Data Bases and Big Data 22Z Part II

Project Documentation

Subject of the project: Web Application for database

Patryk Lipiński,

Jakub Sulikowski,

Warsaw University of Technology

24.01.2023

Project supervisor: mgr. inż. Tomasz Mrożek

Spis treści

1. Introduction

1.1. The purpose of the project

The aim of this project was to create a client application for the database created in the previous part of the course Databases and Big Data. The following planes had to be created:

— 1. Intuitiveness - proper placement of components on the page, transparency, etc.
— 2. Support for diffrent user perspectives - providing data presentation in relation to at least two user perspectives: user and administrator.
— 3. Supporting basic database operations - CRUD (adding data, deleting data, modifying data and viewing them depending on the perspective)

2. Technology used

In our project we used the following technologies:

— Spring Boot with Tomcat server
— Spring Security to secure access to applications
— Spring MVC as a pattern for managing application layers
— Spring JDBC to work with oracle databases
— Boostrap CSS v.4.0.0.2 used for website styling
— Thymeleaf for the view layer
— Oracle JDBC11 driver
— Maven - project management tool (dependency management, etc.)
— Lombok - Java library used to speed up code writing
— JUnit for creating unit tests
— Webjars - web libraries packed in JAR files intended for client applications
— Common Validator - checking data integrity

2.1. Spring Boot

Spring Boot is a framework that facilitates the creation of applications using the Spring framework. Spring Boot has a Tomcat server with which the application can be run on a web server.

2.2. Spring Security

Spring Security is a framework used to secure applications. It allows for:
— creating a login page,
— creating authentication data (login and password) using data from the database,
— blocking access to websites for users with insufficient permissions,
— blocking the use of DAO class methods for users who should not have access to them.

2.3. Spring MVC

Used as a pattern to create an application based on the MVC (Model-View-Controller) architecture. The purpose of such a pattern is to divide applications into 3 layers:
— model layer - responsible for storing information, that will be exchanged between the server and the client,
— controller layer - responsible for receiving requests and sending them to the model and updating views,
— layer - responsible for the presentation of data, it boils down to displaying an HTML file, most often supplemented with data from the model.

2.4. Spring JDBC

Spring JDBC is used to work with Oracle databases. It is an interface that simplifies querying the database. It is used to perform CRUD (create, read, update, delete) operations on tables.

2.5. Bootstrap CSS

Bootstrap CSS is a CSS library. Contains tools useful for creating a graphical user interface.

2.6. Thymeleaf

Thymeleaf is a template engine for building server-side UI elements. It allows usage of HTML files as a template, which are then used to create the displayed pages.

2.7. Oracle JDBC11

The Oracle JDBC11 driver is a set of classes that implement the interfaces of the java.sql package. A low-level programming interface that calls direct SQL commands.

2.8. JUnit

JUnit is a tool for creating unit tests. Unit tests are used to check whether a given functionality works as intended. JUnit itself automates the creation of unit tests and creates test reports.

2.9. Maven

Maven is a project management development tool. Maven dynamically fetches Java libraries and plugins from repositories. In addition, it contains information about the project, the Java version used and dependencies to external tools.

2.10. Lombok

Lombok is a Java library that allows you to reduce the amount of repetitive code by automatically generating code. Using the appropriate annotations, the appropriate constructions will be generated automatically during code compilation.

2.11. WebJars

WebJars are JAR files that package client-side libraries (e.g. CSS) to facilitate their use in a web application.

2.12. Commons Validator

Commons Validator is a tool for validating input data. It is used to validate data - which the user enters into forms - in terms of format or other characteristics.

3. Aplication usage

3.1. Main page

Fig.1 shows the start page, which consists of the top navigation bar and the main part of the page. On the left side of the navigation bar is the logo, while on the right side is the anchor tag directing to the login page. On the main part of the page there is a welcome message and another anchor leading to login.

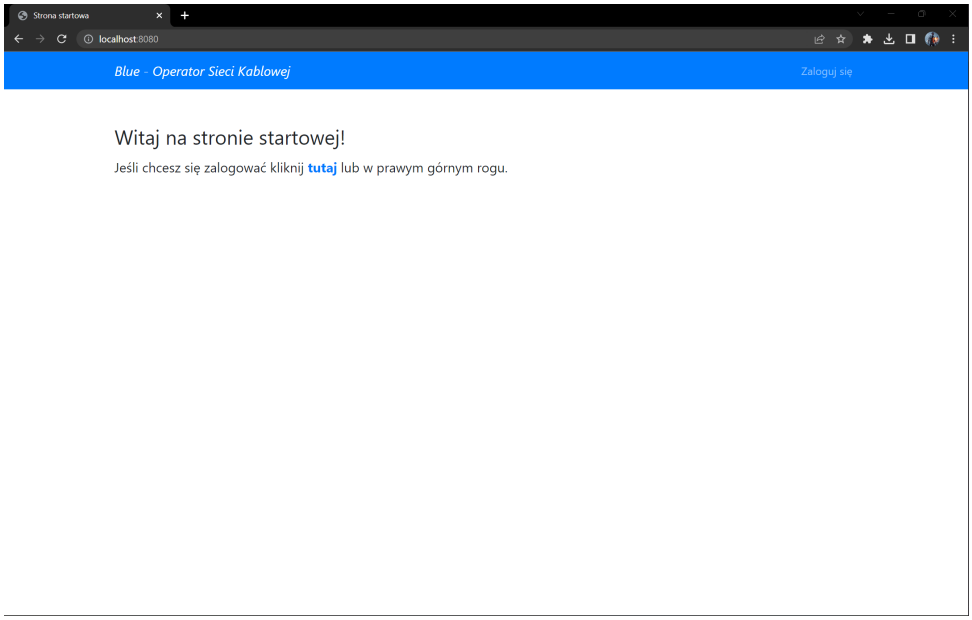

Fig. 1. Starting page

3.2. Login page

Fig.2 shows the login page. It consists of a logo, the inscription "Log in" and a login form completed with a button.

