МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НТУ «Дніпровська політехніка»

Інститут електроенергетики

Факультет інформаційної технології

Кафедра САіУ

ЗВІТ

з лабораторної роботи

дисципліни «Веб-застосунки з Java/Spring»

Виконала: ст. гр. 122м-24-2

Мусієнко Анастасія Сергіївна

Перевірив:

доц. Мінєєв О.С

Дніпро

2025

**Завдання:** Розробити програму мовою Java на базі фреймворку Spring boot. Спілкування кліента/сервера реалізувати через REST.

- Додаток повинен вивантажувати звіт у форматі Excel

- Додаток повинен парсити інформацію з інтернету

- Додаток також повинен брати інформацію по відкритому Rest API

- Додаток повинен мати Front End частину

- Можливо спілкування з БД

**Хід роботи**

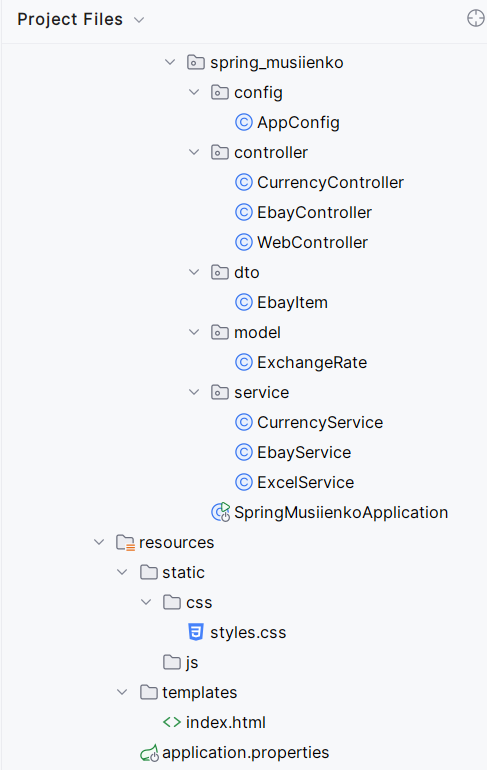


Рис 1. Файлова структура проекту

Мета застосунку — створити простий інструмент, який дозволяє, швидко дізнаватися скільки товар буде коштувати в гривнях, які є пропозиції на eBay, та виваантижити список пропозицій в Excel.

Основний клас SpringMusiienkoApplication запускає додаток. Файл pom.xml — це конфігураційний файл Maven, де зазначені всі залежності, які використовуються в проєкті, зокрема Spring Boot, Jsoup, Apache POI, Thymeleaf тощо.

Клас AppConfig створює RestTemplate, який використовується для HTTP-запитів. CurrencyController обробляє запит для отримання курсу долара, звертаючись до сервісу CurrencyService, який отримує дані з API ПриватБанку.

Контролер EbayController дозволяє шукати товари на eBay за допомогою EbayService, який парсить HTML сторінки результатів пошуку через бібліотеку Jsoup. Також цей контролер надає можливість експорту знайдених товарів у Excel-файл через ExcelService, який формує таблицю з даними за допомогою Apache POI.

Клас WebController відповідає за відображення стартової HTML-сторінки index.html. На сторінці можна переглянути курс долара, здійснити пошук товарів на eBay, та завантажити Excel-файл з результатами.

Модель ExchangeRate описує структуру валютних даних, а EbayItem — товари, отримані з eBay.

Користувач відкриває інтерфейс за адресою http://localhost:8080.

