

**Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
Факультет «Інформаційні системи та технології»
Кафедра «Інформаційні технології та комп'ютерна інженерія»**



**ЗВІТ
про виконання практичної роботи №1
з дисципліни
«Веб-застосунки з Java/Spring»**

Виконав:
студент гр. 126м-24-1

Оленченко Г.М.
(П.І.Б.)

Прийняв:

доцент каф. САУ
Мінєєв О.С.
(П.І.Б.)

**Дніпро
2025**

Розробка програми мовою Java на базі фреймворку SpringBoot.

Завдання

Розробити програму мовою Java на базі фреймворку Springboot. Спілкування клієнта/сервера реалізувати через REST. Тематика та напрям програми – за вибором студента. Обрано наступне – «Парсер магазину Touch». Необхідно задіяти наступні речі:

- Додаток повинен вивантажувати дані у форматі Excel
- Додаток повинен брати/парсити інформацію з інтернету
- Додаток повинен брати інформацію по відкритому Rest API (наприклад курс валют)
- Додаток повинен мати FrontEnd частину
- Додаток повинен мати спілкування з БД (наприклад H2).

Хід роботи

Проект створюється з наступними характеристиками:

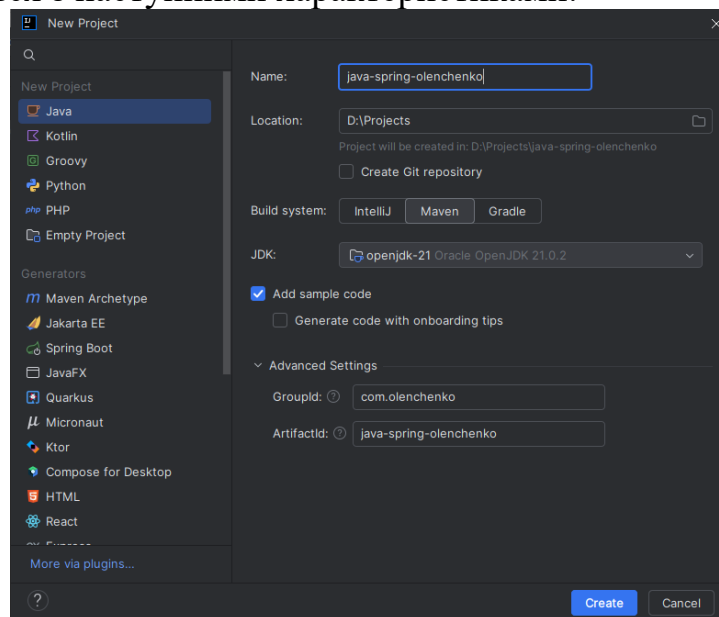


Рисунок 1.1 – Створення проекту

У файл pom.xml додамо необхідні залежності:

```
<dependencies>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.36</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <!-- jsoup HTML parser library @ https://jsoup.org/ -->
    <groupId>org.jsoup</groupId>
```

```

        <artifactId>jsoup</artifactId>
        <version>1.18.3</version>
    </dependency>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.11.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi</artifactId>
        <version>5.4.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi-ooxml</artifactId>
        <version>5.4.0</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
    <!-- providing additional data types that not natively supported by jpa -->
    <dependency>
        <groupId>com.vladmihalcea</groupId>
        <artifactId>hibernate-types-60</artifactId>
        <version>2.21.1</version>
    </dependency>
</dependencies>

```

Для можливості роботи зі SpringBoot необхідно надати «структуру» проекту наступним чином:

