Звіт з курсу

**«Розробка клієнт-серверних застосувань на Java»**

На тему:  
**«Автоматизоване робоче місце»**

Виконали:  
 студентки БП ІПЗ-3  
Руденко Тетяна та  
Сліпушкіна Олександра

Викладач:  
Круп’як Олег Михайлович

**1. Загальний опис проєкту**

Цей проєкт — це веб-додаток, який допомагає автоматизувати роботу складу на невеликому підприємстві. Він дозволяє зручно зберігати інформацію про товари, керувати їх надходженням і списанням, а також аналізувати стан запасів**.**

**Основні функціональні можливості:**

* Управління групами товарів (створення, редагування, видалення)
* Управління товарами (додавання, редагування, видалення)
* Операції зі складом (надходження, списання товарів)
* Пошук товарів та груп за ключовими словами
* Розрахунок загальної вартості товарів
* Система авторизації та аутентифікації користувачів

**2. Архітектура системи**

**2.1 Загальна архітектура**

Проєкт побудований за архітектурою MVC (Model-View-Controller) з використанням Spring Boot Framework:

* **Model** - ентіті та DTO для роботи з даними
* **View** - веб-інтерфейс (HTML-шаблони)
* **Controller** - контролери для обробки HTTP-запитів

**2.2 Технологічний стек**

* **Backend**: Java 8+, Spring Boot, Spring MVC, Spring Security, Spring Data JPA
* **Frontend**: HTML, CSS, JavaScript, Thymeleaf
* **База даних**: JPA/Hibernate (підтримка різних СУБД)
* **Безпека**: Spring Security, BCrypt для хешування паролів

**3. Структура проєкту**

**3.1 Основні компоненти**

**Контролери:**

* **ApplicationController** - основний контролер для роботи з веб-інтерфейсом
* **AuthorizationController** - контролер авторизації та реєстрації
* **GroupRestController** - REST API для роботи з групами товарів
* **ApplicationErrorController** - обробка помилок

**Моделі даних:**

* **Group** - група товарів
* **Product** - товар
* **User** - користувач системи
* **Role** - роль користувача
* **Response<T>** - обгортка для відповідей з обробкою помилок

**4. Детальний опис функціональності**

**4.1 Управління групами товарів**

**Ентіті Group:**

java

@Entity(name = "groups")

public class Group {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private long id;

@Column(unique = true, nullable = false)

private String name;

@Column(nullable = false)

private String description;

@OneToMany(mappedBy = "group", cascade = CascadeType.ALL, orphanRemoval = true)

private List<Product> products;

}

**Основні операції:**

* Створення нової групи товарів
* Редагування існуючої групи
* Видалення групи (каскадне видалення всіх товарів)
* Пошук груп за ключовими словами

**4.2 Управління товарами**

**Ентіті Product:**

java

@Entity(name = "products")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private long id;

@Column(unique = true, nullable = false)

private String name;

@ManyToOne(fetch = FetchType.EAGER)

private Group group;

@Column(nullable = false)

private String producer;

@Column(nullable = false)

private double price;

@Column(nullable = false)

private long amount;

@Column(nullable = false)

private String description;

}

**Основні операції:**

* Додавання нового товару до групи
* Редагування параметрів товару
* Видалення товару
* Операції зі складом (надходження/списання)

**4.3 Складські операції**

**Надходження товару:**

java

public String acceptMoreProduct(@ModelAttribute("productID") long id,

@ModelAttribute("arrivedAmount") long amount,

Model model) {

Response<Product> productResponse = productService.addProductAmount(id, amount);

}

**Списання товару:**

java

public String withdrawProductPage(@ModelAttribute("productID") long id,

@ModelAttribute("withdrawnAmount") long amount,

Model model) {

Response<Product> productResponse = productService.withdrawProduct(id, amount);

}

**4.4 REST API**

**Endpoints для роботи з продуктами:**

* POST /product/receive - надходження товару
* POST /product/consume - списання товару
* GET /products/search - пошук товарів

**Endpoints для роботи з групами:**

* GET /api/groups - отримання всіх груп
* POST /api/groups - створення нової групи
* PUT /api/groups/{id} - оновлення групи
* DELETE /api/groups/{id} - видалення групи

**4.5 Система авторизації**

**Користувач (User):**

java

@Entity(name = "users")

public class User implements UserDetails {

private long id;

private String email;

private String username;

private String password;

private Set<Role> roles;

}

**Конфігурація безпеки:**

Для забезпечення безпеки в системі використано Spring Security, що включає хешування паролів за допомогою алгоритму BCrypt, захист усіх маршрутів від неавторизованого доступу та реалізацію форми авторизації з автоматичним перенаправленням користувачів.