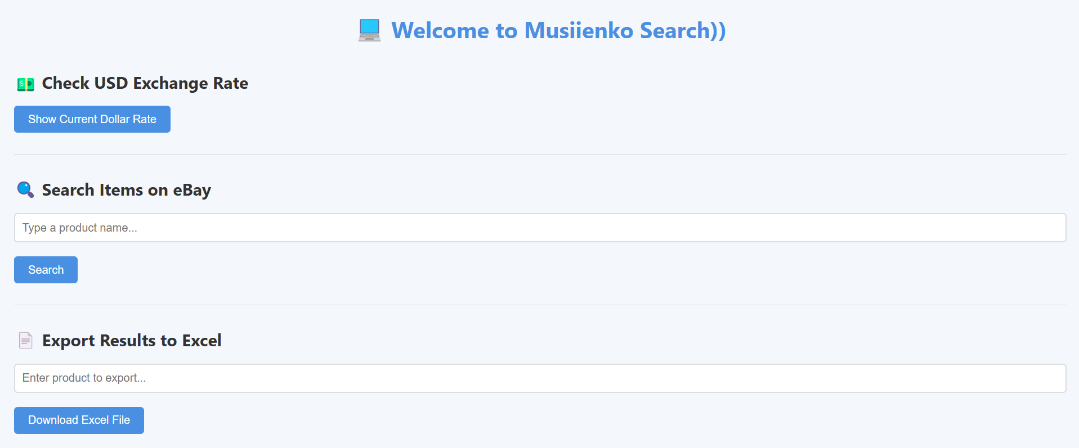


Рис 2. Інтерфейс програми

Користувач натискає кнопку Show Current Dollar Rate, і програма відображає актуальний курс.

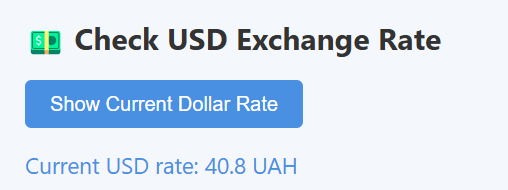


Рис 3. Актуальний курс

Користувач вводить назву товару, натискає Search, отримує список товарів із цінами.

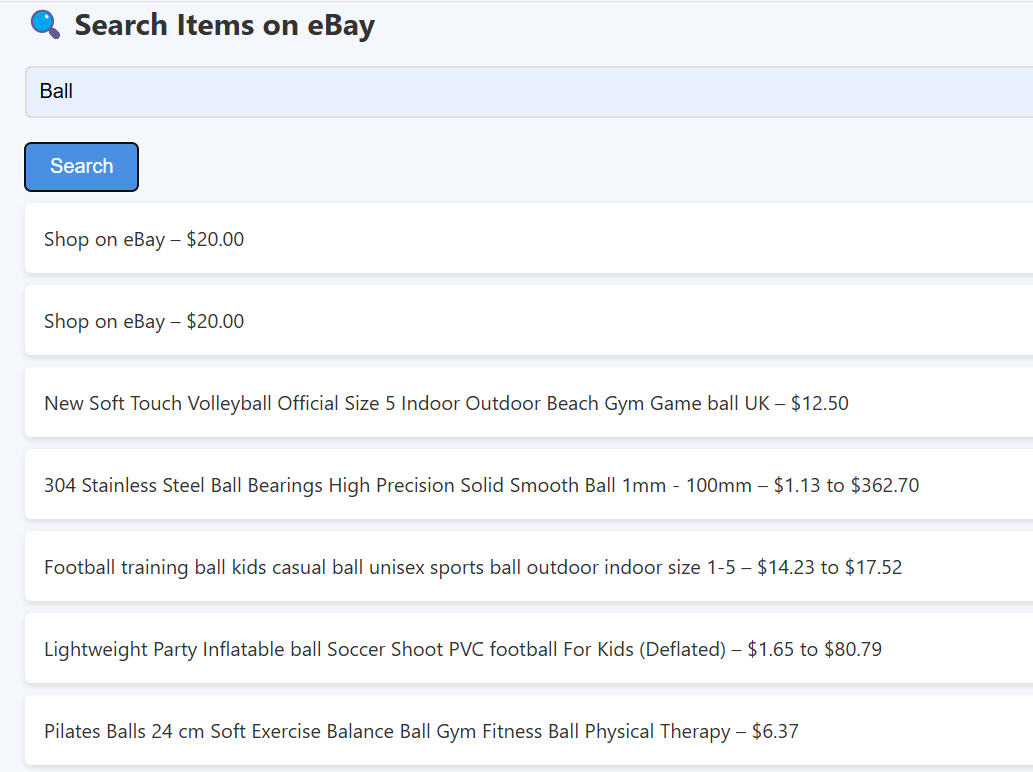


Рис 4. Актуальний курс

Користувач натискає Download Excel File, і у відповідь отримує файл зі знайденими товарами.