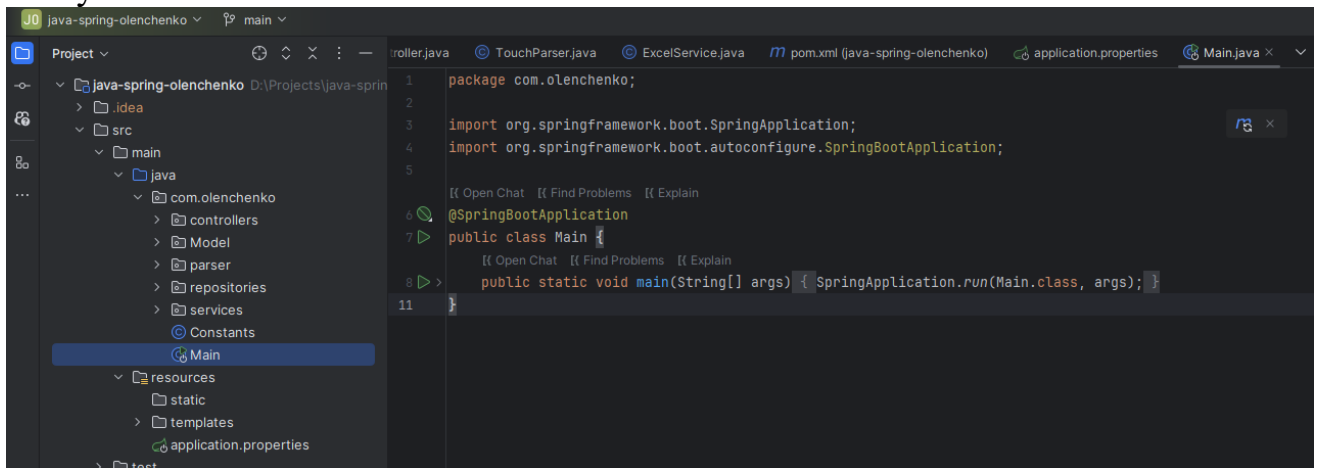


Рисунок 1.2 – Структура проекту

У файлі Main в головному методі відбувається запуск SpringApplication. Для налаштування параметрів застосунку необхідно заповнити файл application.properties:

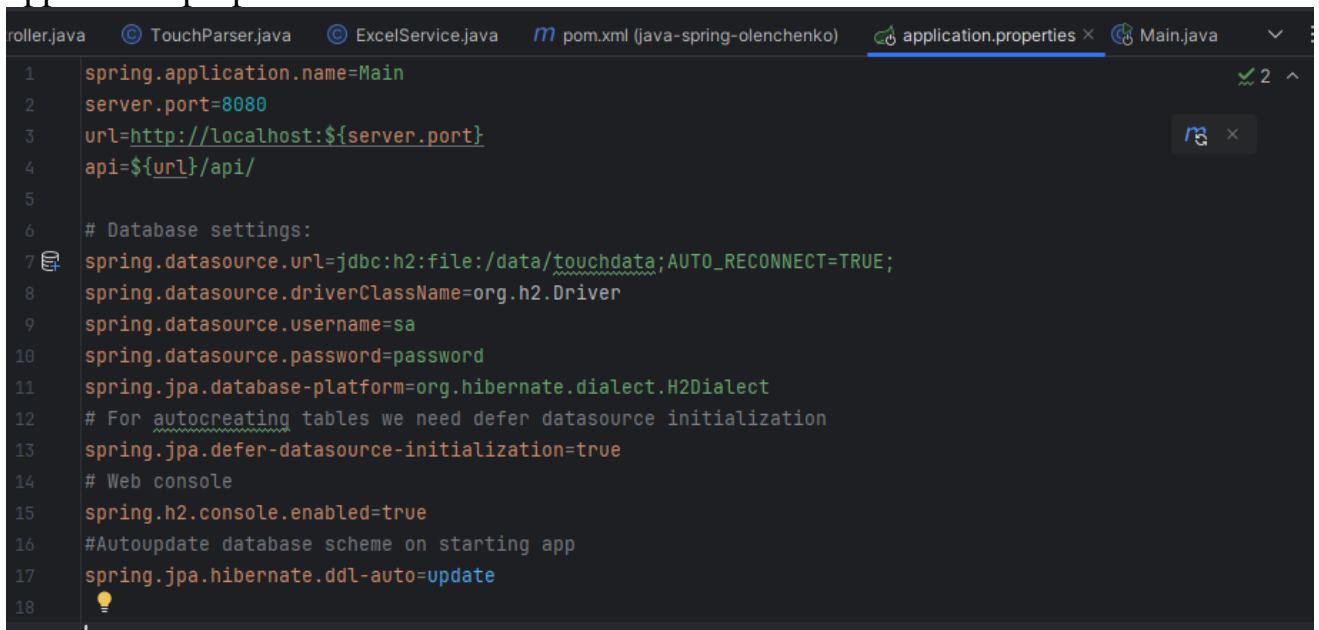


Рисунок 1.3 – Файл конфігурацій

Для уникнення ручного створення необхідних таблиць БД H2, задамо параметри spring.jpa.hibernate.ddl-auto та spring.jpa.defer-datasource-initialization відповідно до рисунку 1.3.