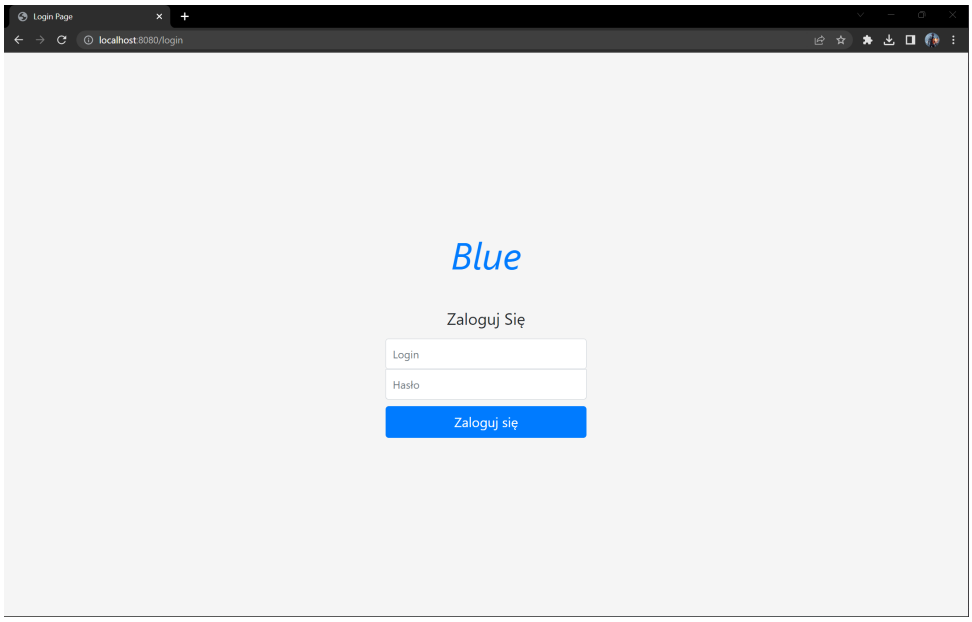

Fig. 2. Login page

When the website user tries to log in with incorrect data, the error shown in Fig. 3 will be displayed:

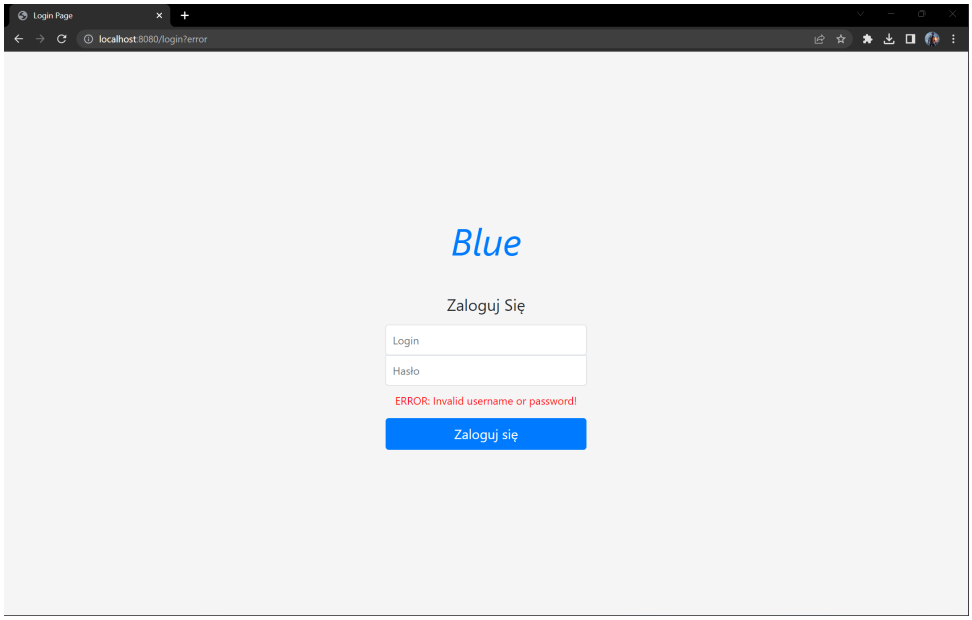

Fig. 3. Login page

3.3. Admin's main page

The main admin page, shown in Fig.4, consists of a top navigation bar, a side navigation bar and a panel where the main content is displayed. The upper navigation bar contains a shortened, exemplary name of the cable network operator, which is also a link to the home page, a button with a scalable vector graphic of a house, which, when pressed, leads to the user's home page, an inscription saying who is logged in and a logout button.

