UNIVERSITY of INFORMATION TECHNOLOGY and MANAGEMENT
in Rzeszow, POLAND

FACULTY OF APPLIED INFORMATION TECHNOLOGY

Field of Study: INFORMATION TECHNOLOGY

Specialty: Programming

OLEKSANDR KOKOTKO W67955,

TIMUR ABDUGALILOV W68606

SpringBoot E-Commerce Platform

Introduction

The SpringBoot E-Commerce Platform is a comprehensive web application designed to facilitate online shopping experiences. Developed using the robust Spring Boot framework, this platform leverages Java to create a secure, efficient, and user-friendly environment for both customers and administrators.

This platform features secure registration and login processes for users, including role-based access control for regular users and administrators. It offers a detailed overview of available products with intuitive search and filter functionalities, ensuring customers can easily find and purchase products. The ordering process is simplified with an easy-to-use order form and order tracking capabilities, enhancing the user experience.

For administrators, the platform provides an extensive dashboard with tools to manage products, user roles, and order statuses, ensuring smooth operation and maintenance. Advanced security configurations protect user data and application integrity, including encrypted passwords and customizable access restrictions.

1. Technology Used

1.1 POM.XML

The `pom.xml` file is used in the Maven project for an application written in Java, utilizing the Spring Boot framework. This file contains detailed configuration information for the project, including dependencies, framework versions, and other tools and libraries used in the application.

The project uses Spring Boot version `2.6.1`.

The project is configured with various Maven dependencies, including:

- `spring-boot-starter-thymeleaf`: for creating views based on Thymeleaf.
- `spring-boot-starter-web`: basic support for creating web applications, including RESTful applications.
- `spring-boot-starter-jdbc`: support for JDBC.

- `spring-boot-starter-security`: tools for application security.
- `spring-boot-starter-data-jpa`: support for the Java Persistence API.
- JDBC driver for Oracle (`ojdbc11`).
- Various testing tools (`spring-boot-starter-test`, `spring-security-test`).

The `pom.xml` file also contains information about the Java version used (`java.version` set to 11) and other dependencies such as Lombok (`lombok.version`), Bootstrap (`bootstrap.version`), and various libraries supporting webjars and validation.

1.2 Models

In the Spring Boot project, the models folder serves as the repository for model classes, which represent the application's data and business logic. It contains entity classes where each class represents a database table, and its fields correspond to table columns. Annotations like `@Entity`, `@Table`, `@Id`, and `@Column` are used to specify mappings.

These are later used in controllers, views, and DAO classes.

1.3 DAO Files

Each DAO class typically corresponds to one database table or a specific set of data-related operations. These classes contain methods for performing CRUD (Create, Read, Update, Delete) operations and other application-specific queries.

DAO classes are used to retrieve, update, delete, or add data to the database. This adheres to the Single Responsibility Principle, where the DAO is responsible only for database operations, while business logic is handled in the service layer.

1.4 Controllers

Controllers are crucial in the MVC (Model-View-Controller) architecture. They are responsible for processing incoming client requests, deciding on actions to be performed, and delegating tasks to appropriate application components.

Controllers contain methods that respond to various HTTP requests (GET, POST, PUT, DELETE, etc.). These methods are marked with appropriate annotations such as `@GetMapping` and `@PostMapping`.

The project also includes an exception handling controller to manage errors during request processing.

1.5 Security Configuration

The Security Configuration file is set up to secure the web application by specifying which resources are publicly accessible and which require authentication. For example, paths such as "/", "/index", and "/register" are publicly accessible, allowing access to main and registration pages without logging in.

A `CustomUserDetailsService` is used to define a custom method for retrieving user data. A `PasswordEncoder` (BCryptPasswordEncoder in this case) is used for secure hashing and verifying user passwords.

Different access levels are specified for different user roles (ADMIN, USER), allowing control over access to specific parts of the application. For instance, only users with the "ADMIN" role can access "/main admin".

1.6 Application Properties File

The `application.properties` file in Spring Boot is used to configure various aspects of the application, such as database settings, logging, and other configuration parameters. It also allows disabling the default whitelabel error page in Spring Boot.

1.7 Templates

In the project, the templates folder is used to store template files, which are part of the view layer in the MVC (Model-View-Controller) architecture. These templates are used to represent data in a format understandable and accessible to the end user.

