POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH
TELEKOMUNIKACJA

BAZY DANYCH I BIG DATA

Hurtownia części elektronicznych

Autorzy:
Anh Quan Do 325 268
Agata Zatorska 325 338

Prowadzący: dr inż. Tomasz Mrozek

Spis treści

1 Zakres i cel projektu **2**

2 Wykorzystane tabele **2**

3 Perspektywy użytkowników **2**

4 Implementacja **2**
4.1 Logowanie 2
4.2 Strona główna 3
4.3 Perspektywa administratora 4
4.4 Perspektywa klienta 6
4.5 Zarządzanie błędami 6
4.6 Komunikaty 7

5 Wygląd aplikacji **8**

1 Zakres i cel projektu

Celem projektu było zaprojektowanie oraz zaimplementowanie aplikacji webowej dla stworzonej w poprzedniej części projektu bazy danych. Przypisanym nam tematem była hurtownia części elektronicznych. Aplikację wykonaliśmy z wykorzystaniem framework'a Spring w języku java. Aplikacja miała być intuicyjna i przejrzysta, obsługiwać różne perspektywy użytkowników oraz wspierać podstawowe operacje w ramach komunikacji z bazą danych.

Do realizacji projektu wykorzystaliśmy następujące narzędzia.

- Intellij IDEA
- Maven
- Oracle Database 19c
- sql developer

2 Wykorzystane tabele

W aplikacji uwzględniliśmy cztery tabele z naszej bazy danych: hurtownie, adresy, klienci oraz produkty.

3 Perspektywy użytkowników

W naszej aplikacji obsługujemy dwie perspektywy użytkowników: wąską - klienta oraz szeroką - administratora. Administrator ma prawo do wykonywania wszystkich, oferowanych przez naszą aplikację, operacji na bazie danych: przeglądania, dodawania, modyfikowania oraz usuwania danych z tabel. Klient może obejrzeć informacje z tabeli produkty, w ramach przeglądania dostępnych ofert. Może też zebrać informacje o naszych hurtowniach, w tym ich adresach, oraz zobaczyć i edytować swoje dane.

4 Implementacja

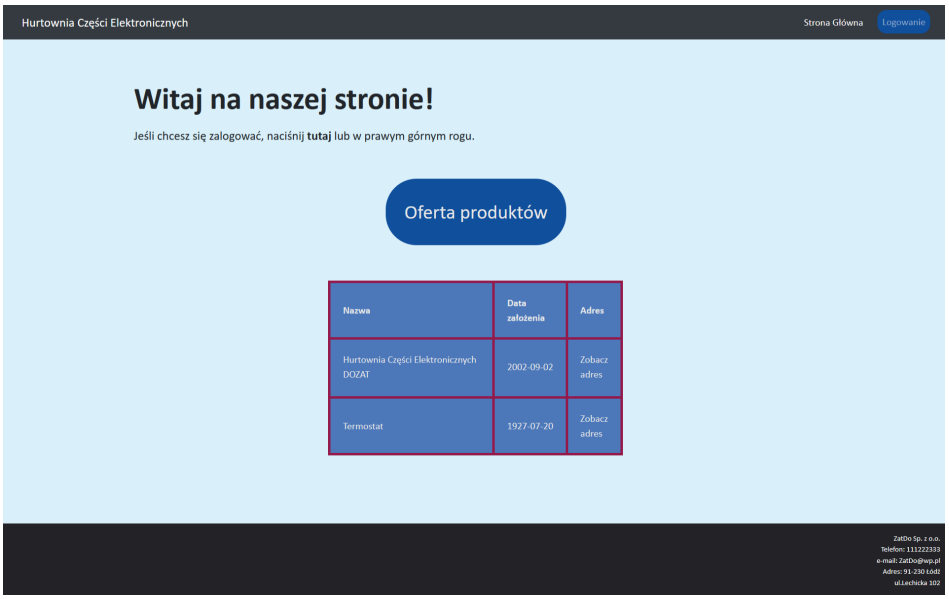
Tak jak wspomnieliśmy wcześniej aplikację tworzyliśmy na framework'u Spring z wykorzystaniem narzędzia Maven, który wspierał automatyzację dzięki plikowi POM.XML, w którym znajdują się wszelkie informacje o konfiguracji projektu.