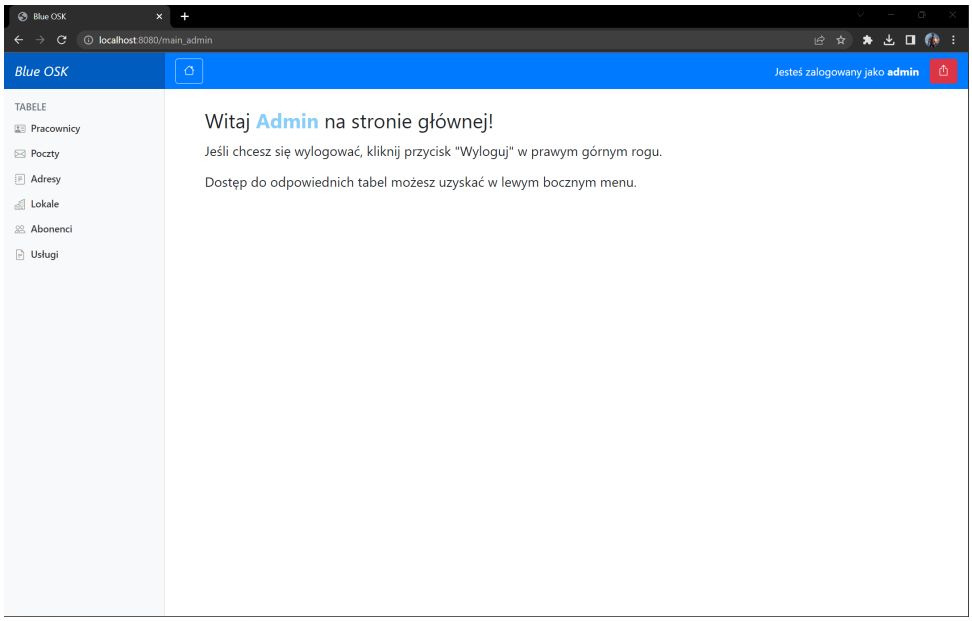

Fig. 4. Admin's main page

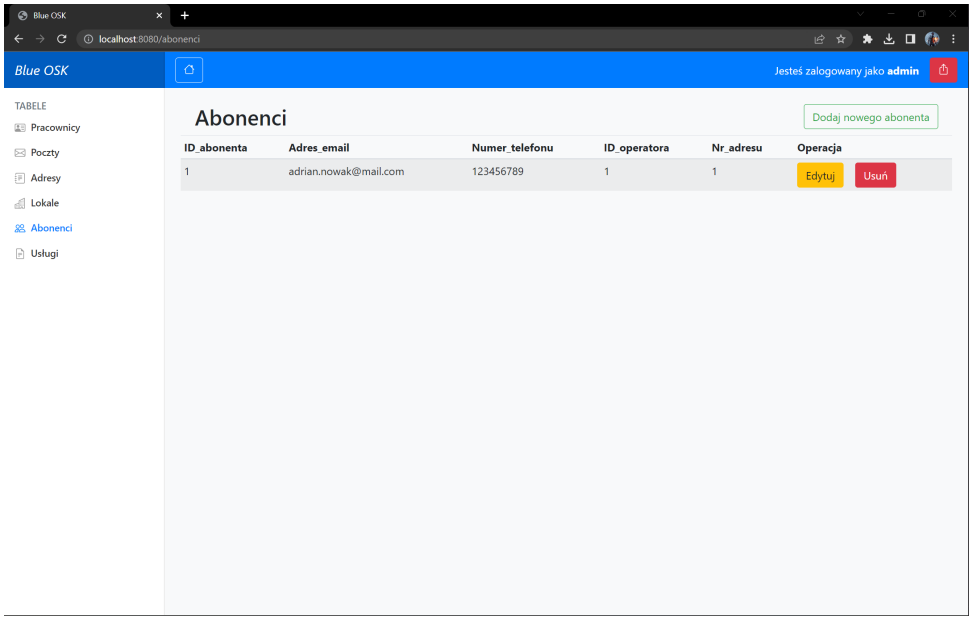

Fig. 5. One of the tables selected from the sidebar

Fig.5 shows the table of subscribers in the main panel of the administrator's page. To the right of the panel header there is a button to add a new entry to the table. In the "Operations" column you can see two buttons: "Edit" used to modify the table entry and "Delete" which deletes the entry when pressed.

3.3.1. Adding an entry to the table

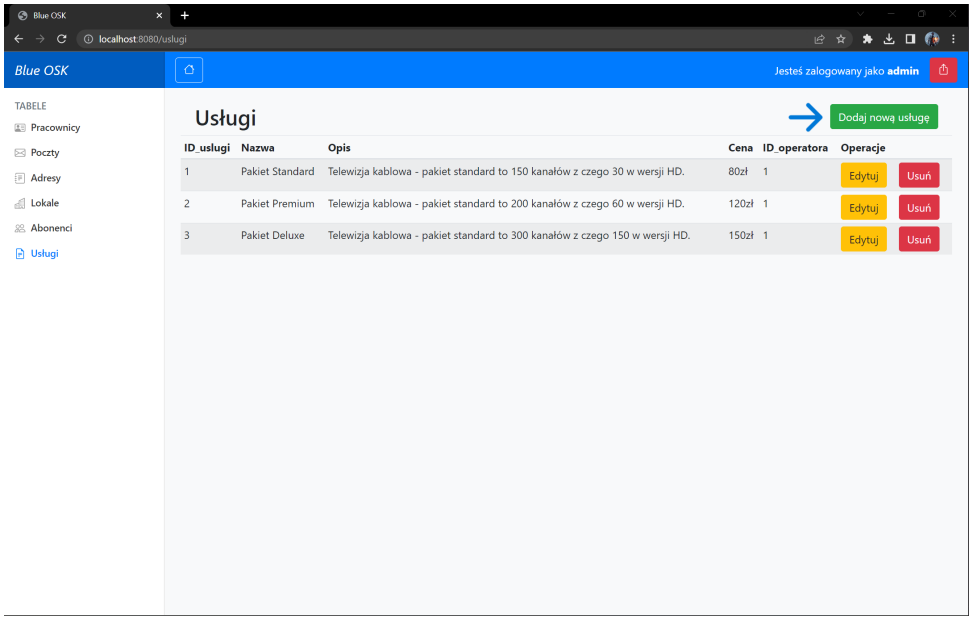

Fig. 6. Pressing the table add button

The panel for adding a new service, shown in Fig.7, consists of a form in which we enter the appropriate data and confirm it with the button.

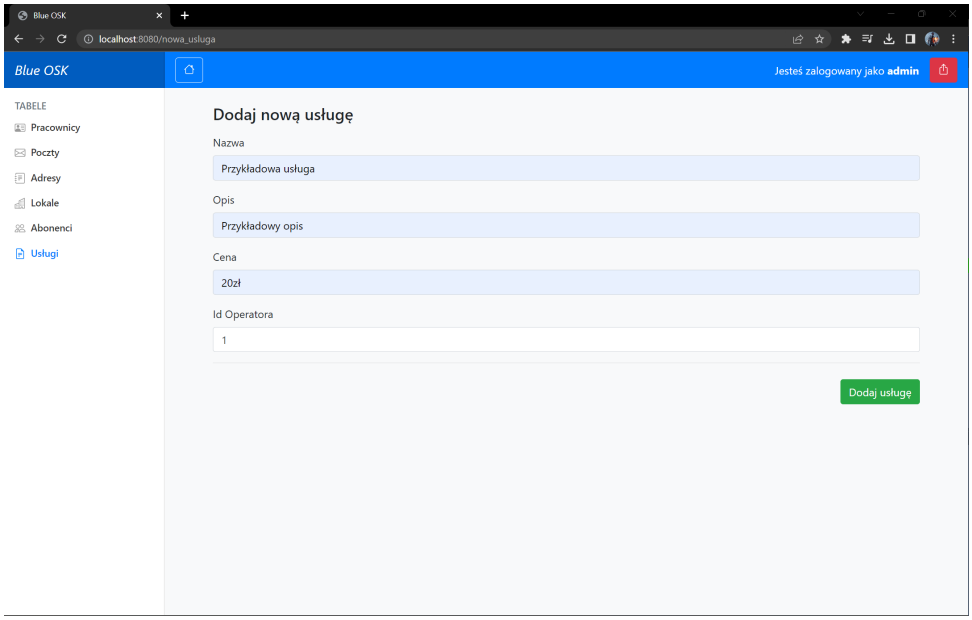

Fig. 7. Filling out the form with data for a new entry in the table

Fig.8 shows that an example entry has been added to the table.

