java»**

package com.example.demo.service;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

@Component

public class VacancyUrlBuilder {

private final String baseUrl;

public VacancyUrlBuilder(@Value("${parser.base-url}") String baseUrl) {

this.baseUrl = baseUrl;

}

public String buildUrl(String jobName, int page) {

return String.format("%s/jobs-%s?page=%d", baseUrl, jobName, page);

}

public String getBaseUrl() {

return baseUrl;

}

}

**Файл «WorkUaVacancyParser.java»**

package com.example.demo.service;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

import org.springframework.stereotype.Component;

@Component

public class WorkUaVacancyParser implements VacancyParser{

private static final int MAX\_PAGES = 5;

private final VacancyUrlBuilder urlBuilder;

public WorkUaVacancyParser(VacancyUrlBuilder urlBuilder) {

this.urlBuilder = urlBuilder;

}

@Override

public List<String> getVacancyLinks(String jobName){

List<String> vacancyLinks = new ArrayList<>();

for (int page = 1; page <= MAX\_PAGES; page++) {

String url = urlBuilder.buildUrl(jobName, page);

Document doc = new Document("");

try {

doc = Jsoup.connect(url).get();

} catch (IOException e) {

e.printStackTrace();

}

Elements jobElements = doc.select("#pjax-jobs-list h2 a");

if (jobElements.isEmpty()) {

break;

}

for (Element job : jobElements) {

String jobUrl = urlBuilder.getBaseUrl() + job.attr("href");

vacancyLinks.add(jobUrl);

}

}

return vacancyLinks;

}

}

**Файл «Vacancies.html»**

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<meta charset=*"UTF-8"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1.0"*>

<title>Vacancies</title>

<link href=*"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"* rel=*"stylesheet"*>

</head>

<body>

<div class=*"container mt-5"*>

<h1 class=*"text-center"*>Vacancies</h1>

<form id=*"searchForm"* method=*"POST"* action=*"/import"*>

<div class=*"mb-3"*>

<label for=*"job"* class=*"form-label"*>Enter Job Title</label>

<input type=*"text"* class=*"form-control"* id=*"job"* name=*"job"* required>

</div>

<button type=*"submit"* class=*"btn btn-primary"*>Import Vacancies</button>

</form>

<br>

<a th:href= *"@{/api/vacancies/export}"* class=*"btn btn-success mb-3"*>Download Excel</a>

<h3>Курс долара</h3>

<div class=*"alert alert-info"* role=*"alert"*>

(USD): <strong th:text=*"${dollarRate}"*></strong>

</div>

<hr>

<h2 class=*"mt-4"*>Vacancy List</h2>

<table class=*"table table-bordered mt-3"*>

<thead>

<tr>

<th>ID</th>

<th>Title</th>

<th>Company</th>

<th>Salary</th>

<th>Location</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr th:each=*"vacancy : ${vacancies}"*>

<td th:text=*"${vacancy.id}"*></td>

<td th:text=*"${vacancy.title}"*></td>

<td th:text=*"${vacancy.company}"*></td>

<td th:text=*"${vacancy.salary}"*></td>

<td th:text=*"${vacancy.location}"*></td>

<td>

<form th:action=*"@{/delete}"* method=*"POST"* style="display:*inline*;">

<input type=*"hidden"* th:name=*"id"* th:value=*"${vacancy.id}"* />

<button type=*"submit"* class=*"btn btn-danger"*>Delete</button>

</form>

</td>

</tr>

</tbody>

</table>

</div>

<script src=*"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"*></script>

</body>

</html>

**Файл «application.properties»**

spring.application.name=demo

spring.jpa.hibernate.ddl-auto=update

spring.mvc.hiddenmethod.filter.enabled=true

spring.datasource.url=jdbc:h2:file:./db

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

parser.base-url=https://www.work.ua

parser.max-pages=5

**Висновок**

Створено програмний модуль на основі Java Spring, що здійснює збір і фільтрацію даних із зовнішніх джерел. Реалізовано функціональність автоматичного отримання та обробки інформації шляхом парсингу. Передбачено експорт результатів у файл Excel для подальшого аналізу чи архівації. Розроблено інтерфейс користувача, який дозволяє працювати з даними в інтерактивному режимі. Впроваджено окремий сервіс для взаємодії з відкритими API, зосереджений на обробці та виведенні актуального валютного курсу.