2. Application Presentation

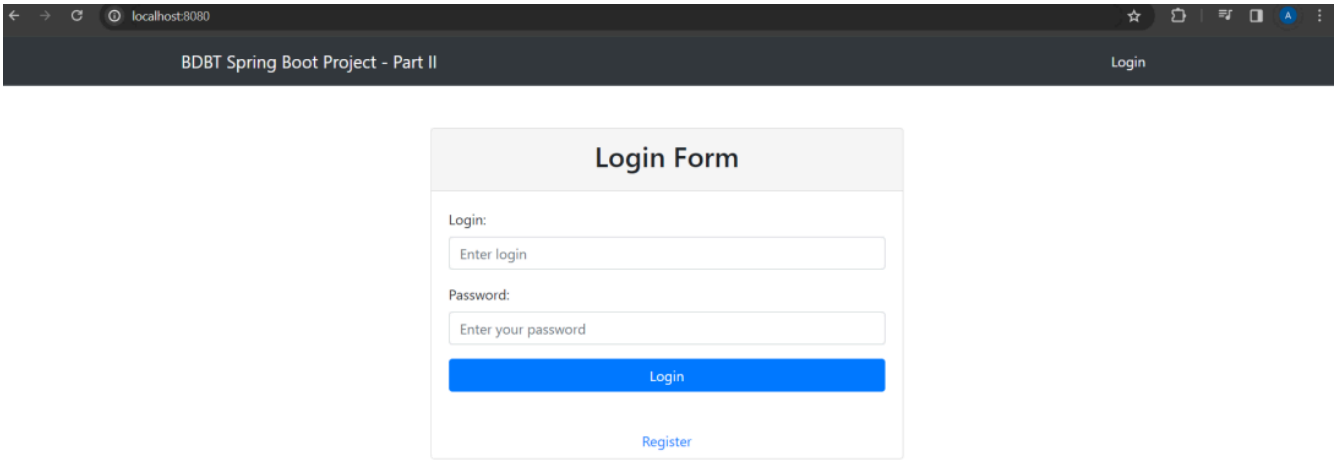


Figure 1: Login menu

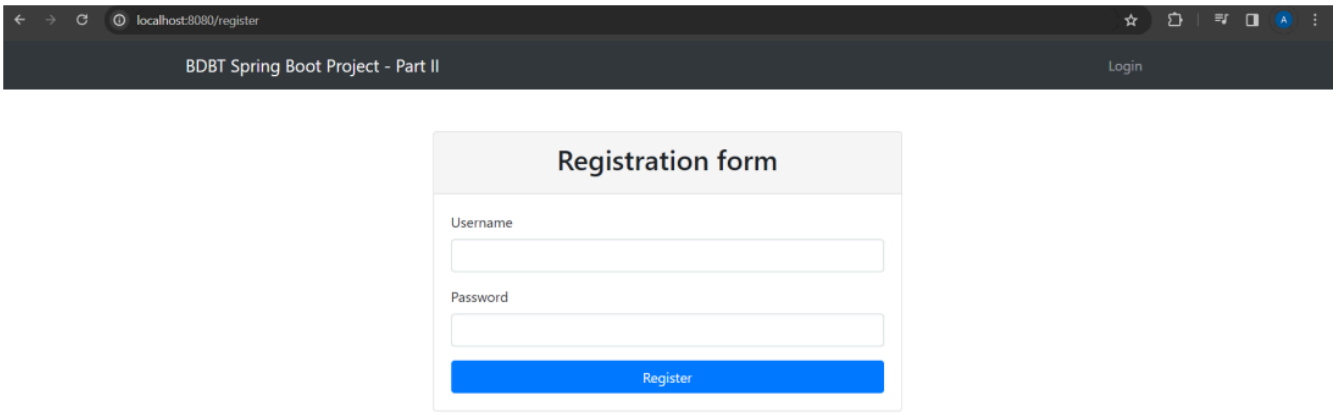