У папці resources створимо дві папки – static та templates (для статичних файлів, таких як стилі та скрипти, та шаблони html сторінок відповідно). Серед головного з роботою шаблонами – отримання даних та робота з ними відбувається за допомогою атрибутів th:**** (рисунок 1.4 а)), попередньо додавши в тег html xmlns:th="http://www.thymeleaf.org", передача даних в шаблон через контролери (рисунок 1.4 б)).

```

<div class="container mt-3">
  <h1>Touch напcep</h1>
  <div class="mb-3">
    <a href="/search" class="btn btn-primary">Бажаєте шукати товар? Натисніть сюди.</a>
  </div>
  <div class="product-container" th:each="categories : ${mergedData}">
    <h2 class="section-title" th:text="${categories.key}"></h2>
    <div id="productsCarousel" class="carousel slide d-flex" data-bs-ride="carousel">
      <div class="carousel-inner">

```

а)

```

61      @GetMapping("/search")
62      public String search(Model model) {
63          model.addAttribute("apiUrl", apiUrl);
64          return "search";
65      }
66  }

```

б)

Рисунок 1.4 – Робота з даними шаблону

Контролери – по суті є швидкими «створювачами» точок входу (сторінок сайту). Загалом їх структура виглядає наступним чином:

```

16  @RestController
17  @RequestMapping("/api")
18  public class ApiController {
19
20      ExcelService excelService; 2 usages
21      ProductRepository productRepository; 5 usages
22
23      TouchParser touchParser; 10 usages
24      private final List<String> sortFields = List.of("SHOWS", "PRICE_ASC", "PRICE_DESC", "DATE"); 1 usage
25
26      @Autowired
27      public ApiController(TouchParser touchParser, ExcelService excelService, ProductRepository productRepository) {
28          this.excelService = excelService;
29          this.touchParser = touchParser;
30          this.productRepository = productRepository;
31      }
32
33      @GetMapping(value = "/newproducts", produces = "application/json")
34      public String getMainPage() {
35          Gson gson = new Gson();
36          return gson.toJson(touchParser.getNewProducts());
37      }
38  }

```

Рисунок 1.5 – Структура контролерів

Анотацією `RestController` ми зазначаємо, що це клас контролер, `RequestMapping` – частина адреси (після `localhost:8080` у випадку локальної роботи), за якою даний

контролер знаходиться. Autowired – використовується для автоматичного додання залежностей компонентів. GetMapping – для роботи з конкретним запитом (налаштовується адреса, тип контенту-відповіді тощо).

Розглянемо загальну структуру проекту. Розроблено два контролери – для /api/ ендпоінтів та фронтендових сторінок, дві моделі – ProductCard (коротка інформація про товар), яку наслідує Product (доповнює інформацію). Папка Services – містить сервіс для роботи з XLSX файлами (для збереження інформації про товар в файл XLSX), parser містить в собі головний функціонал програми – копіювання інформації з обраного сайту. ProductRepository – інтерфейс, створений для роботи з базою даних. Constants – містить статичні дані, які не можуть бути змінені.