4.1 Logowanie

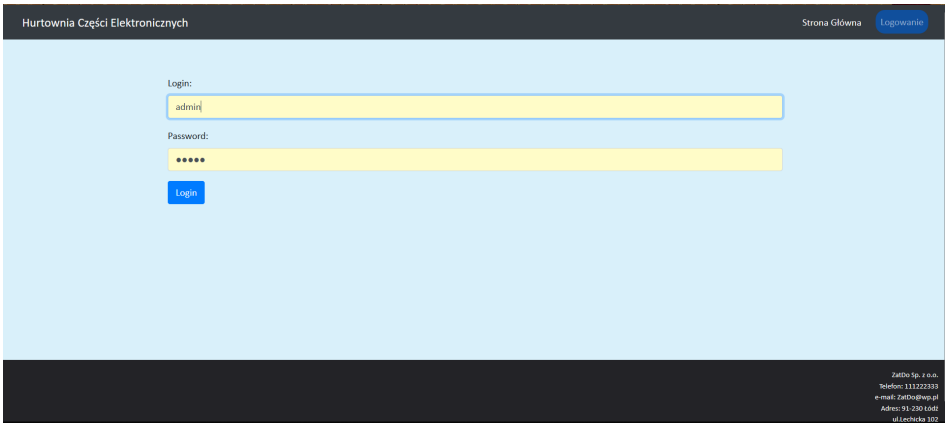
Prace implementacyjne zaczęliśmy od stworzenia systemu logowania. W tym celu wykorzystaliśmy moduł Spring Security odpowiedzialny za uwierzytelnienie i autoryzacje. Upewniliśmy się, że w pliku POM znajdują się zależności dodające funkcje bezpieczeństwa. W klasie SecurityConfiguration skonfigurowaliśmy 5 użytkowników: czterech klientów i jednego administratora oraz zabezpieczyliśmy wybrane endpointy za pomocą wyrażeń antMatchers().hasRole() w taki sposób, że tylko użytkownicy z wybranymi rolami mogą mieć wgląd do odpowiednich stron. Na przykład klient może zobaczyć tylko swoje dane i formularz ich edycji, a wgląd do danych wszystkich klientów ma wyłącznie administrator.

Jak widać na rysunku 1 na stronie głównej aplikacji dodaliśmy przycisk przekierowujący użytkownika na stronę logowania, która jest zaprezentowana na rysunku 2. Po udanym logowaniu użytkownik zostaje przekierowany na stronę główną administratora lub użytkownika. Ta decyzja jest podejmowana za pomocą funkcji zapisanej w kontrolerze aplikacji, która przyjmuje jako parametr żądanie HTTP, które zostało wysłane do serwera.

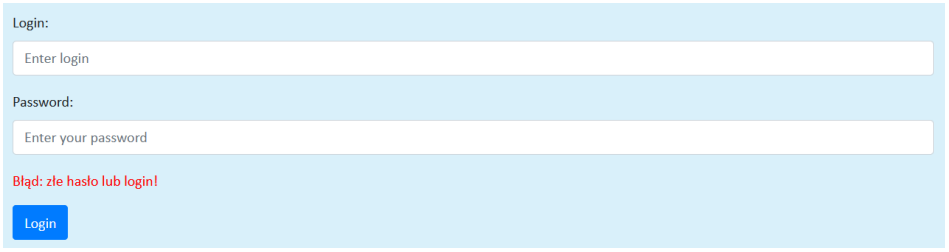
Jeżeli użytkownik źle wpisze hasło lub login, zostaje o tym poinformowany komunikatem (Rysunek 3.).