Figure 2: Registration page

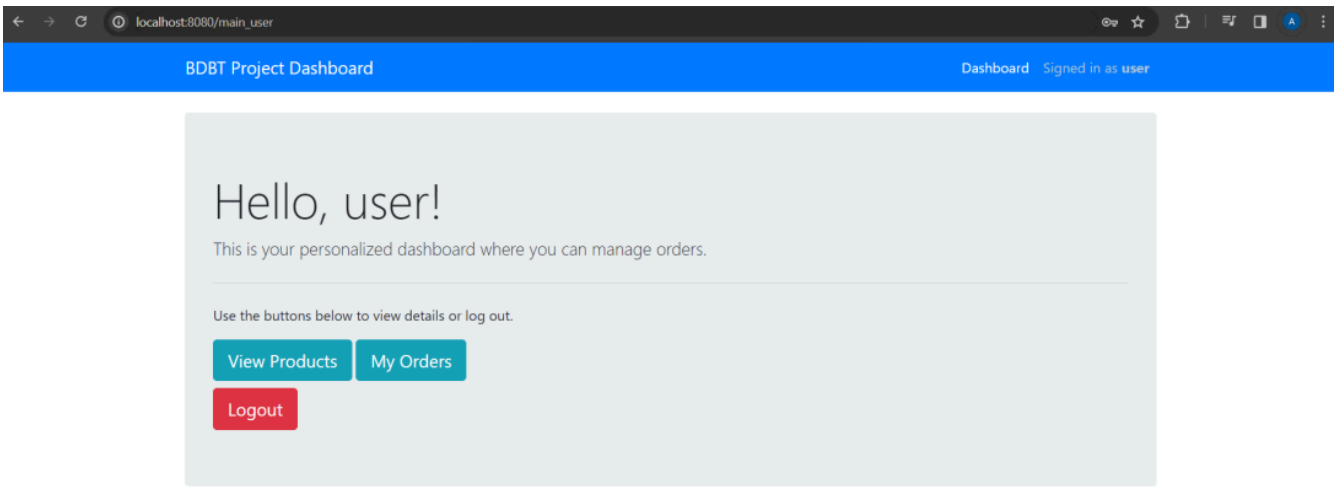
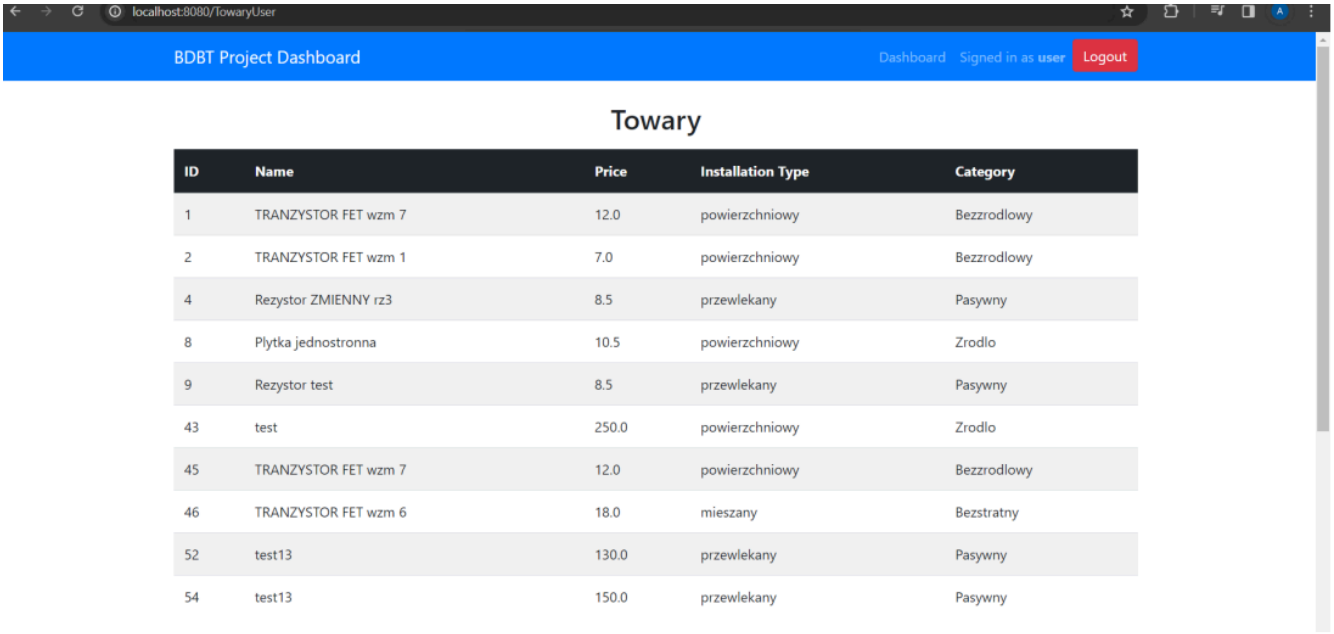