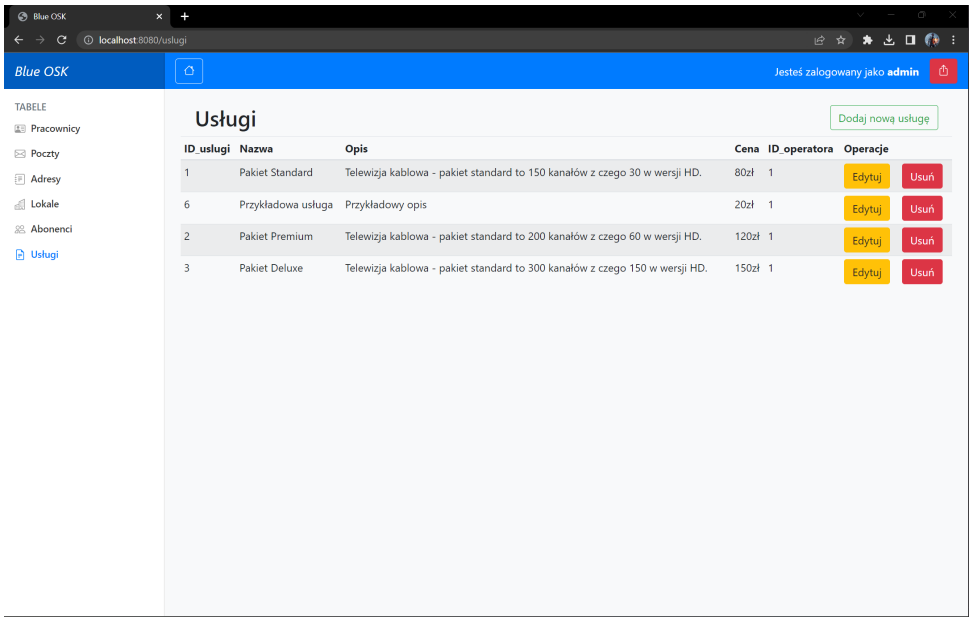

Fig. 8. The effect of the operation

3.3.2. Delete an entry from the table

An entry can be deleted after pressing the ”Delete” button to the right of the table entry shown in Fig.9.

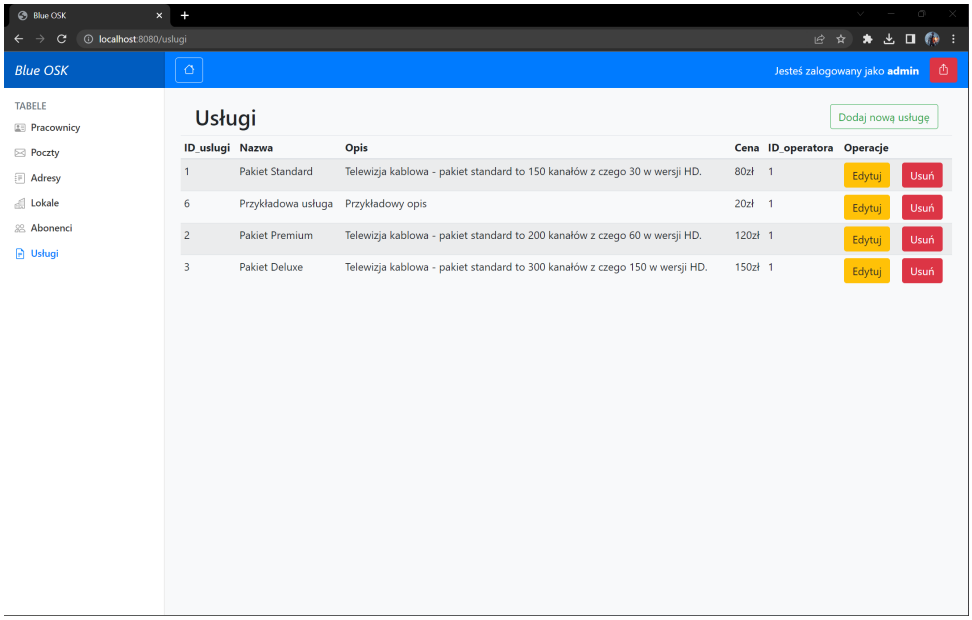

Fig. 9. Before deleting the entry

Fig.10 shows the effect of the operation.

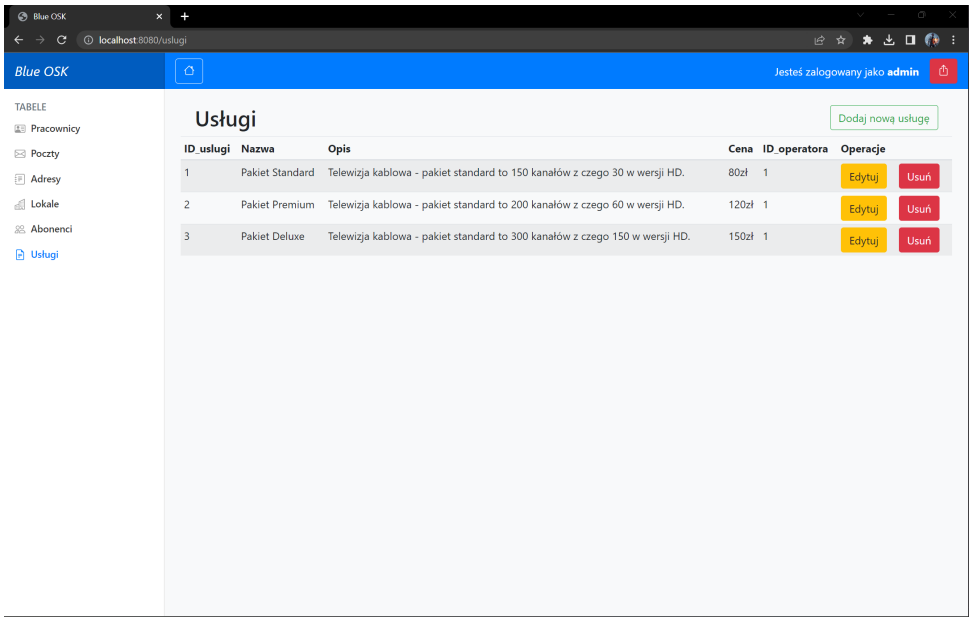

Fig. 10. After deleting the entry

3.3.3. Editing table

The table can be edited using the "Edit" button shown in Fig.11.