Rysunek 1: Strona główna



Rysunek 2: Logowanie



Rysunek 3: Komunikat o błędnym logowaniu

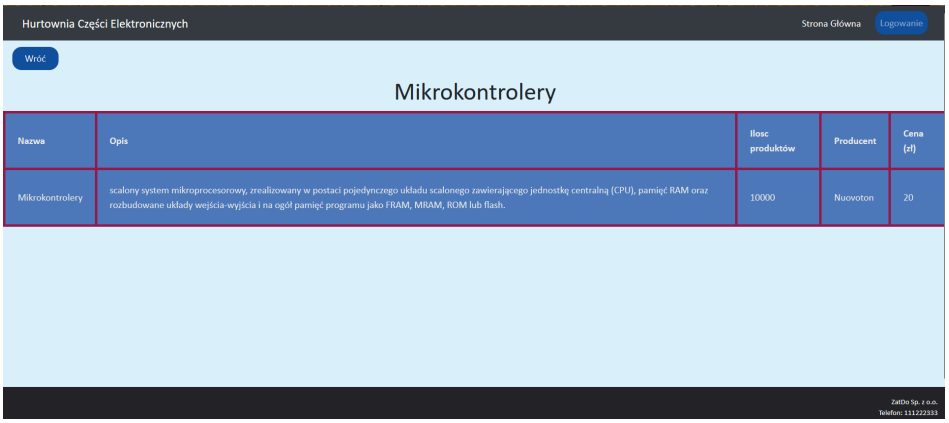
4.2 Strona główna

Poza linkiem do logowania na stronie głównej znajduje się tabela z naszymi hurtowniami, wraz z opcją zobaczenia ich adresów po wybraniu odpowiedniego przycisku. Jest tam również link, który kieruje użytkownika na stronę z ofertą naszych produktów, która jest pokazana na rysunku 4. Informacje pokazywane w tabeli aplikacja pobiera z bazy danych Oracle. Aby umożliwić wyświetlanie ich na stronie dla każdej z tabel stworzyliśmy oddzielną klasę DAO w Javie. Znajdują się tam funkcje, które mają za zadanie tworzyć komendy w języku SQL z np. prośbą o podanie wszystkich rekordów z tabeli na podstawie klucza

głównego, zaktualizowanie widniejących tam informacji, lub usunięcie rekordu z wybranym kluczem głównym.

Funkcje z klas DAO są następnie w ramach potrzeby egzekwowane w kontrolerze aplikacji i zebrane z bazy danych informacje są przekazywane do pliku HTML strony głównej lub strony oferty produktów, który jest odpowiedzialny za ich wyświetlanie. Na przykład metoda showProducts pobiera informacje o produkcie o określonym kluczu głównym, dodaje ją do modelu, a następnie przekierowuje do widoku informacjeOProdukcie, w którym wybrane informacje są wyświetlane na stronie w formie tabeli, co można zobaczyć na rysunku 5.