Figure 3: Customer's page



ID	Name	Price	Installation Type	Category
1	TRANZYSTOR FET wzm 7	12.0	powierzchniowy	Bezzrodlowy
2	TRANZYSTOR FET wzm 1	7.0	powierzchniowy	Bezzrodlowy
4	Rezystor ZMIENNY rz3	8.5	przewlekany	Pasywny
8	Plytka jednostronna	10.5	powierzchniowy	Zrodlo
9	Rezystor test	8.5	przewlekany	Pasywny
43	test	250.0	powierzchniowy	Zrodlo
45	TRANZYSTOR FET wzm 7	12.0	powierzchniowy	Bezzrodlowy
46	TRANZYSTOR FET wzm 6	18.0	mieszany	Bezstratny
52	test13	130.0	przewlekany	Pasywny
54	test13	150.0	przewlekany	Pasywny

Figure 4: Available products page

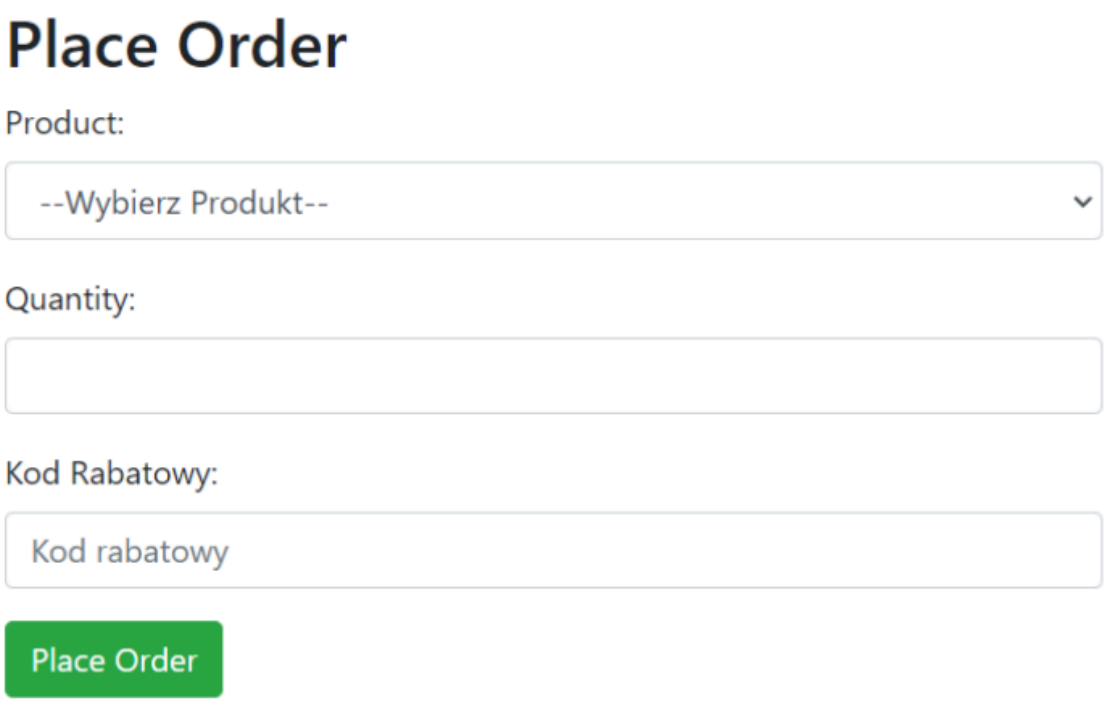


Figure 5: Order menu

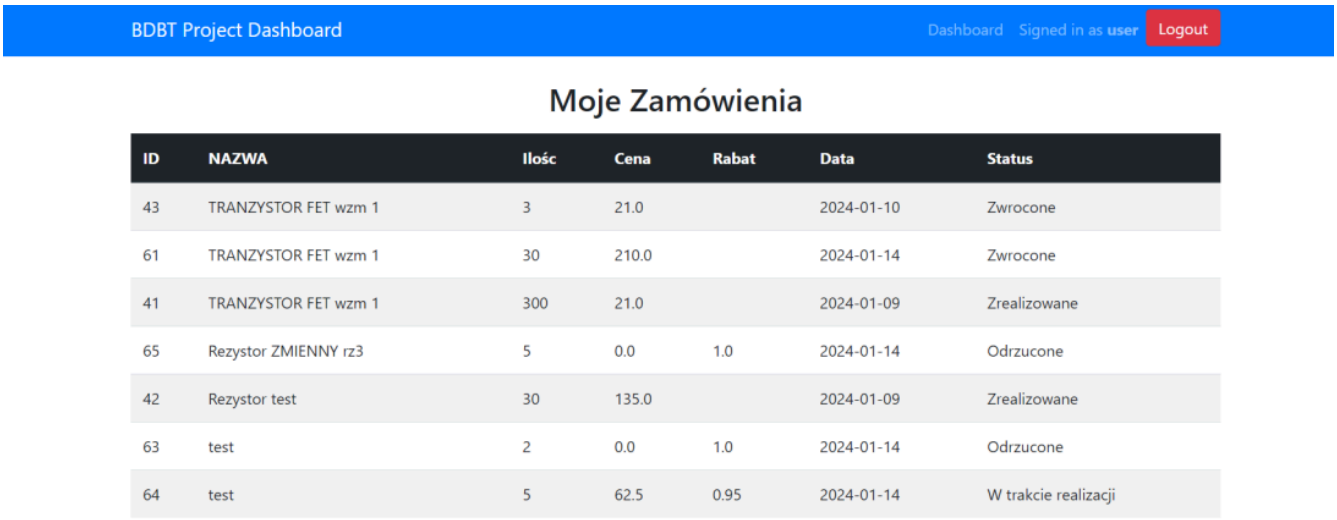