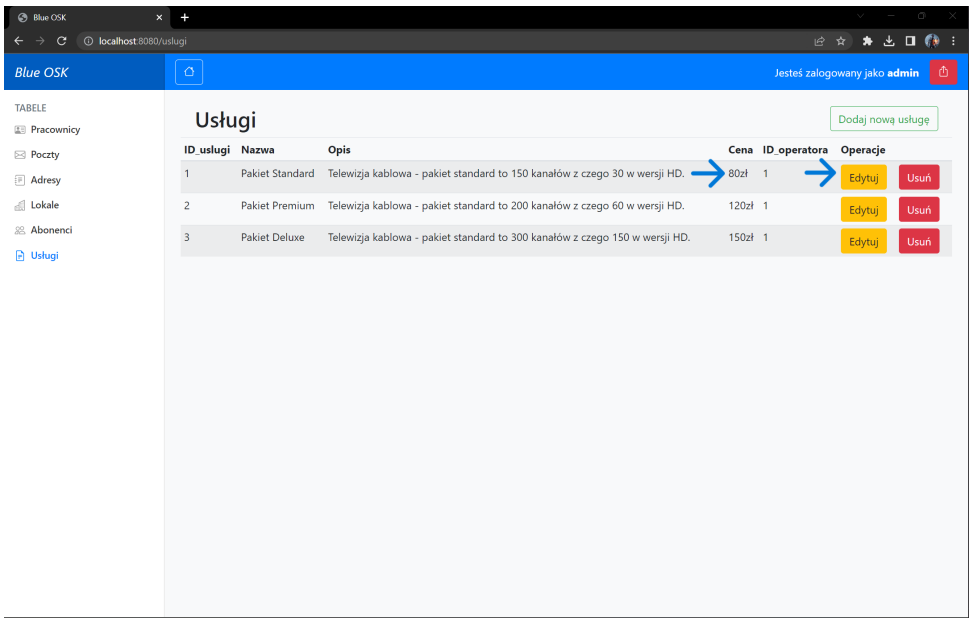

Fig. 11. Clicking the table entry edit button

Fig.12 shows the entry editing page. The form is filled with data that we can change. Confirm the changes with the "Save" button. In this example, we're changing the value of the "Price" field.

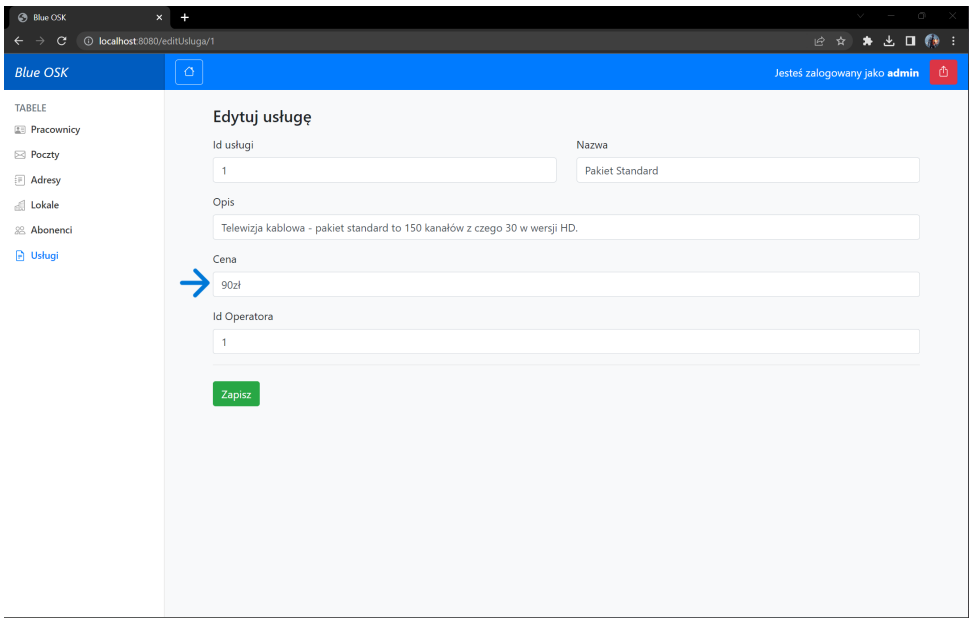

Fig. 12. Changing the price of one of the services

Fig.13 shows the effect of the operation.

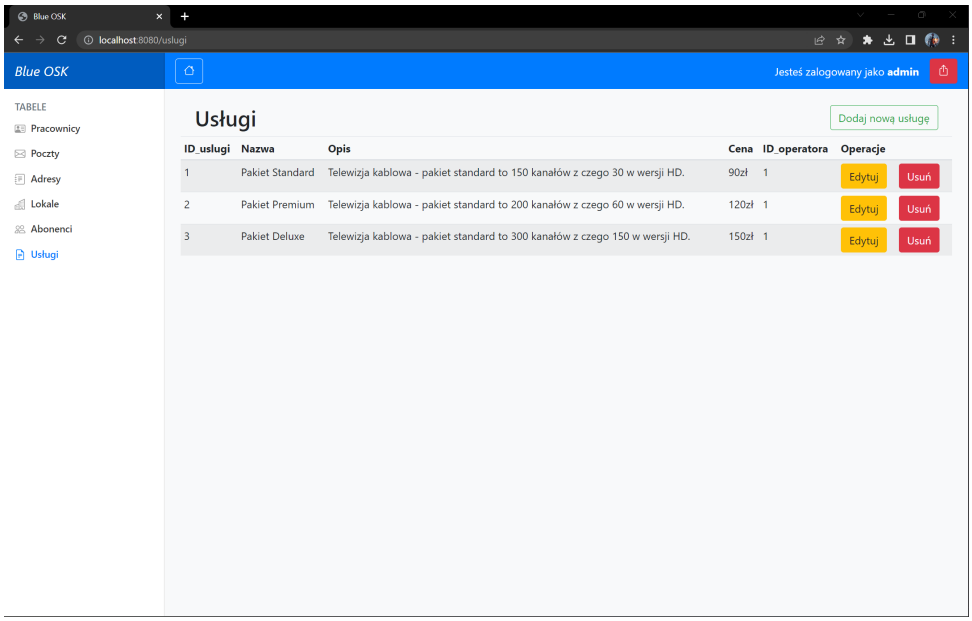

Fig. 13. Table view after changing the price

3.4. User's main page

Basic user login successfully implemented. If the subscriber is in the database, he can log in by entering his e-mail address in the login field and his user ID as a password. Unfortunately, we ran out of time to implement email and password login. Below is an example of logging in to the subscriber's account with the email address adrian.nowak@mail.com.