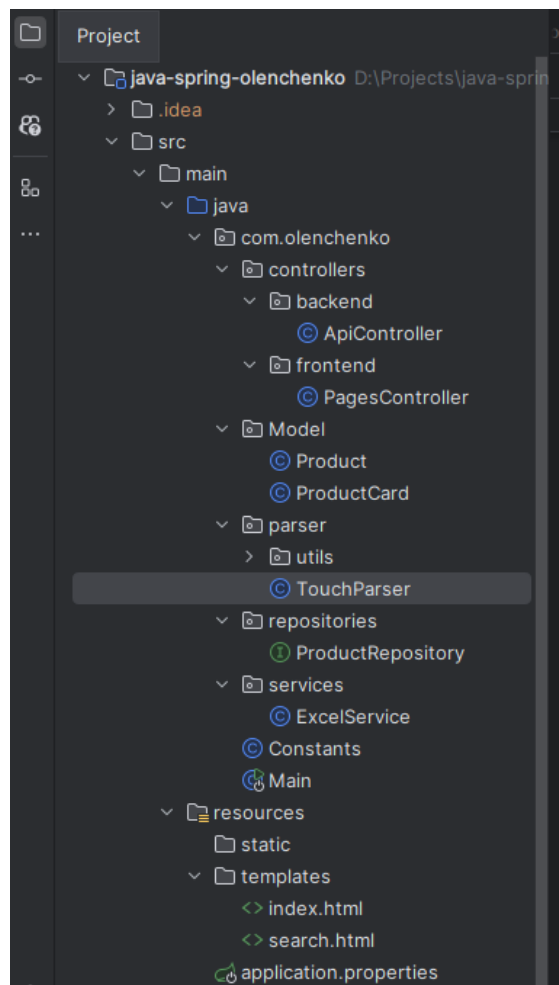


Рисунок 1.6 – Загальна структура проекту

TouchParser – головний клас парсера. Отримує, обробляє, віддає дані у вигляді списку ProductCard. У випадку запиту інформації про конкретний товар, надається об'єкт Product (завантажується файл product.xlsx). Задля зменшення кількості запитів, що надходять на сайт, даний об'єкт зберігається у базу даних у дві таблиці – PRODUCT_CARDS ТА PRODUCTS. Це реалізується завдяки зазначенню спеціальних анотацій (рисунок 1.7). У об'єктах, що зберігаються в базу даних, наявні HashMaps, що не підтримуються за замовчуванням, тому для вирішення

проблеми з їх додаванням в БД використовується бібліотека com.vladmihalcea.hibernate-types-60, що додає додаткові типи даних, і зберіганням варіацій та характеристик як Json строк.

The screenshot shows a database management interface. On the left, a tree view displays the database schema: jdbc:h2:file:/data/touchdata, PRODUCTS (DESCRIPTION, PROPERTIES, ID), PRODUCT_CARDS (ID, ARTICLE, IMAGE_URL, PRICE_INUSDWITH_DISCOUNT, PRICE_INUSDWITHOUT_DISCOUNT, PRICE_WITHOUT_DISCOUNT, TITLE, URL, VARIATIONS), INFORMATION_SCHEMA, and Users. The main area shows a SQL query: SELECT * FROM PRODUCT_CARDS. Below the query, the results are displayed in a table with 4 rows and 5 columns: ID, ARTICLE, IMAGE_URL, PRICE_INUSDWITH_DISCOUNT, and PRICE_INUSDWITHOUT_DISCOUNT. The first row shows ID 125230, ARTICLE https://touch.com.ua/upload/resize_cache/webp/block/81a/d4em0sua3c3wzh4m95tydnr1z0c0la5.webp, IMAGE_URL https://touch.com.ua/upload/resize_cache/webp/block/896/lp5zhnqwk4gpyta8gfr9tc3r2c99g.webp, PRICE_INUSDWITH_DISCOUNT 109.91611099161109, and PRICE_INUSDWITHOUT_DISCOUNT 0.0.

ID	ARTICLE	IMAGE_URL	PRICE_INUSDWITH_DISCOUNT	PRICE_INUSDWITHOUT_DISCOUNT
1	125230	https://touch.com.ua/upload/resize_cache/webp/block/81a/d4em0sua3c3wzh4m95tydnr1z0c0la5.webp	109.91611099161109	0.0
2	104433	https://touch.com.ua/upload/resize_cache/webp/block/896/lp5zhnqwk4gpyta8gfr9tc3r2c99g.webp	38.21610382161038	42.99610429961043
3	109197	https://touch.com.ua/upload/resize_cache/webp/block/483/1kpwtdn78h9gtem8xjdjm777d8wz9mn.webp	268.13412681341265	329.7961329796133
4	51374	https://touch.com.ua/upload/resize_cache/webp/block/a38/a38c7e65d8d075b7a395cc613612e490.webp	739.6811739681174	884.2761884276189