Figure 6: Customer's orders

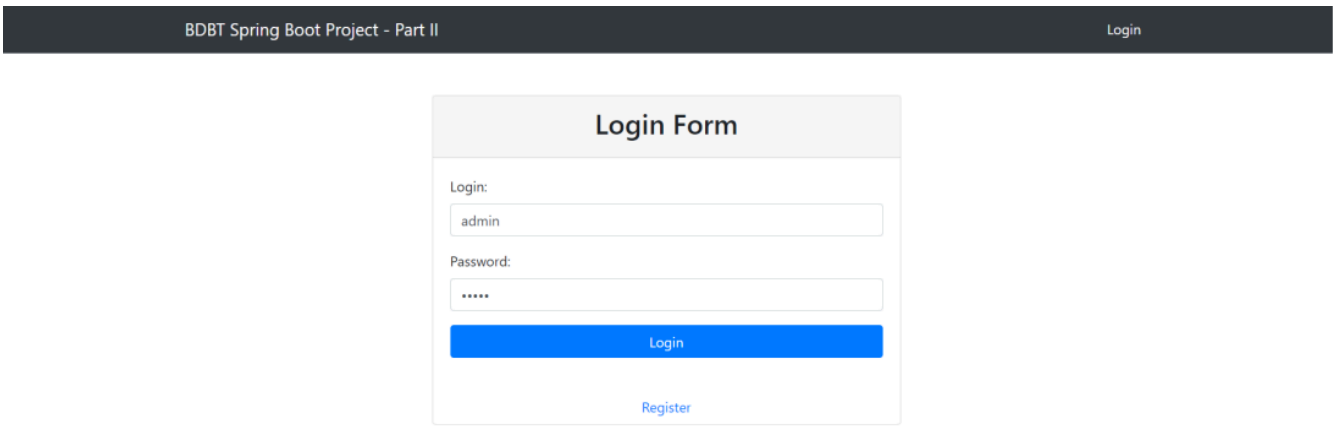


Figure 7: Admin login page

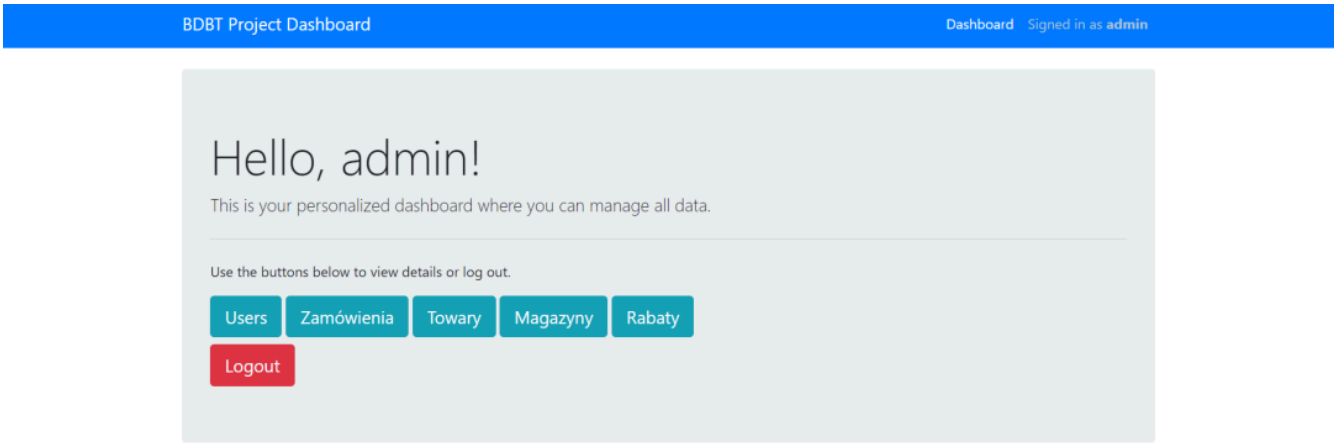


Figure 8: Admin page

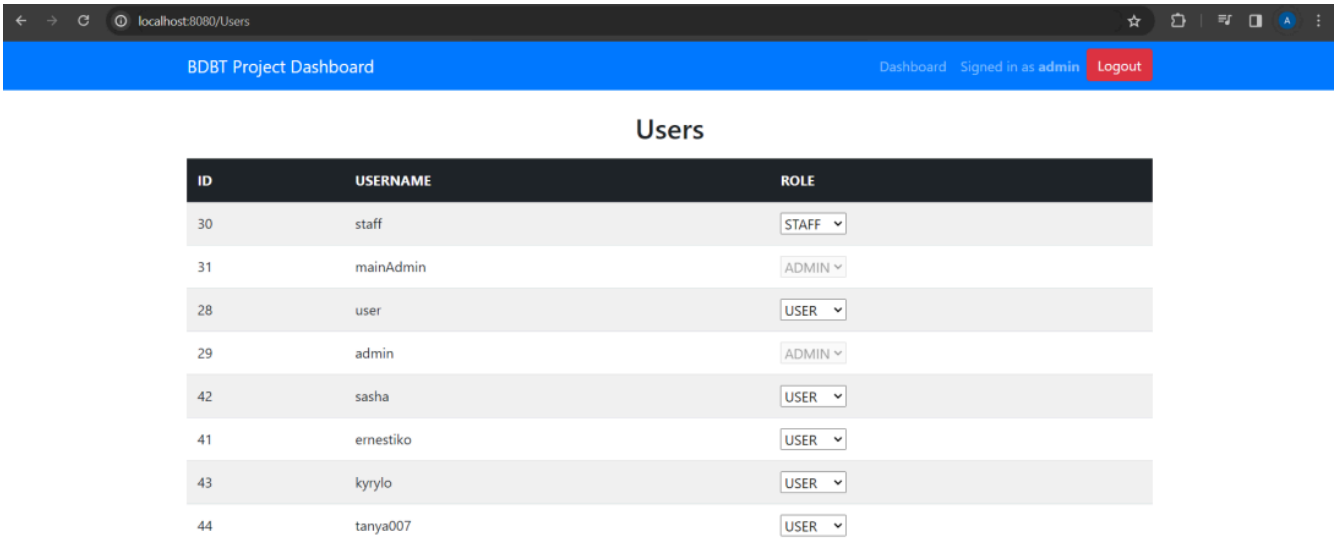


Figure 9: Table of registered customers

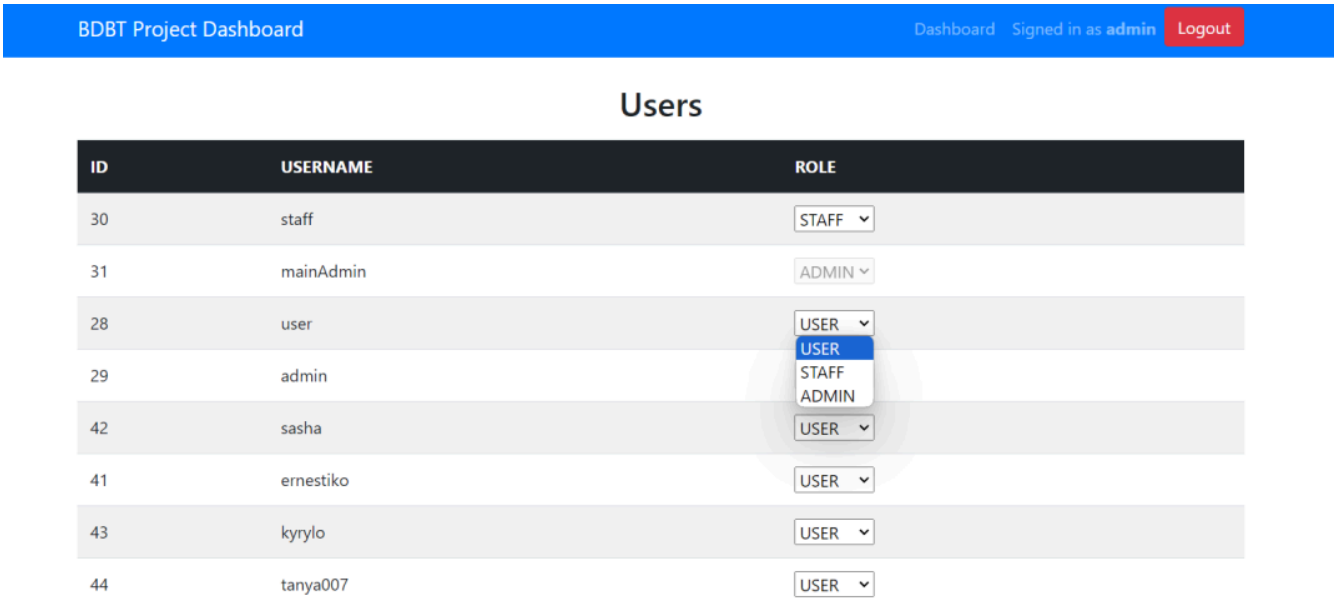


Figure 10: Managing users page

