



RENTALS

INTERNETOWA APLIKACJA BAZODANOWA,
OBSŁUGUJĄCA WYPOŻYCZALNIĘ POJAZDÓW

Streszczenie

Dokumentacja techniczna opisująca strukturę oraz działanie aplikacji internetowej, oraz bazy danych

Maciej Myszka, Katarzyna Matla, Filip Lichwała

SPIS TREŚCI.....	I
1. Krótki opis projektu	2
2. Widok aplikacji	2
2.1. Gość.....	2
2.1.1. Wyświetlanie listy pojazdów.....	3
2.1.2. Menedżer konta	4
2.1.2.1. Logowanie	4
2.1.2.2. Rejestracja	5
2.1.2.3. Wylogowanie.....	7
2.2. Użytkownik.....	8
2.2.1. Wypożyczanie pojazdu	8
2.2.2. Wyświetlanie historii wypożyczeń	10
2.2.2.1. Ocena pojazdu	10
2.2.2.2. Zwrot pojazdu	11
2.3. Pracownik.....	11
2.3.1. Dodawanie pojazdu	11
2.3.2. Ogólna historia wypożyczeń	16
2.4. Administrator	17
2.4.1. Historia zmian.....	17
3. Struktura bazy danych.....	19
3.1. Schemat relacji	19
3.2. Budowa przez sql.....	21
3.3. Funkcje	27
3.3.1. Procedury składowe.....	27
3.3.2. Wyzwalacze.....	30

I. KRÓTKI OPIS PROJEKTU

Aplikacja **Rentals**, jest internetową aplikacją bazodanową, zbudowaną na strukturze projektu BLAZOR w środowisku .Net 8.0, używając przy tym grupy rozwiązań pisanych w językach C#, JavaScript, HTML oraz CSS.

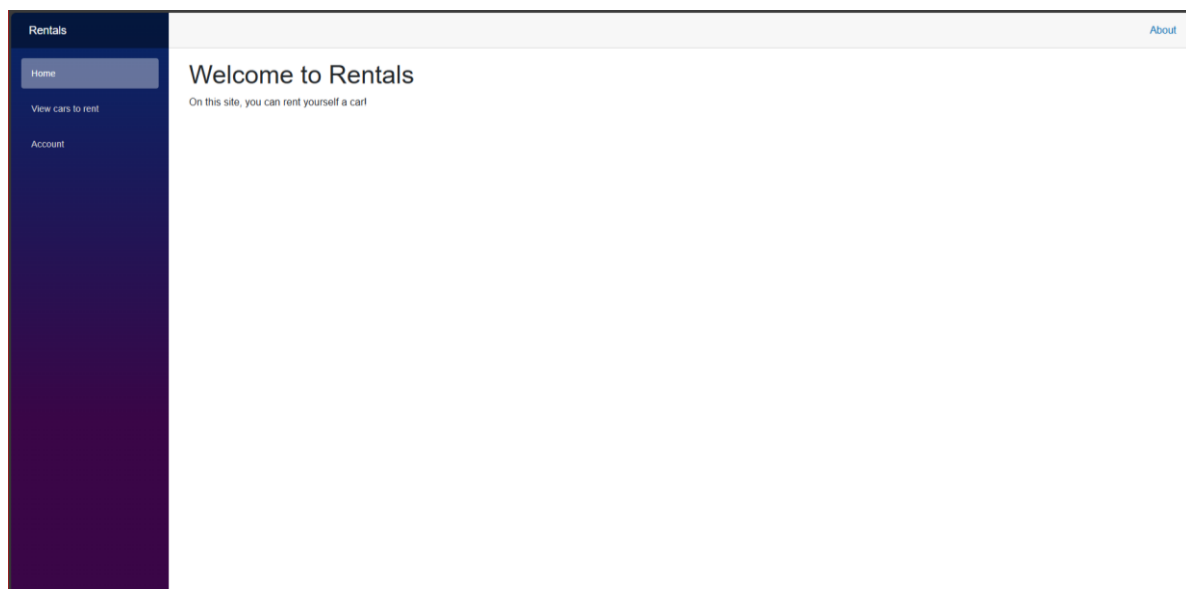
Zbudowana została w celu prezentacji sposobu obsługi bazy danych przy użyciu aplikacji internetowej, wykorzystującej takie zasoby jak pakiety Npgsql, NavigationManager, EntityFramework oraz BCrypt przykładowo w celu łączenia aplikacji z bazą, wykonywania poleceń w bazie, nadania kontroli nad całym rozwiązaniem między projektami, bądź umożliwieniem szyfrowania danych.