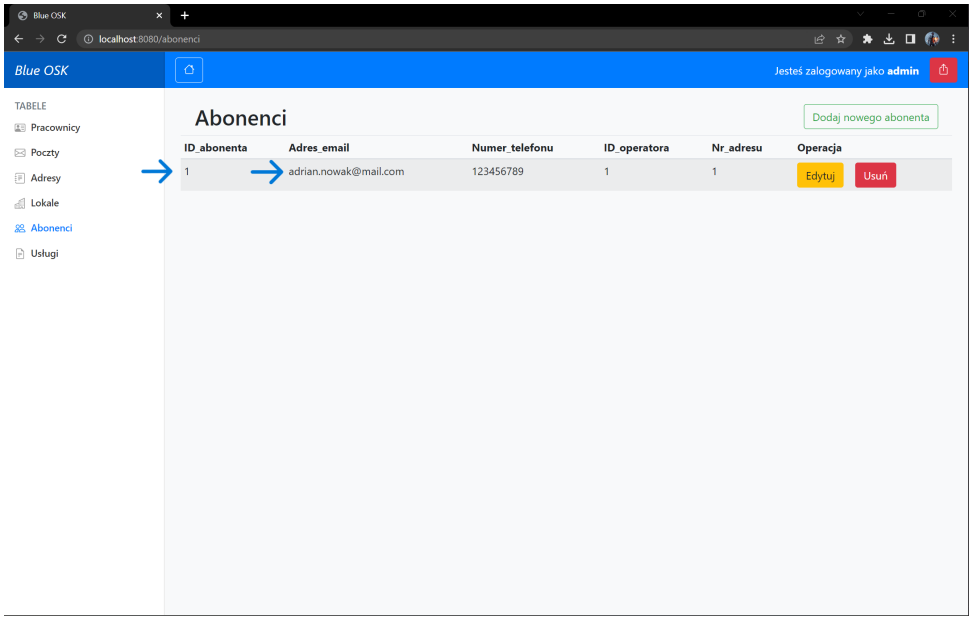

Fig. 14. Subscriber Adrian Nowak has identifier 1

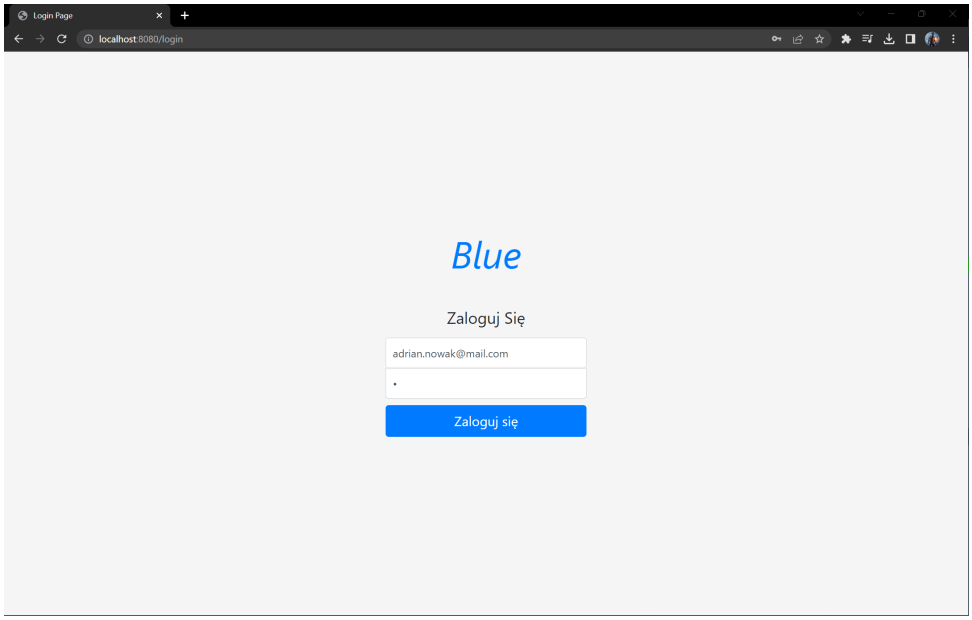

Fig. 15. Login to the subscriber's account by providing e-mail and ID

The logged-in subscriber's main page is similar to the administrator's main page. It differs in the number of options displayed in the side navigation bar and the username displayed.

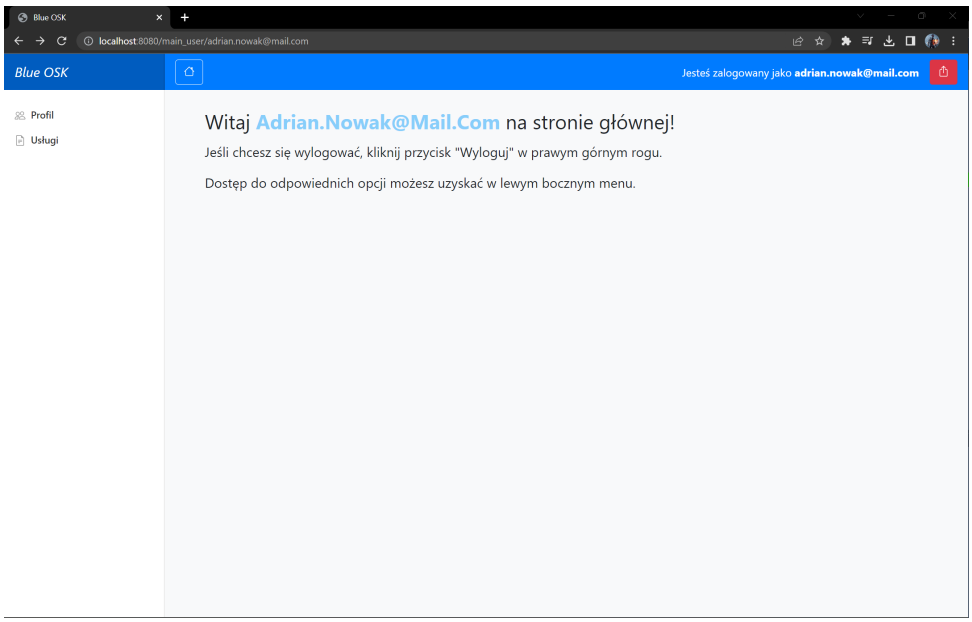

Fig. 16. Subscriber's main page after logging in

3.4.1. Editing profile details

Fig. 17 shows the subscriber data modification panel. The "Subscriber ID" field is read-only, as modification of this value would lead to errors in the database.