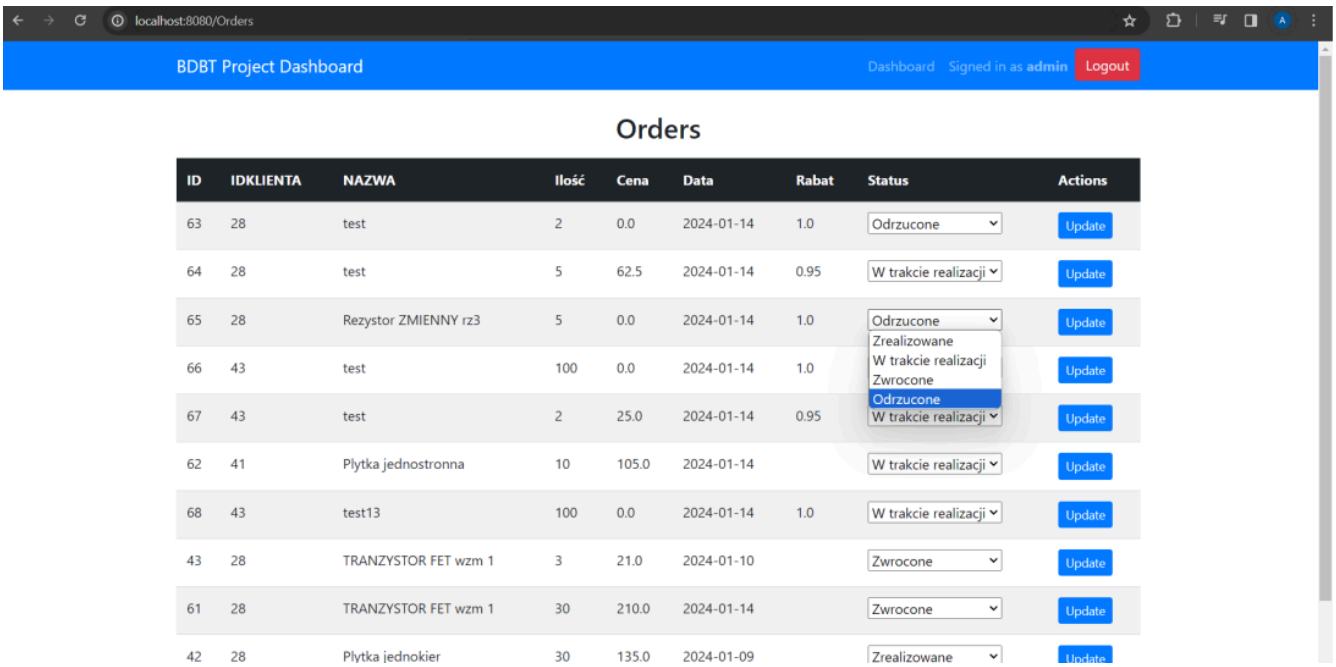


Figure 11: Orders managing page

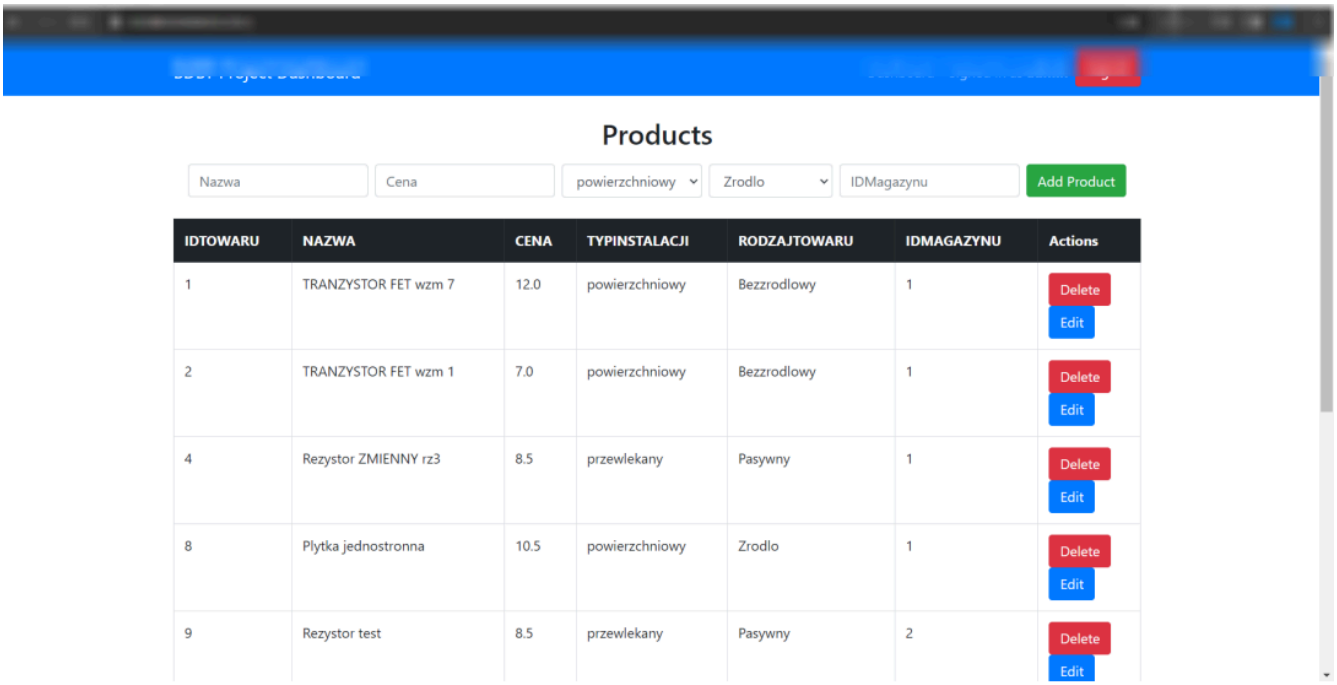


Figure 12: Table of products

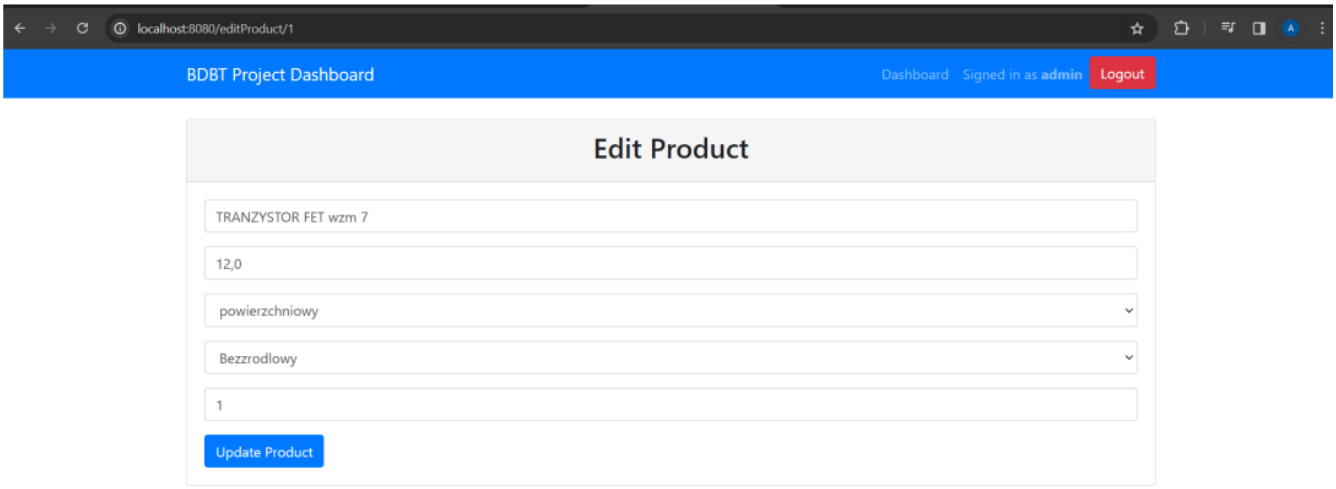


Figure 13: Changing products page

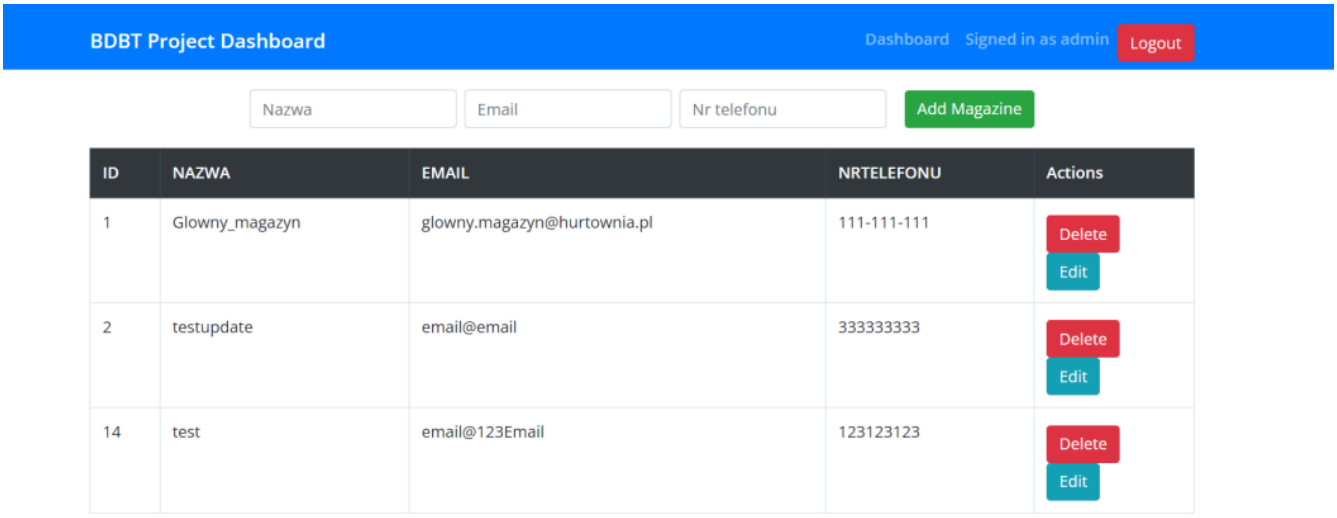


Figure 14: Table of users

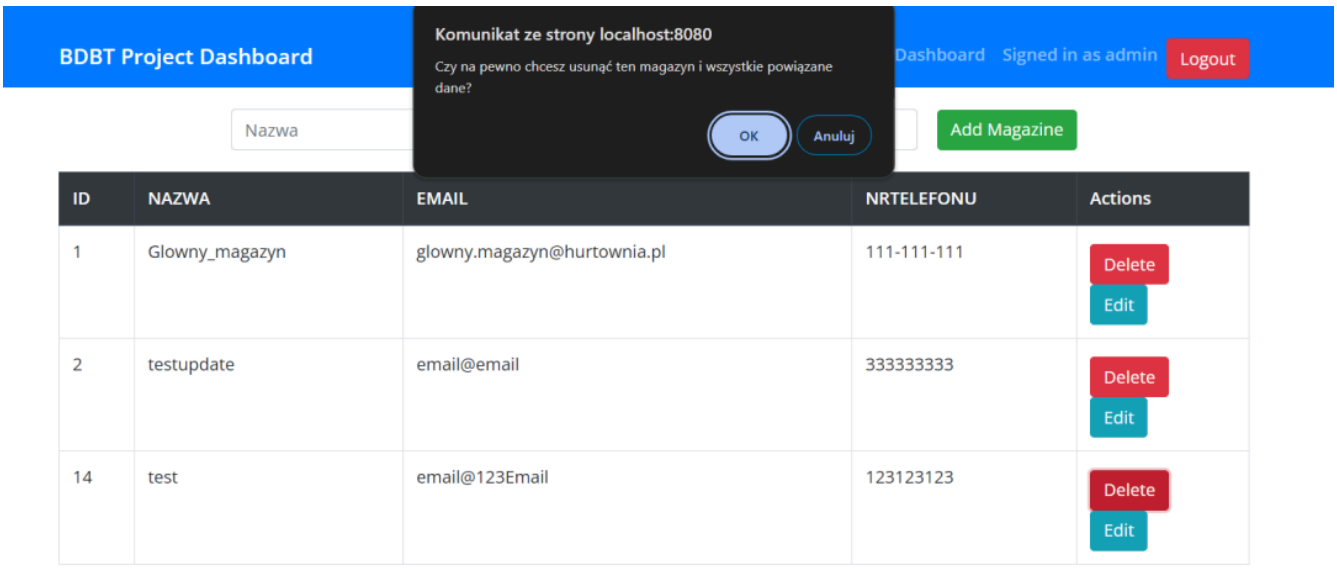