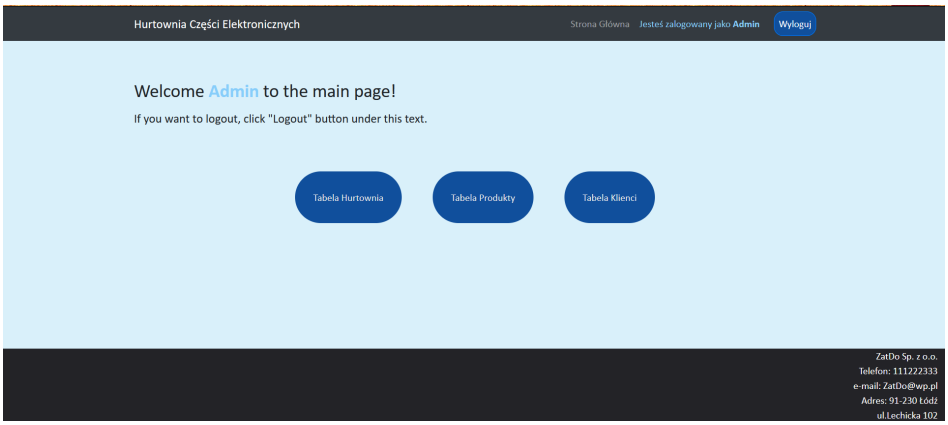
Rysunek 4: Oferta produktów



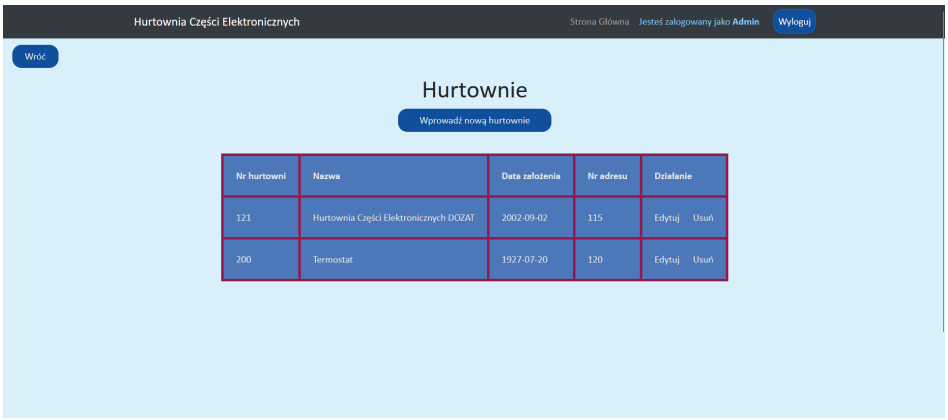
Rysunek 5: Więcej o produkcie

4.3 Perspektywa administratora

Jak widać na rysunku 6 administrator ma na swojej stronie do wyboru trzy przyciski, które kierują go do widoku tabel hurtowni, produktów i klientów. Przykład jednej z tabel znajduje się na rysunku 7. Z poziomu każdej z nich może wybrać opcję dodawania, edytowania lub usuwania wybranych rekordów. W tym celu stworzyliśmy dwa dodatkowe pliki HTML, dla każdej z tabel, po jednym formularzu do modyfikowania danych (Rysunek 8.) i po jednym formularzu do tworzenia nowego wpisu w bazie (Rysunek 9). Wszystkie te czynności są możliwe dzięki funkcjom zapisanym w kontrolerze aplikacji. Funkcje save i update obsługują żądania HTTP typu POST i przesyłają w formularzu HTML dane mapowane np. na obiekt Hurtownia lub Klient. Natomiast funkcja delete usuwa z tabeli rekord dotyczący obiektu o danym kluczu głównym.

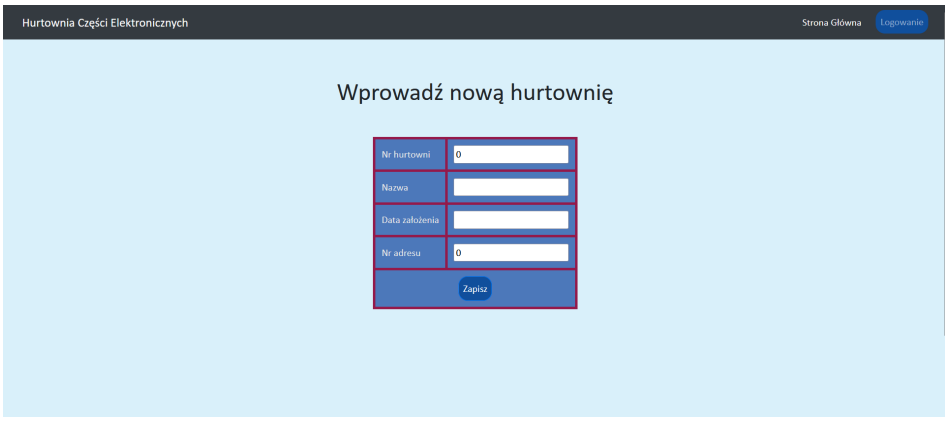


Rysunek 6: Strona main admin



Rysunek 7: Tabela z hurtowniami

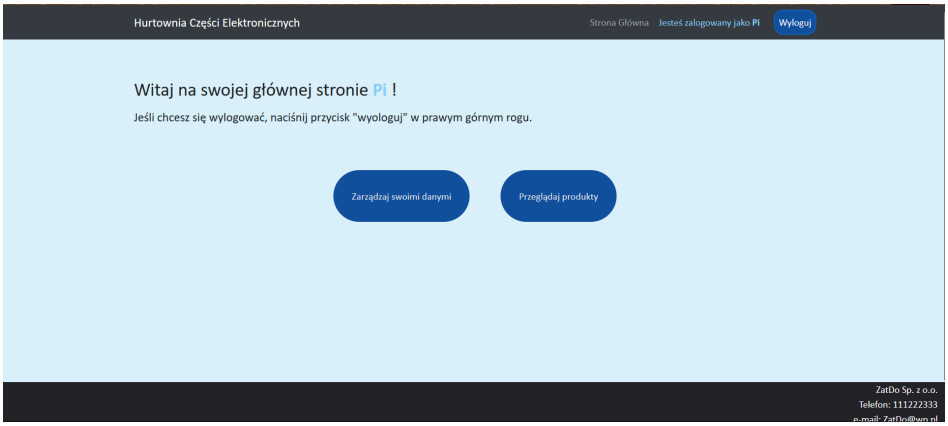
Rysunek 8: Formularz edycji hurtowni



Rysunek 9: Formularz tworzenia hurtowni

4.4 Perspektywa klienta

Klient po zalogowaniu ma do wyboru obejrzeć i ewentualnie edytować swoje dane lub zobaczyć ofertę produktów (Rysunek 10.). Ta pierwsza opcja ma podobny schemat działania co w przypadku administratora i częściowo wykorzystuje te same funkcje. Po wybraniu opcji edycji danych klient zostaje przenoszony na ten sam formularz co administrator.