Aplikacja pozwala na wykonywanie różnych czynności, w zależności od poziomu przywilejów użytkownika; użytkownik do momentu zalogowania bądź rejestracji, zaczyna jako Gość, po czym może dokonać wypożyczenia pojazdu oraz wyświetlenia historii swoich wypożyczeń, gdzie będzie mógł zwrócić pojazd i nadać mu wybraną przez siebie ocenę. Pracownik będzie mógł nad to dodać do bazy nowy pojazd, oraz wyświetlić ogólną historię wszystkich wypożyczeń, a Administrator pozyska możliwość wyświetlenia historii zmian na bazie.

2. WIDOK APLIKACJI

2.1. GOŚĆ

Widok bazowy strony:



Gość dostaje możliwość [wyświetlenia listy pojazdów](#) dostępnych do wypożyczenia, oraz uruchomienia [menedżera konta](#).

2.1.1. WYŚWIETLANIE LISTY POJAZDÓW

Widok strony:

Car Make	Car Model	Price For Day	Rating
Skoda	Octavia	200	Rating: 0 - Ratings: 0
Autko Brum	Mieszkin Brum	69	Rating: 0 - Ratings: 0
Skoda	Octavia	200	Rating: 3,8 - Ratings: 5
Mazda	Mx5	100	Rating: 3 - Ratings: 3
Mazda	Mx5	1000	Rating: 5 - Ratings: 1
Mazda	Mx5	500	Rating: 0 - Ratings: 0
Audi	Q7	400	Rating: 5 - Ratings: 1
Dodge	Charger	500	Rating: 3,75 - Ratings: 4

Po wybraniu podstrony „View Cars to rent”, użytkownikowi strony wyświetla się tabela pojazdów dostępnych do wypożyczenia; tabela zawiera informacje od marce pojazdu, jego modelu, ceny wynajmu pojazdu za dzień oraz jego oceny. Stąd użytkownik może wybrać pojazd który chciałby wypożyczyć. poprzez kliknięcie na wybrany wiersz i wyświetlić dodatkowe informacje na temat wypożyczenia (sam proces wypożyczania będzie przedstawiony w sekcji [Użytkownik/Wypożyczenie pojazdu](#)):

Rent a Car

Car Make	Car Model	Price For Day	Rating
Skoda	Octavia	200	Rating: 0 - Ratings: 0
Autko Brum	Mieszkin Brum	69	Rating: 0 - Ratings: 0
Skoda	Octavia	200	Rating: 3,8 - Ratings: 5
Mazda	Mx5	100	Rating: 3 - Ratings: 3
Mazda	Mx5	1000	Rating: 5 - Ratings: 1
Mazda	Mx5	500	Rating: 0 - Ratings: 0
Audi	Q7	400	Rating: 5 - Ratings: 1
Dodge	Charger	500	Rating: 3,75 - Ratings: 4