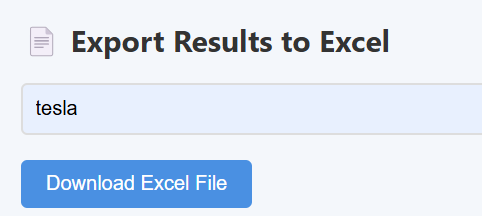


Рис 5. Вибір товару для формування звіту

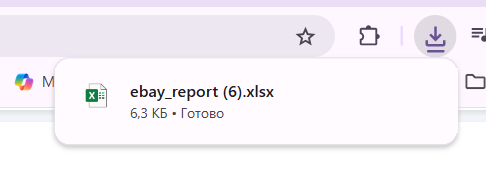


Рис 6. Завантаження звіту

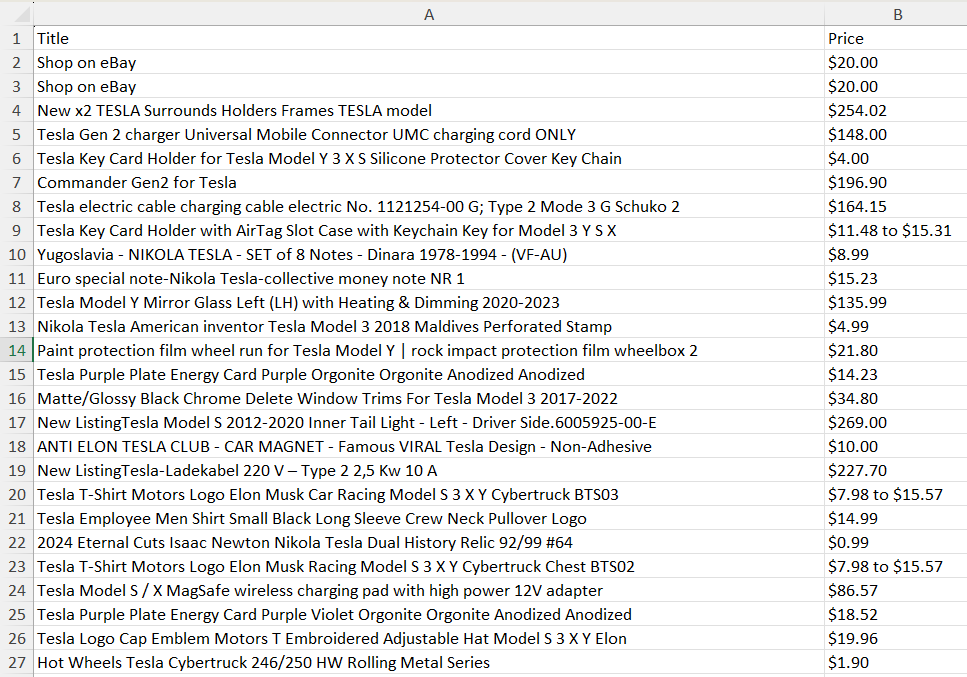


Рис 7. Звіту

**Висновок**:

У ході лабораторної роботи було розроблено повнофункціональний вебзастосунок на базі Spring Boot, який реалізує підключення до відкритої REST API приватбанку, парсинг вебсторінок ebay, вивантаження звіту форматі Excel, REST архітектура для обміну даними між клієнтом та сервером, проста фронтенд-частина. Ця лабораторна робота дозволила краще зрозуміти роботу Spring Boot та інтеграції сторонніх сервісів у Java-застосунки.