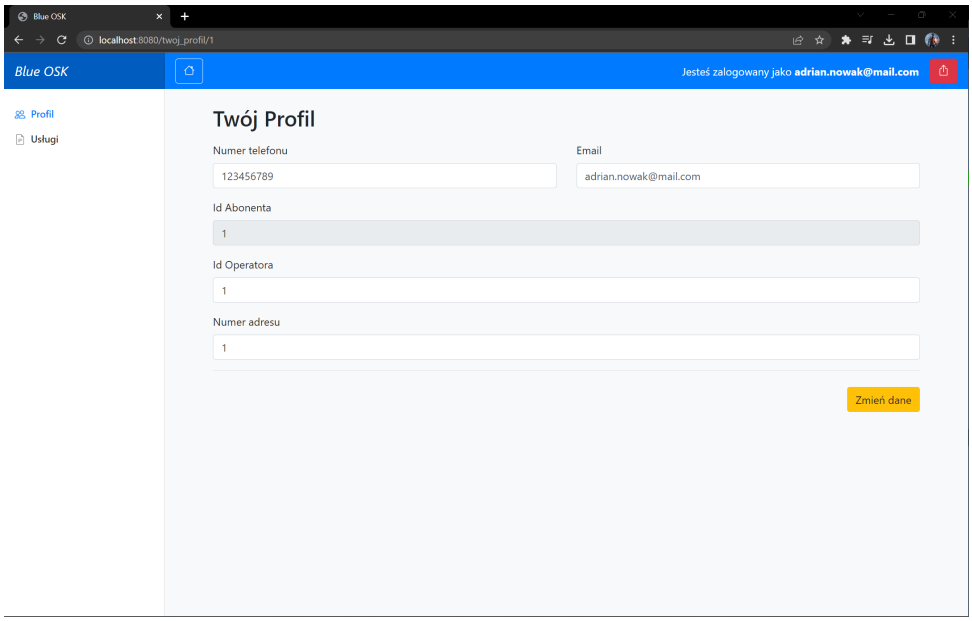

Fig. 17. A panel of services available to the user

3.4.2. Browsing available services

The second option in the user account is to browse and order the service. The latter option has not been implemented.

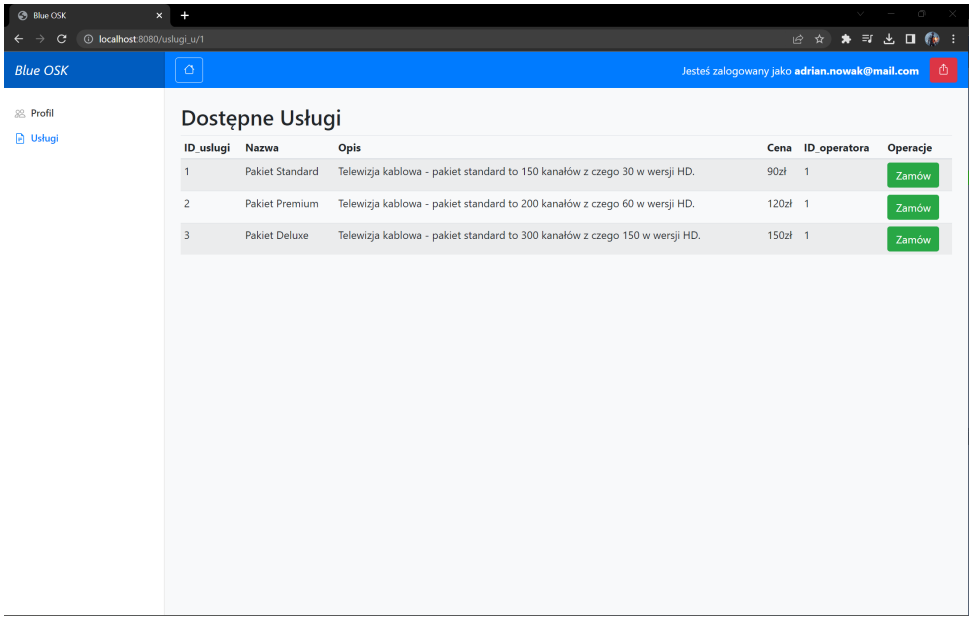

Fig. 18. A panel of services available to the user

3.5. Error handling

Our project has handled some errors while using the site. The errors handled are:
— 403 - Forbidden Page
— 404 - Page Not Found
— 500 - Internal Server Error
— 504 - Gateway Timeout
The rest of the errors have also been handled on the error page. Below are examples of error handling. Unfortunately, the 504 error could not be triggered.