Leather Upholstery - 100 ☐

Smoked Windows - 200 ☐

Spare Wheel - 500 ☐

Renting Length: days

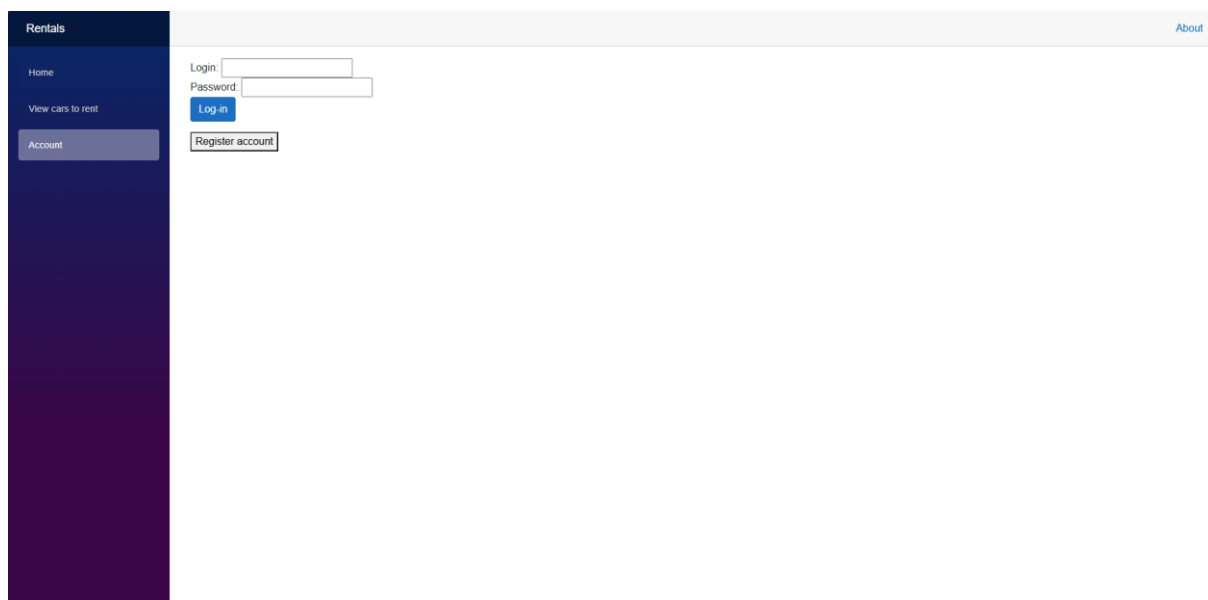
Total Price: 500

Zakreślony na jasnoszary  kolor wiersz jest wybranym pojazdem.

Obecnie, użytkownik mógł kliknąć przycisk *Rent*, jednakże, z uwagi na fakt że dotychczas jest zalogowany jako Gość, nie będzie mógł dokonać wypożyczenia; z tej uwagi kliknięcie na przycisk przekieruje go do [menedżera konta](#).

2.1.2. MENEDŻER KONTA

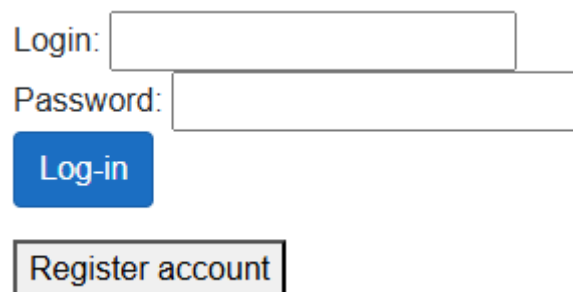
Widok strony:



The screenshot shows a web application interface. On the left is a dark blue sidebar with the title 'Rentals' at the top. Below it are links: 'Home', 'View cars to rent', and 'Account' (which is highlighted). On the right, there is a light gray header with a link 'About' on the far right. Below the header, there is a login form with two input fields: 'Login:' and 'Password:'. Below these fields are two buttons: a blue 'Log-in' button and a gray 'Register account' button.

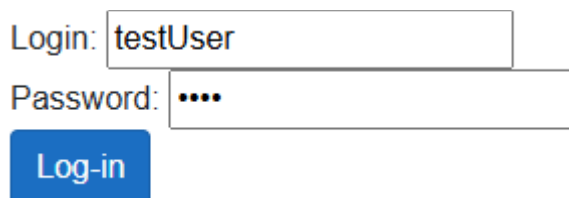
Stąd, użytkownik będzie mógł dokonać [logowania](#), bądź [rejestracji](#) konta.