**5. База даних**

**5.1 Схема бази даних**

**Таблиці:**

* groups - групи товарів
* products - товари
* users - користувачі
* user\_roles - ролі користувачів

**Зв'язки:**

* products.group\_id → groups.id (Many-to-One)
* users.roles ↔ user\_roles (Many-to-Many)

**5.2 Репозиторії**

**GroupRep:**

java

public interface GroupRep extends JpaRepository<Group, Long> {

Group findGroupById(long id);

Group findGroupByName(String name);

List<Group> searchGroups(@Param("keyword") String keyword);

}

**ProductRep:**

java

public interface ProductRep extends JpaRepository<Product, Long> {

Product findProductById(long id);

Product findProductByName(String name);

List<Product> findAllByGroup(Group group);

BigDecimal sumForList(List<Long> productIds);

List<Product> searchProducts(@Param("keyword") String keyword);

}

**6. Пошук та фільтрація**

**6.1 Пошук товарів**

Реалізований пошук по всіх полях товару:

* Назва товару
* Опис товару
* Виробник

**6.2 Пошук груп**

Пошук по полях групи:

* Назва групи
* Опис групи

**6.3 Комбінований пошук**

Можливість пошуку товарів з фільтрацією по:

* Групі товарів
* ID товару
* Частині назви товару

**7. Статистичні дані**

**7.1 Розрахунок вартості**

java

@Query(value = "select sum(p.amount \* p.price) from products p where p.id in ?1",

nativeQuery = true)

BigDecimal sumForList(List<Long> productIds);

**8. Валідація даних**

**8.1 Валідація на рівні моделі**

У системі реалізовано валідацію даних, яка включає перевірку унікальності назв товарів та груп, обов’язковість заповнення ключових полів, а також контроль коректності введених числових значень.

**8.2 Обробка помилок**

Використання класу Response<T> для уніфікованої обробки помилок:

java

public class Response<T> {

private List<String> errorMessage;

private T object;

public boolean isOkay() {

return errorMessage.size() == 0;

}

}

**9. Безпека**

**9.1 Аутентифікація**

Додаток містить форму логіну з перевіркою облікових даних користувача, використовуючи хешування паролів за допомогою алгоритму BCrypt. Для забезпечення безпеки доступу реалізована сесійна аутентифікація.

**9.2 Авторизація**

У додатку реалізовано захист усіх маршрутів, крім логіну та реєстрації, з використанням системи ролей користувачів. Неавторизовані користувачі автоматично перенаправляються на сторінку входу.

**10. Веб-інтерфейс**

**10.1 Основні сторінки**

* Головна сторінка - список товарів
* Управління групами товарів
* Додавання/редагування товарів
* Операції зі складом (надходження/списання)
* Сторінки авторизації та реєстрації

**10.2 Функціональність**

* Пошук в реальному часі
* Фільтрація за групами
* Розрахунок загальної вартості
* Responsive дизайн

**11. Особливості реалізації**

**11.1 Каскадні операції**

При видаленні групи товарів автоматично видаляються всі товари з цієї групи завдяки:

java

@OneToMany(mappedBy = "group", cascade = CascadeType.ALL, orphanRemoval = true)

private List<Product> products = new ArrayList<>();

**11.2 Безпека операцій**

Всі операції зміни кількості товарів мають захист від від'ємних значень:

java

public void changeAmount(long change){

this.amount += change;

if (this.amount < 0) this.amount = 0;

}

**11.3 Гнучкість пошуку**

Реалізований гнучкий пошук з підтримкою часткового співпадіння та нечутливості до регістру.

**13. Юніт тести**

Реалізовані юніт тести: GroupServiceImplTest, ProductServiceImplTest

**14. Інтерфейс**

Изображение выглядит как текст, Шрифт, число, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным. **15. Розподіл ролей  
Руденко Тетяна:**Розробила загальну архітектуру системи

1. Реалізувала backend-логіку:
2. Контролери: ApplicationController, GroupRestController, ApplicationErrorController
3. Сервіси для роботи з товарами та групами
4. Репозиторії GroupRep, ProductRep
5. Валідація на рівні моделей
6. Каскадне видалення груп з товарами
7. REST API: надходження, списання, пошук товарів/груп
8. Реалізація складських операцій (надходження/списання)
9. Написала юніт тести (GroupServiceImplTest, ProductServiceImplTest)
10. Реалізувала розрахунок вартості та логіку фільтрації

**Сліпушкіна Олександра:**

1. Розробила frontend-інтерфейс
2. Реалізувала функціональність:
3. Пошук і фільтрація товарів та груп
4. Форма авторизації/реєстрації
5. Веб-сторінки: додавання/редагування товарів, робота з групами
6. Налаштувала систему авторизації та аутентифікації:
7. Spring Security, конфігурація захисту маршрутів
8. Хешування паролів через BCrypt
9. Тестувала функціональність веб-інтерфейсу та інтеграцію з backend

**16. Висновки**

Розроблений веб-додаток успішно реалізує повний цикл автоматизації складських операцій для малого та середнього бізнесу. Завдяки структурованій архітектурі, зручному веб-інтерфейсу та використанню сучасних технологій (Spring Boot, Spring Security, JPA, Thymeleaf), система забезпечує ефективне управління товарами, групами, обліком залишків та аналітикою.

Окрему увагу приділено безпеці: реалізовано систему ролей, авторизацію та аутентифікацію, захист маршрутів і хешування паролів. Продуманий функціонал пошуку, звітності та каскадної обробки даних робить систему гнучкою і зручною в користуванні.

Таким чином, проєкт відповідає поставленим вимогам, має масштабовану структуру та є надійною основою для подальшого розвитку і впровадження в реальному середовищі.