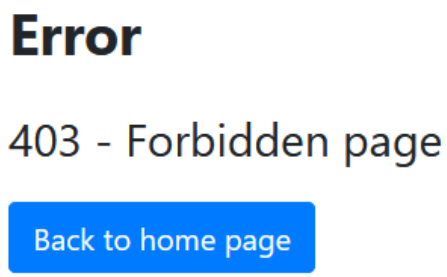


Rysunek 10: Strona main user

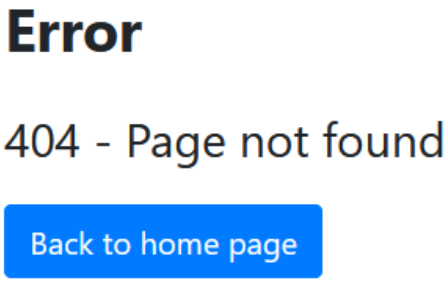
4.5 Zarządzanie błędami

Naszą aplikację zabezpieczyliśmy przed różnego rodzaju błędami.

Na samym początku rozpatrzyliśmy błędy 403 - forbidden page, 404 - page not found, 500 - internal server error, 504 - gateway timeout, jeżeli aplikacja napotkała inny błąd to był traktowany jako "unexpected error". W tym celu stworzyliśmy kontroler do obsługi błędów. Znajduje się w nim metoda, która odczytuje kod statusu błędu HTTP z żądania i przekierowuje użytkownika do odpowiadającego mu widoku. Dla każdego z błędów stworzyliśmy oddzielny plik HTML, który informuje użytkownika jaki błąd wystąpił. Przykłady widać na rysunkach 11 i 12.



Rysunek 11: 403 - Forbidden page

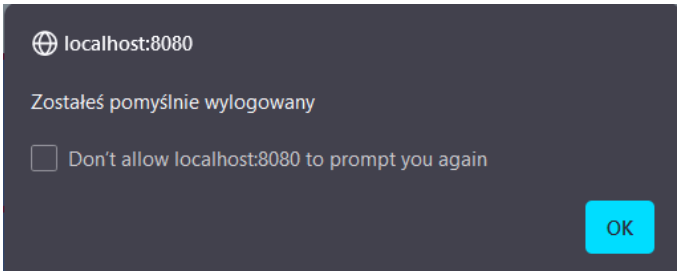


Rysunek 12: 404 - Strona main user

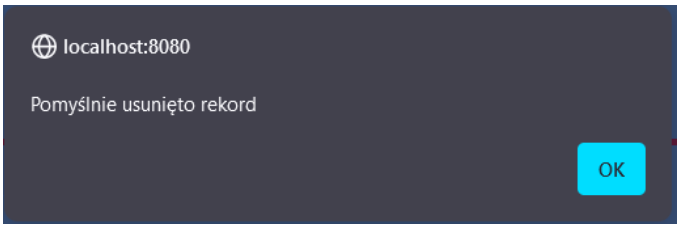
Aplikacja jest również zabezpieczona przed błędnym uzupełnieniem, przez użytkownika, formularza edycji. Jeżeli nie wypełni w nim wszystkich pól lub na przykład poda nieistniejący klucz nadrzędny, to aplikacja przekieruje go z powrotem na widok tabeli, którą chciał zmodyfikować. Osiągneliśmy to dodając do funkcji update i save konstrukcję catch exeption.

4.6 Komunikaty

Za pomocą javascript zaimplementowaliśmy pop-up'y wyświetlające informację dla użytkownika, że został poprawnie wylogowany (Rysunek 13.) lub, że rekord tabeli z bazy danych został poprawnie usunięty (Rysunek 14.).



Rysunek 13: Pop-up poprawne wylogowanie



Rysunek 14: Pop-up usunięty rekord

5 Wygląd aplikacji

Na koniec zajęliśmy się dostosowaniem wyglądu naszej strony internetowej. Obraliśmy sobie jako główny motyw kolor "baby-blue" - rgb(217, 240, 250). Pobocznym kolorem (secondary color) był u nas dark-blue o wartości rgb(16, 79, 156). Aby dokomplementować powyższy kolory użyliśmy koloru red-violet - rgb(135, 11, 67) - do wypełnienia brzegów tabel oraz do podświetlenia linków, gdy są one najeżdżane. Ostatnim kolorem, służącym jako główny kolor liter, był odcień białego - rgb(232, 232, 232).

Wszystkie modyfikacje graficzne wykonaliśmy jako **internal css**. Jako model układu użyliśmy layout'u flexbox, ponieważ jest najlatwiejszy i najwygodniejszy w użyciu.

Zgodnie z instrukcjami prowadzącego, użyliśmy szablonu stron interentowych **thymeleaf**. Pomógł on nie tylko w wyświetlaniu danych pobranych z bazy danych i przetransportowanych przez Spring'a, ale i był przydatny przy takich rzeczach jak nagłówek (navbar).