2.1.2.1. LOGOWANIE



This is a close-up of the login form. It features two input fields: 'Login:' and 'Password:'. Below the 'Login:' field is a blue 'Log-in' button. Below the 'Password:' field is a gray 'Register account' button.

Użytkownik jest poproszony o podanie loginu i hasła; ma też możliwość przełączenia na stronę [rejestracji](#).



This is a close-up of the login form with test data. The 'Login:' field contains the text 'testUser'. The 'Password:' field contains four dots '....'. Below the 'Login:' field is a blue 'Log-in' button.

Użytkownik wpisuje swój login i hasło, po czym może kliknąć przycisk *Log-in*, aby wykonać logowanie.

Login:

Password:

Failed to log in. Login or password incorrect.

Jeśli użytkownik podał błędny login bądź hasło, zostanie o tym poinformowany; Po pomyślnym logowaniu, zostanie przeniesiony do swojego Panelu Użytkownika, czyli powitanie, przycisk do [wylogowania](#), oraz [historia wypożyczeń](#):

Hello, Test!

No rentals found.

2.1.2.2. REJESTRACJA

Create Login:

Create Password:

Użytkownik poproszony jest o podanie wybranego przez siebie loginu oraz hasła; ma też możliwość przełączenia na stronę [logowania](#).

Create Login:

Create Password:

Użytkownik wpisuje podany login oraz hasło, po czym może kliknąć przycisk *Register*, by dokonać rejestracji.

Create Login:

Create Password:

Please input valid login and password.

Jeżeli użytkownik nie poda obu, loginu i hasła, zostanie poproszony o podanie prawidłowych.

Create Login:

Create Password:

[Register](#)

Chosen login is in use. Try another

Jeżeli użytkownik podał prawidłowe dane, ale login jest już zajęty, poproszony będzie o podanie nowego.

Create Login:

Create Password:

[Register](#)

Jeśli nowo podane dane są prawidłowe, użytkownik zostanie przeniesiony do drugiego etapu logowania, gdzie będzie poproszony o podanie swoich danych:

Name:

Surname:

Email:

PESEL:

Phone Number:

[Register](#)

Użytkownik następnie wpisuje swoje dane, przykładowo:

Name:

Surname:

Email:

PESEL:

Phone Number:

[Register](#)

Po wpisaniu danych, użytkownik może kliknąć przycisk *Register*, tym samym rejestrując swoje konto w bazie i natychmiastowo logując się na nie. Istotnym do dodania jest, że po wpisaniu błędnego numeru pesel bądź numeru telefonu, tj. znaków które nie są cyframi i o niepoprawnej długości, użytkownik nie będzie mógł dokonać rejestracji:

Name:

Surname:

Email:

PESEL:

Phone Number:

Po pomyślnej rejestracji, użytkownik pozostaje natychmiastowo zalogowany.