a)

```

14  // Open Chat  // Find Problems  // Explain
15  @Entity 34 usages
16  @Table(name = "products")
17  @NoArgsConstructor
18  @Getter
19  @Setter
20  public class Product extends ProductCard {
21  // Open Chat  // Find Problems  // Explain
22  @Column(columnDefinition = "TEXT")
23  private String description;
24  // Open Chat  // Find Problems  // Explain
25  @Type(JsonType.class)
26  @Column(columnDefinition = "json")
27  private HashMap<String, String> properties;
28
29  // Open Chat  // Find Problems  // Explain
30  @Override
31  public String toString() {
32      return "Product{" +
33          "description='" + description + '\'' +
34          ", properties=" + properties +
35          "} " + super.toString();
36  }
37
38  // Open Chat  // Find Problems  // Explain
39  @Override
40  public boolean equals(Object o) {

```

б)

```

14  // Open Chat  // Find Problems  // Explain
15  @Entity 25 usages 1 inheritor
16  @Table(name = "product_cards")
17  @Inheritance(strategy = InheritanceType.JOINED)
18  @NoArgsConstructor
19  @Getter
20  @Setter
21  public class ProductCard {
22  // Open Chat  // Find Problems  // Explain
23  @Id
24  @GeneratedValue(strategy = GenerationType.IDENTITY)
25  private Long id;
26  @Column
27  private int article;
28  @Column
29  private String title;
30  @Column
31  private String imageUrl;
32  @Column
33  private String url;
34  @Column
35  private double priceWithDiscount;
36  @Column
37  private double priceInUSDWithDiscount;
38  @Column
39  private double priceWithoutDiscount;
40  @Column
41  private double priceInUSDWithoutDiscount;
42
43  // Open Chat  // Find Problems  // Explain
44  @Type(JsonType.class)
45  @Column(columnDefinition = "json")
46  private HashMap<String, List<HashMap<String, String>>> variations;

```

в)

Рисунок 1.7 – Робота з базою даних

Файл XLSX, що формується, має наступний вигляд:

	A	B	C	D	E	F	G	H	I	J	K
1	Назва товару	Ноутбук Apple MacBook Air 13" Space Gray Late 2020 (MGN63)									
2	Опис	Apple MacBook Air 13" 2020 с чіпом M1. З новою силою. З появою чіпа Apple M1 найтонший і легкий ноутбук повністю змінився. Центральний процесор тепер працює до 3,5 рази швидше. Графічний - до 5 разів. А завдяки передовій системі Neural Engine швидкість машинного навчання зросла до 9 разів. Новий MacBook Air працює без підзарядки довше, ніж попередні моделі. І зовсім не шумить, тому що у нього немає вентилятора. Потужність ще ніколи не була такою компактною. Маленький чіп. Грандіозний прорив. Зустрічати. Перший чіп, розроблений спеціально для Mac. Вражає, але система на чіпі Apple M1 вміщує 16									
3	Ціна	Без знижки: 36999.02 (884,28\$)					Зі знижкою: 30949.02 (739,68\$)				
4	Посилання	https://touch.com.ua/ua/item/apple-macbook-air-13-space-gray-late-2020-mgn63/									
5	Артикул	51374									
6	Характеристики	Кількість ядер			8						
7		Акумулятор			49,9 Вт/год						
8		Гарантійний термін			3 міс.						
9		Колір			Сірий космос						
10		Діагональ екрану			13.3"						
11		Відвантаження			Сьогодні						
12		Бренд			Apple						
13		Тип екрану			IPS						
14		Об'єм накопичувача			256ГБ						
15		Вага			1.29 кг						
16		Тип оперативної пам'яті			DDR4						
17		Підсвічування клавіатури			Є						
18		Графічний чіп			Apple M1						
19		Тип ноутбука			Ультрабук						
20		Процесор			Apple M1						
21		Бездротовий зв'язок			Wi-Fi 6 (802.11ax), Bluetooth 5.0						
22		Тип відеокарти			Інтегрована						
23		Об'єм оперативної пам'яті			8ГБ						
24		Серія			Apple MacBook Air 13 M1 (2020)						
25		Виробник відеокарти			Apple						
26		Розміри			304.1 x 212.4 x 4.1–16.1 мм						
27		Роздільна здатність екрану			WQXGA (2560x1600)						
28		Роз'єми та порти			2x Thunderbolt 3 (USB 4), 3.5 mm headphone jack						
29		Частота процесора			2.0 - 3.2 ГГц						
30		Короткі характеристики			Ноутбук • Класичний • 13,3 • IPS • 2560x1600 • Apple M1 •						
31		Тип			Ноутбуки						
32		Сумісна модель			Apple MacBook Air 13 M1 (2020)						
33	Варіації	Вбудована пам'ять	256ГБ			https://touch.com.ua/ua/item/apple-macbook-air-13-					
34			512ГБ			https://touch.com.ua/ua/item/apple-macbook-air-13-					
35		Колір	Золотий (Gold)			https://touch.com.ua/ua/item/apple-macbook-air-13-					
36			Сірий (Space Gray)			https://touch.com.ua/ua/item/apple-macbook-air-13-					
37			Сріблястий (Silver)			https://touch.com.ua/ua/item/apple-macbook-air-13-					
38		Оперативна пам'ять	8ГБ			https://touch.com.ua/ua/item/apple-macbook-air-13-					
39			16ГБ			https://touch.com.ua/ua/item/apple-macbook-air-13-					