Figure 15: Removing the chosen user

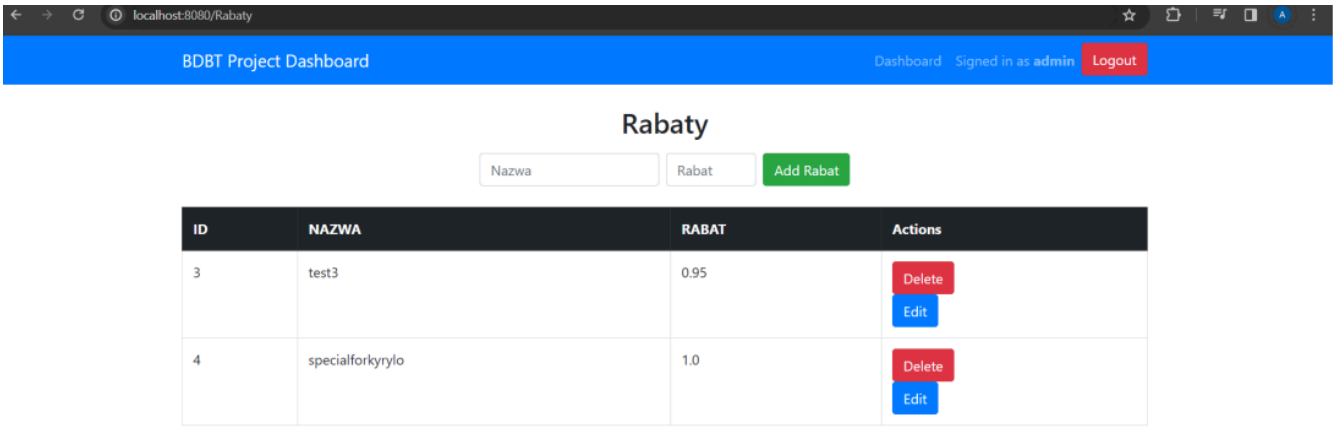


Figure 16: Table of discounts

Introduction **2**
1. Technology Used **2**
1.1 POM.XML 2
1.2 Models 3
1.3 DAO Files 3
1.4 Controllers 3
1.5 Security Configuration 4
1.6 Application Properties File 4
1.7 Templates 4
2. Application Presentation **5**