**Лістинг програми**

**SpringMusiienkoApplication.java**

package com.anastasiia.spring\_musiienko;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringMusiienkoApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringMusiienkoApplication.class, args);  
 }  
  
}

**Pom.xml**

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.4.4</version>  
 <relativePath/> *<!-- lookup parent from repository -->* </parent>  
 <groupId>com.anastasiia</groupId>  
 <artifactId>Spring\_Musiienko</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Spring\_Musiienko</name>  
 <description>Spring\_Musiienko</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 <spring-cloud.version>2024.0.1</spring-cloud.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.jsoup</groupId>  
 <artifactId>jsoup</artifactId>  
 <version>1.16.1</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.poi</groupId>  
 <artifactId>poi-ooxml</artifactId>  
 <version>5.2.3</version>  
 </dependency>  
 <dependency>  
 <groupId>jakarta.persistence</groupId>  
 <artifactId>jakarta.persistence-api</artifactId>  
 <version>3.1.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-webflux</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
 <scope>provided</scope>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.dataformat</groupId>  
 <artifactId>jackson-dataformat-xml</artifactId>  
 <version>2.13.1</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 </dependencies>  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring-cloud.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**AppConfig.java**

package com.anastasiia.spring\_musiienko.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.client.RestTemplate;  
  
@Configuration  
public class AppConfig {  
  
 @Bean  
 public RestTemplate restTemplate() {  
 return new RestTemplate();  
 }  
}

**CurrencyController.java**

package com.anastasiia.spring\_musiienko.controller;  
  
import com.anastasiia.spring\_musiienko.service.CurrencyService;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class CurrencyController {  
  
 private final CurrencyService currencyService;  
  
 public CurrencyController(CurrencyService currencyService) {  
 this.currencyService = currencyService;  
 }  
  
 @GetMapping("/get-dollar-rate")  
 public String getDollarRate() {  
 return currencyService.getDollarExchangeRate();  
 }  
}

**EbayController.java**

package com.anastasiia.spring\_musiienko.controller;  
  
  
import com.anastasiia.spring\_musiienko.dto.EbayItem;  
import com.anastasiia.spring\_musiienko.service.EbayService;  
import com.anastasiia.spring\_musiienko.service.ExcelService;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.util.List;  
  
@RestController  
public class EbayController {  
  
 private final EbayService ebayService;  
 private final ExcelService excelService;  
  
 public EbayController(EbayService ebayService, ExcelService excelService) {  
 this.ebayService = ebayService;  
 this.excelService = excelService;  
 }  
  
 @GetMapping("/parse-ebay")  
 public ResponseEntity<List<EbayItem>> parseEbay(@RequestParam String searchQuery) {  
 List<EbayItem> ebayItems = ebayService.getEbayData(searchQuery);  
 return ResponseEntity.*ok*(ebayItems);  
 }  
  
 @GetMapping("/download-excel")  
 public ResponseEntity<byte[]> downloadExcel(@RequestParam String searchQuery) {  
 List<EbayItem> ebayItems = ebayService.getEbayData(searchQuery);  
 byte[] excelFile = null;  
  
 try {  
 excelFile = excelService.generateExcelReport(ebayItems);  
 } catch (Exception e) {  
 return ResponseEntity.*status*(500).body(null);  
 }  
  
 HttpHeaders headers = new HttpHeaders();  
 headers.add("Content-Disposition", "attachment; filename=ebay\_report.xlsx");  
  
 return ResponseEntity.*ok*()  
 .headers(headers)  
 .body(excelFile);  
 }  
}

**WebController.java**

package com.anastasiia.spring\_musiienko.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
  
@Controller  
public class WebController {  
  
 @GetMapping("/")  
 public String home() {  
 return "index";  
 }  
}

**EbayItem.java**

package com.anastasiia.spring\_musiienko.dto;  
  
public class EbayItem {  
 private String title;  
 private String price;  
  
 public EbayItem(String title, String price) {  
 this.title = title;  
 this.price = price;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getPrice() {  
 return price;  
 }  
  
 public void setPrice(String price) {  
 this.price = price;  
 }  
}

**ExchangeRate.java**

package com.anastasiia.spring\_musiienko.model;  
  
public class ExchangeRate {  
 private String ccy; *// Код валюты* private String base\_ccy; *// Базовая валюта* private double buy;  
 private double sale;  
  
 public String getCcy() {  
 return ccy;  
 }  
  
 public void setCcy(String ccy) {  
 this.ccy = ccy;  
 }  
  
 public String getBase\_ccy() {  
 return base\_ccy;  
 }  
  
 public void setBase\_ccy(String base\_ccy) {  
 this.base\_ccy = base\_ccy;  
 }  
  
 public double getBuy() {  
 return buy;  
 }  
  
 public void setBuy(double buy) {  
 this.buy = buy;  
 }  
  
 public double getSale() {  
 return sale;  
 }  
  
 public void setSale(double sale) {  
 this.sale = sale;  
 }  
}

**CurrencyService.java**

package com.anastasiia.spring\_musiienko.service;  
  
import com.anastasiia.spring\_musiienko.model.ExchangeRate;  
import org.springframework.stereotype.Service;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.http.ResponseEntity;  
  