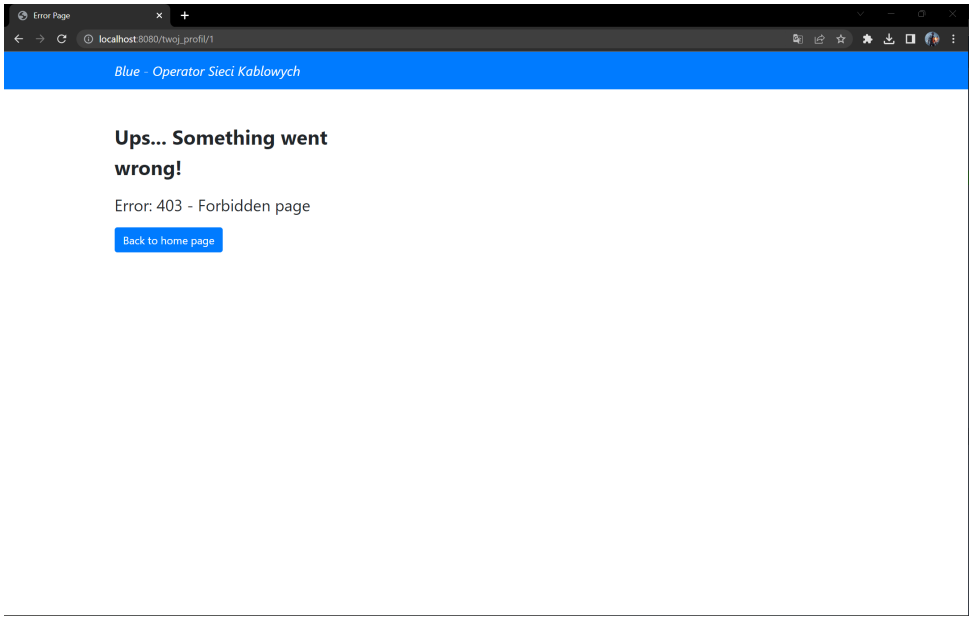

Fig. 19. 403 Forbidden page error handling

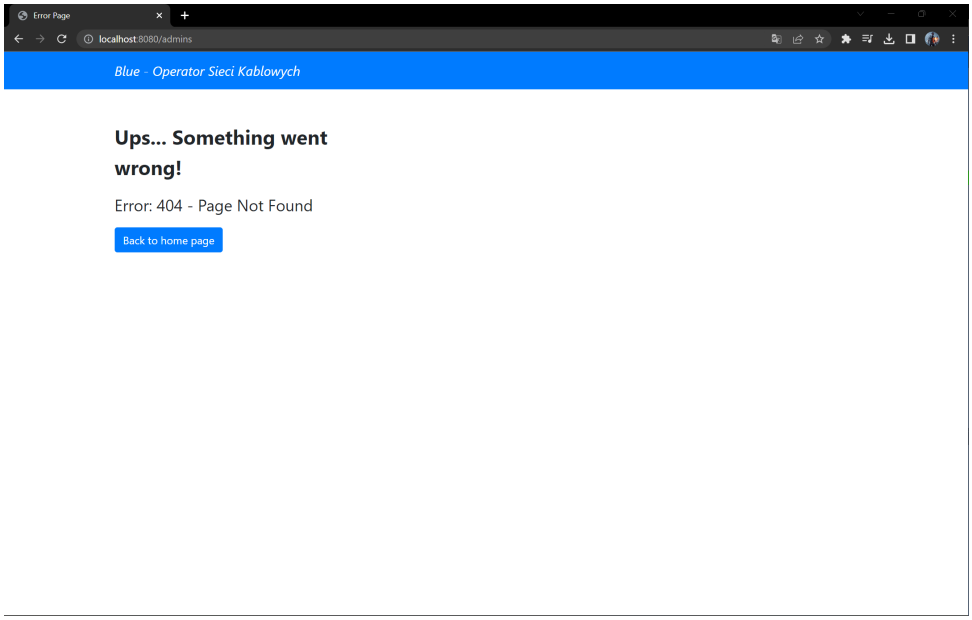

Fig. 20. 404 page not found error handling

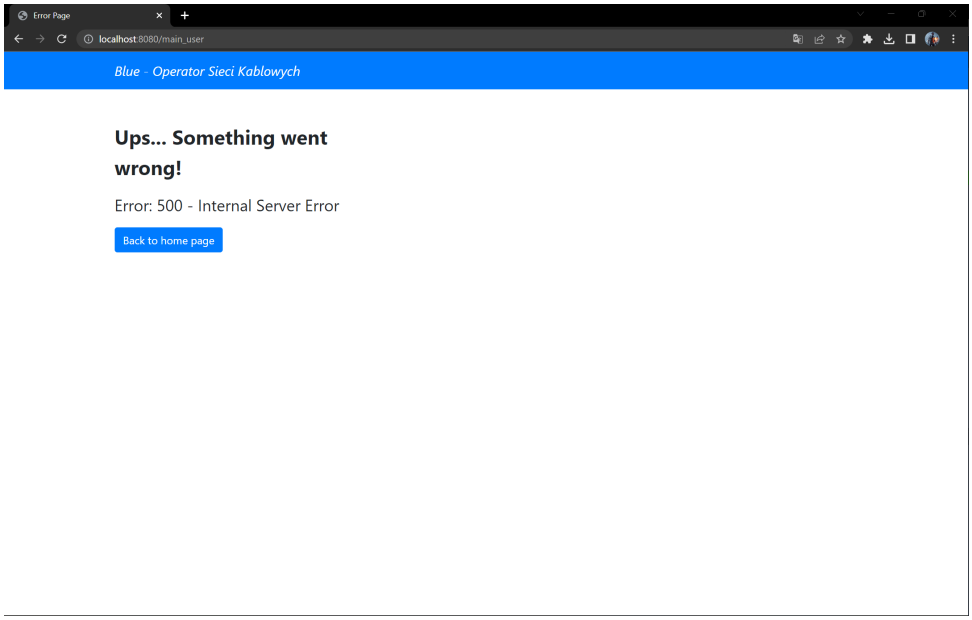

Fig. 21. 500 Internal server error handling

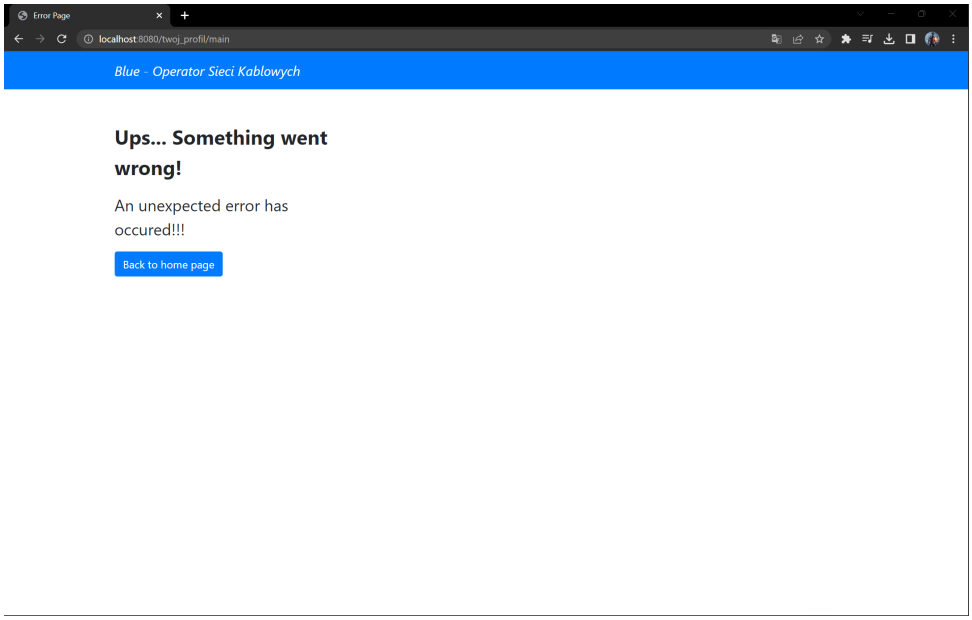

Fig. 22. Handling other errors

4. Podsumowanie

We managed to implement the basic design assumptions and implement additional elements, such as logging in using subscriber data from the database.