Рисунок 1.8 – Вигляд XLSX файлу

Для надання стилю клітинкам використовується CellStyle (причому оновити деякі параметри стилю клітинки неможливо – потрібно замінювати стиль новим), об'єднання клітинок за допомогою `sheet.addMergedRegion(new CellRangeAddress(rowNum1, rowNum2, colNum1, colNum2))`;

На рисунках 1.9-1.10 наведено знімки екрану головної сторінки та сторінки пошуку товарів відповідно.

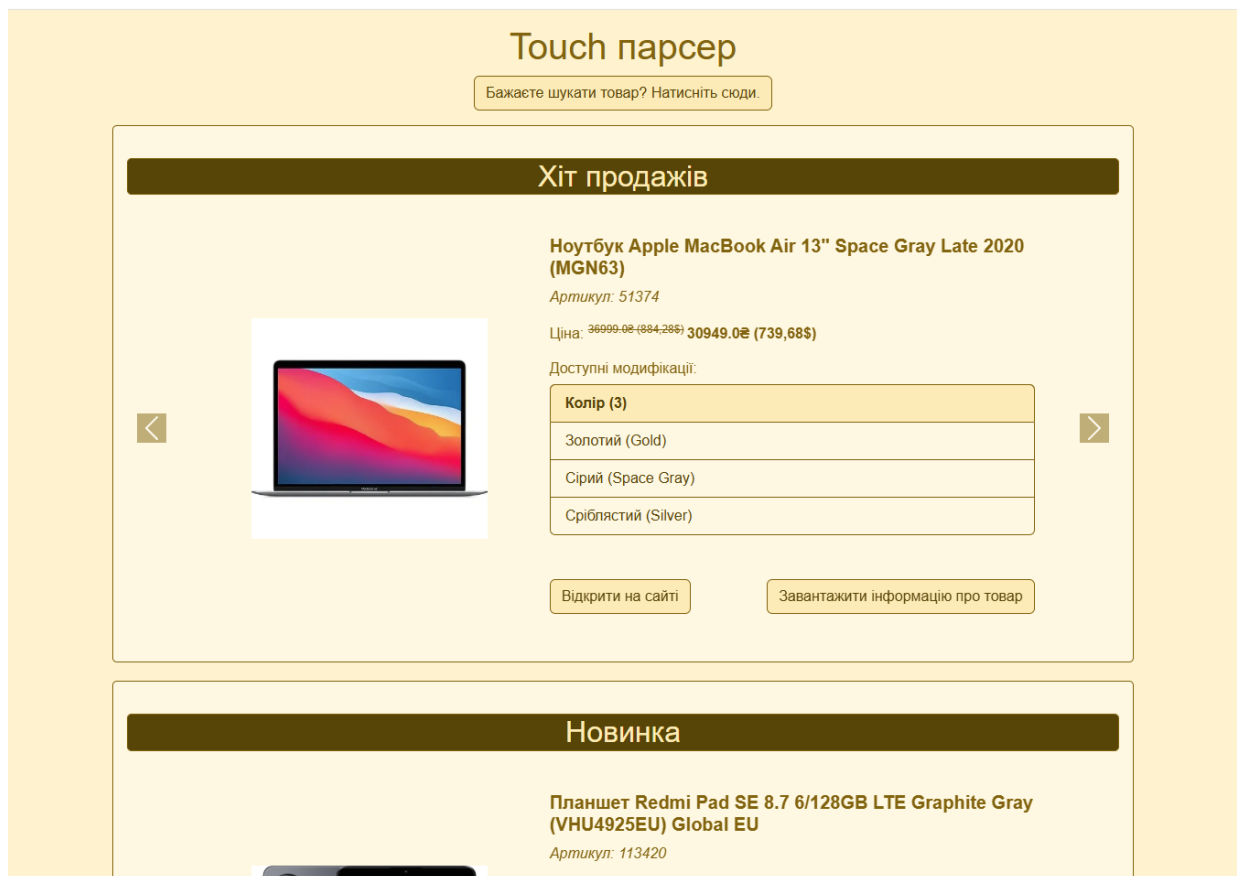


Рисунок 1.9 – Головна сторінка сайту

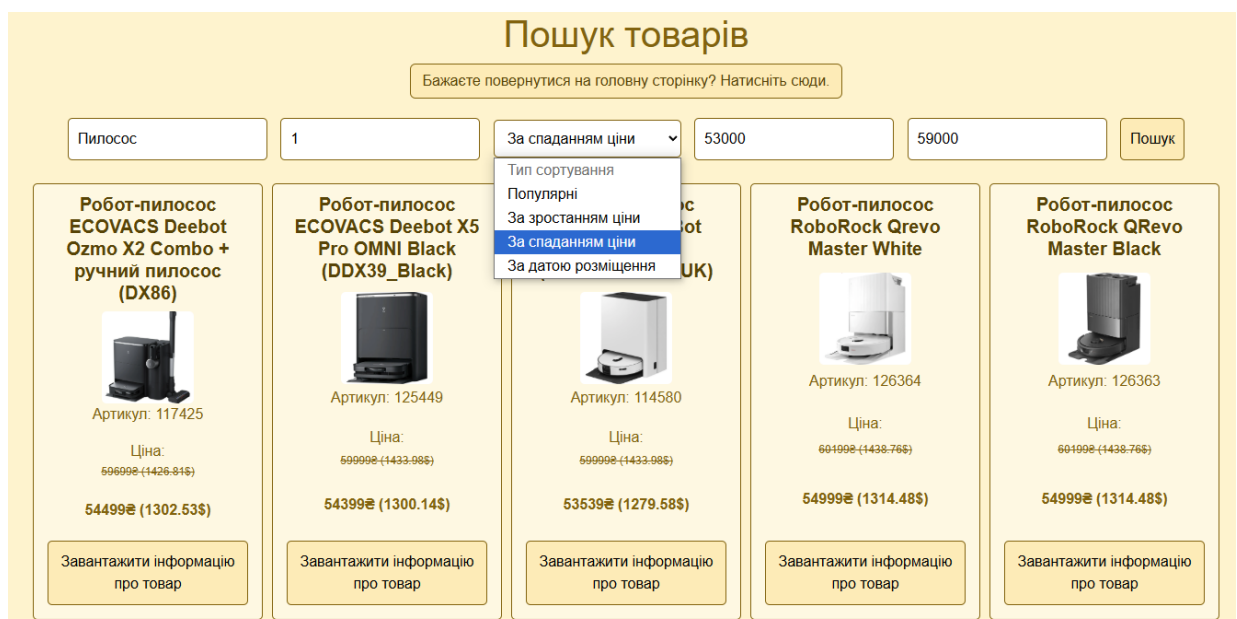


Рисунок 1.10 – Сторінка пошуку товарів

Висновки: під час виконання практичної роботи №1, було розроблено програму мовою Java на базі фреймворку Springboot, що отримує інформацію з сайту інтернет-магазину та надає користувачеві в зручній формі. Реалізований функціонал завантаження даних про товар та кешування даних про нього в базі даних. Повний вихідний код додається архівом до звіту та доступний публічно в

репозиторії за посиланням <https://github.com/golenchenko/java-spring-olnenchenko>.
Попередньо зкопільовані версії програми можливо завантажити за посиланням <https://github.com/golenchenko/java-spring-olnenchenko/releases> .