@Service  
public class CurrencyService {  
  
 private final RestTemplate restTemplate;  
  
 public CurrencyService(RestTemplate restTemplate) {  
 this.restTemplate = restTemplate;  
 }  
  
 public String getDollarExchangeRate() {  
 String url = "https://api.privatbank.ua/p24api/pubinfo?json&exchange&coursid=5";  
  
 ResponseEntity<ExchangeRate[]> response = restTemplate.getForEntity(url, ExchangeRate[].class);  
  
 if (response.getBody() != null) {  
 for (ExchangeRate rate : response.getBody()) {  
 if ("USD".equals(rate.getCcy())) {  
 return "Current USD rate: " + rate.getBuy() + " UAH";  
 }  
 }  
 }  
  
 return "It was not possible to get the dollar rate.";  
 }  
}

**EbayService.java**

package com.anastasiia.spring\_musiienko.service;  
  
import com.anastasiia.spring\_musiienko.dto.EbayItem;  
import org.jsoup.Jsoup;  
import org.jsoup.nodes.Document;  
import org.jsoup.nodes.Element;  
import org.jsoup.select.Elements;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.stereotype.Service;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
  
@Service  
public class EbayService {  
  
 private static final Logger *logger* = LoggerFactory.*getLogger*(EbayService.class);  
  
 public List<EbayItem> getEbayData(String searchQuery) {  
 String url = "https://www.ebay.com/sch/i.html?\_nkw=" + searchQuery;  
 List<EbayItem> items = new ArrayList<>();  
  
 try {  
 *logger*.info("Connecting to eBay with search query: {}", searchQuery);  
 Document doc = Jsoup.*connect*(url).get();  
 Elements itemElements = doc.select(".s-item");  
  
 for (Element item : itemElements) {  
 String title = item.select(".s-item\_\_title").text();  
 String price = item.select(".s-item\_\_price").text();  
  
 if (!title.isEmpty() && !price.isEmpty()) {  
 items.add(new EbayItem(title, price));  
 }  
 }  
  
 if (items.isEmpty()) {  
 throw new IOException("No items found.");  
 }  
  
 } catch (IOException e) {  
 *logger*.error("Error while parsing eBay data", e);  
 throw new RuntimeException("Error parsing eBay data: " + e.getMessage());  
 }  
  
 *logger*.info("Found {} items", items.size());  
 return items;  
 }  
}

**ExcelService.java**

package com.anastasiia.spring\_musiienko.service;  
  
  
import com.anastasiia.spring\_musiienko.dto.EbayItem;  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import org.springframework.stereotype.Service;  
  
import java.io.ByteArrayOutputStream;  
import java.io.IOException;  
import java.util.List;  
  
@Service  
public class ExcelService {  
  
 public byte[] generateExcelReport(List<EbayItem> ebayItems) throws IOException {  
 Workbook workbook = new XSSFWorkbook();  
 Sheet sheet = workbook.createSheet("eBay Items");  
  
 Row headerRow = sheet.createRow(0);  
 headerRow.createCell(0).setCellValue("Title");  
 headerRow.createCell(1).setCellValue("Price");  
  
 int rowNum = 1;  
 for (EbayItem item : ebayItems) {  
 Row row = sheet.createRow(rowNum++);  
 row.createCell(0).setCellValue(item.getTitle());  
 row.createCell(1).setCellValue(item.getPrice());  
 }  
  
 for (int i = 0; i < 2; i++) {  
 sheet.autoSizeColumn(i);  
 }  
  
 ByteArrayOutputStream outputStream = new ByteArrayOutputStream();  
 workbook.write(outputStream);  
 workbook.close();  
  
