OPIS I STRUKTURA PROJEKTU

Magdalena Jaśkiewicz 233968

Marcelina Matuszewska 233986

**Zaawansowane programowanie aplikacji bazodanowych**

Projekt

Wypożyczalnia samochodów

Celem stworzenia aplikacji było zastosowanie zdobytej przez nas wiedzy podczas zajęć laboratoryjnych. Zadaniem projektu było wykonanie aplikacji dla firmy zajmującej się wypożyczaniem samochodów

**Język programowania**: Java

**Serwer baz danych**: Microsoft SQL Server Managment

**Środowisko programistyczne**: IntelliJ IDEA

**Dodatkowe wykorzystane technologie:**

* Postman
* Maven
* Spring Boot
* Spring Security
* Html
* CSS
* Thymeleaf

Struktura projektu aplikacji – backend została przedstawiona poniżej (rys. 1):

|  |
| --- |
|  |
| Rysunek 1. Struktura projektu aplikacji - backend. |

Konfiguracja właściwości aplikacji zostały ustawione w pliku *application.properties*, który jest plikiem tekstowym korzystającym z wbudowanych właściwości (rys. 2):

|  |
| --- |
|  |
| Rysunek 2. Plik application.properties. |

Łączenie z serwerem Microsoft SQL Server Managment było wykonywane w następujący sposób (rys. 3):

|  |
| --- |
|  |
| Rysunek 3. Łączenie z serwerem MS SQL Server Managment. |

Z poziomu Microsoft SQL Server Managment wygenerowano skrypt z przykładową strukturą bazy danych (plik o nazwie*: car\_rental.sql*). Tabele odpowiednio: klienci   
(rys. 4), wypożyczenia (rys. 5), samochody (rys. 6) zostały zaprezentowane poniżej:

|  |  |  |
| --- | --- | --- |
|  | | |
| Rysunek 4. Tabela dbo.customer zawierająca wszystkich klientów. | | |
|  |
| Rysunek 5. Tabela dbo.rental zawierająca wypożyczenia. |
|  |
| Rysunek 6. Tabela dbo.car zawierająca dostępne w bazie samochody. |

ZAŁOŻENIA PROJEKTOWE – FUNKCJONALNOŚĆ

W projekcie aplikacji założyłyśmy dwie role aplikacyjne z podziałem na poszczególne funkcjonalności, które zostały przedstawione na schemacie (rys. 7) oraz opisane poniżej:

* **ADMIN** - osoba zarządzająca stroną posiadająca najwięcej uprawnień   
  (m.in. usuwanie klientów, samochodów, wypożyczeń z bazy oraz ich dodawanie)
* **USER** - klient, osoba korzystająca z aplikacji, która ma możliwość logowania, przegląd oferty samochodów w wypożyczalni oraz dodawanie swojego wypożyczenia.

Dodatkowo, jeśli użytkownik się nie zaloguje, może przeglądać samochody, które są na stanie wypożyczalni.

|  |
| --- |
|  |
| Rysunek 7. Diagram sekwencji zdarzeń. |

Fragment klasy *SecurityConfig* przedstawiający role aplikacyjne (admin i user) oraz ich podział na dane funkcjonalności został przedstawiony poniżej (rys. 8):

|  |
| --- |
|  |
| Rysunek 8. Fragment kodu przedstawiający role aplikacyjne. |

Kodowanie hasła zostało wykonane w następujący sposób (rys. 9 - 10):

|  |
| --- |
|  |
| Rysunek 9. Kodowanie hasła. |
|  |
| Rysunek 10. Kodowanie hasła – przykładowy klient. |

Rezultat kodowania został przedstawiony poniżej (rys. 11):

|  |
| --- |
|  |
| Rysunek 11. Rezultat kodowania hasła – przykładowy klient. |

Przykładowe przypadki użycia dla każdej grupy przedstawiono na rysunkach 12., 13. i 14.

|  |
| --- |
|  |
| Rysunek 12. Przykład interakcji użytkownika niezalogowanego z aplikacją. Użytkownik niezalogowany może jedynie wyświetlić samochody, które wypożyczalnia ma w swojej ofercie oraz zapoznać się z ogólnymi informacjami zamieszczonymi na stronie. Wszelkie inne próby zakończą się odmową dostępu. |
|  |
| Rysunek 13. Diagram UML działań użytkownika. Na diagramie przedstawiono przykładowe dwa działania, jakie może wykonać użytkownik. Pierwszym z nich (punkty od 1 do 4) jest zalogowanie się do aplikacji. Dane, jakie wpisuje użytkownik są porównywane z przechowywanymi w bazie danych i zakładając, że są poprawnie wpisane, użytkownik zostaje zalogowany do aplikacji, co daje mu możliwość, m.in. wypożyczenia samochodu (punkty 5-8). Żądanie wypożyczenia samochodu skutkuje utworzeniem nowego rekordu w tabeli rentals. |
|  |
| Rysunek 14. Przykład interakcji administratora z aplikacją. Administrator może wykonywać wszystkie oferowane przez aplikację działania. Na rysunku powyżej przedstawiono jedynie dwa z nich, tj. zapisywanie nowego samochodu do bazy danych (punkty 1-4) oraz wyświetlenie wszystkich rekordów z tabeli wypożyczeń (punkty 5-8). |

Schemat struktury bazy danych przedstawiono na rys. 15.

|  |
| --- |
|  |
| Rysunek 15. Schemat zastosowanej bazy danych z wyróżnionymi relacjami łączącymi tabelę rentals z tabelą cars oraz tabelę rentals z tabelą customers. Symbol klucza oznacza primary key, niebieski romb – pola wymagane (not null), biały romb z niebieskimi konturami – niewymagane pole, brak ikony – foreign key. |