 return outputStream.toByteArray();  
 }  
}

**styles.css**

\* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
}  
  
body {  
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
 background-color: #f4f7fb;  
 color: #333;  
 line-height: 1.6;  
 padding: 20px;  
 margin: 0;  
}  
  
h1 {  
 text-align: center;  
 color: #4a90e2;  
 margin-bottom: 30px;  
}  
  
h2 {  
 color: #333;  
 font-size: 24px;  
 margin-bottom: 15px;  
 text-align: left;  
}  
  
button {  
 background-color: #4a90e2;  
 color: #fff;  
 border: none;  
 padding: 10px 20px;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 16px;  
 transition: background-color 0.3s;  
}  
  
button:hover {  
 background-color: #357ab7;  
}  
  
input[type="text"] {  
 width: 100%;  
 padding: 10px;  
 font-size: 16px;  
 border: 2px solid #ddd;  
 border-radius: 5px;  
 margin-bottom: 20px;  
 transition: border-color 0.3s;  
}  
  
input[type="text"]:focus {  
 border-color: #4a90e2;  
 outline: none;  
}  
  
#dollar-rate {  
 font-size: 18px;  
 margin-top: 15px;  
 color: #4a90e2;  
}  
  
#ebay-items {  
 list-style-type: none;  
 padding: 0;  
 margin-top: 10px;  
}  
  
#ebay-items li {  
 background-color: #fff;  
 margin-bottom: 10px;  
 padding: 15px;  
 border-radius: 5px;  
 box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);  
 font-size: 16px;  
}  
  
#ebay-items li span {  
 display: block;  
 font-weight: bold;  
 color: #333;  
}  
  
hr {  
 border: 0;  
 border-top: 1px solid #ddd;  
 margin: 30px 0;  
}  
  
*/\* Responsive Design \*/*@media (max-width: 768px) {  
 body {  
 padding: 10px;  
 }  
  
 input[type="text"] {  
 font-size: 14px;  
 }  
  
 button {  
 font-size: 14px;  
 padding: 8px 15px;  
 }  
  
 #dollar-rate {  
 font-size: 16px;  
 }  
  
 h2 {  
 font-size: 20px;  
 }  
}

**index.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Musiienko App</title>  
 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
 <link rel="stylesheet" href="/css/styles.css">  
</head>  
<body>  
<h1>💻 Welcome to Musiienko Search))</h1>  
  
<div>  
 <h2>💵 Check USD Exchange Rate</h2>  
 <button id="get-dollar-rate">Show Current Dollar Rate</button>  
 <p id="dollar-rate"></p>  
</div>  
  
<hr>  
  
<div>  
 <h2>🔍 Search Items on eBay</h2>  
 <input type="text" id="ebay-search-query" placeholder="Type a product name...">  
 <button id="parse-ebay">Search</button>  
 <ul id="ebay-items"></ul>  
</div>  
  
<hr>  
  
<div>  
 <h2>📄 Export Results to Excel</h2>  
 <input type="text" id="excel-search-query" placeholder="Enter product to export...">  
 <button id="download-excel">Download Excel File</button>  
</div>  
  
<script>  
 $(document).ready(function() {  
 $('#get-dollar-rate').click(function() {  
 $.get('/get-dollar-rate', function(data) {  
 $('#dollar-rate').text(data);  
 });  
 });  
  
 $('#parse-ebay').click(function() {  
 const searchQuery = $('#ebay-search-query').val();  
 $.get('/parse-ebay', { searchQuery: searchQuery }, function(data) {  
 $('#ebay-items').empty();  
 if (data.length === 0) {  
 $('#ebay-items').append('<li>No items found</li>');  
 } else {  
 data.forEach(function(item) {  
 $('#ebay-items').append('<li>' + item.title + ' – ' + item.price + '</li>');  
 });  
 }  
 });  
 });  
  
 $('#download-excel').click(function() {  
 const searchQuery = $('#excel-search-query').val();  
 window.location.href = '/download-excel?searchQuery=' + searchQuery;  
 });  
 });  
</script>  
</body>  
</html>