2.1.2.3. WYLOGOWANIE

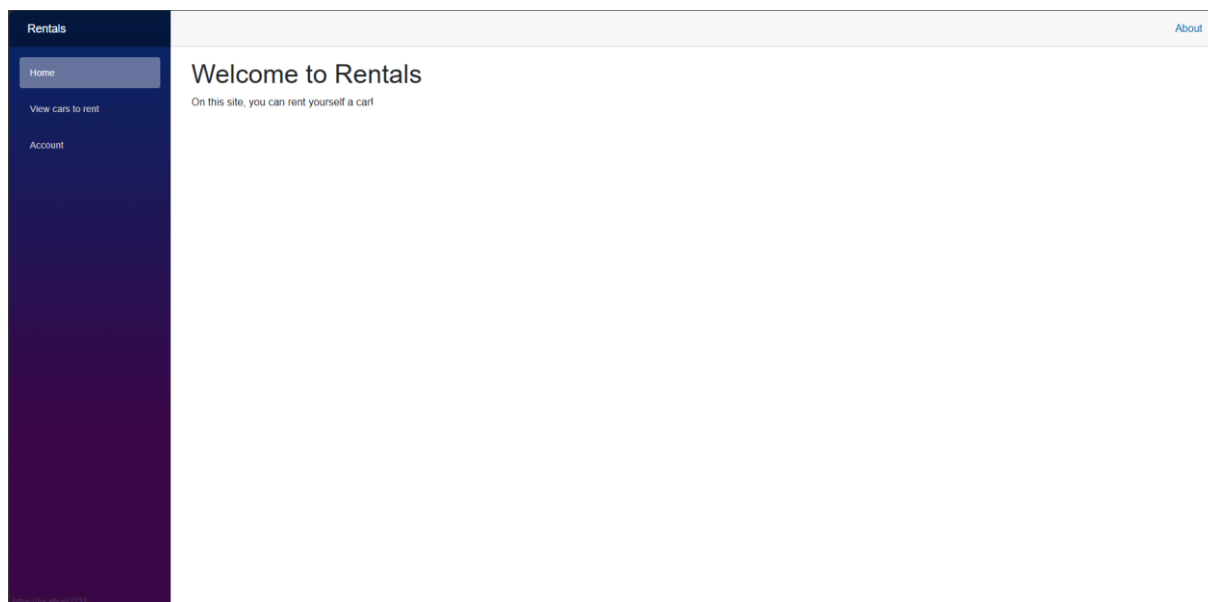
Hello, Test!

No rentings found.

Użytkownik może kliknąć przycisk *Log-out*, aby wylogować się ze swojego konta; po jego kliknięciu zostanie cofnięty do panelu [logowania](#).

2.2. UŻYTKOWNIK

Widok bazowy strony:



Użytkownik pozyskuje możliwości które ma Gość, oraz możliwości [wypożyczania pojazdów](#), oraz wyświetlenia swojej [historii wypożyczeń](#).

2.2.1. WYPOŻYCZANIE POJAZDU

W celu wypożyczenia pojazdu, użytkownik będzie musiał odwołać się do wcześniej opisanego [wyświetlenia pojazdu](#).

Rent a Car

Car Make	Car Model	Price For Day	Rating
Skoda	Octavia	200	Rating: 0 - Ratings: 0
Autko Brum	Mieszkin Brum	69	Rating: 0 - Ratings: 0
Skoda	Octavia	200	Rating: 3,8 - Ratings: 5
Mazda	Mx5	100	Rating: 3 - Ratings: 3
Mazda	Mx5	1000	Rating: 5 - Ratings: 1
Mazda	Mx5	500	Rating: 0 - Ratings: 0
Audi	Q7	400	Rating: 5 - Ratings: 1
Dodge	Charger	500	Rating: 3,75 - Ratings: 4

Leather Upholstery - 100 ☐

Smoked Windows - 200 ☐

Spare Wheel - 500 ☐

Renting Length: days

Total Price: 500

Zakreślony na jasnoszary ☐ kolor wiersz jest wybranym pojazdem.

Dodge	Charger
Leather Upholstery - 100 <input type="checkbox"/> Smoked Windows - 200 <input type="checkbox"/> Spare Wheel - 500 <input type="checkbox"/> Renting Length: <input type="text" value="1"/> days Total Price: 500 <button>Rent</button>	

Tutaj użytkownik może zobaczyć szczegóły dotyczące wynajmu, tj. możliwe udogodnienia, czas wypożyczenia pojazdu, oraz sumę którą będzie musiał zapłacić za wynajem przy swoim wyborze.

Dodge	Charger
Leather Upholstery - 100 <input checked="" type="checkbox"/> Smoked Windows - 200 <input type="checkbox"/> Spare Wheel - 500 <input checked="" type="checkbox"/> Renting Length: <input type="text" value="4"/> days Total Price: 2600 <button>Rent</button>	

Tutaj widzimy że użytkownik wybrał skórzaną tapicerkę, zapasowe koło, oraz cztery dni okresu wynajmu. Cena która mu się wyświetla, jest równa sumie ceny wynajmu za dzień pomnożonej razy długość wynajmu i sumy wartości wybranych udogodnień. Następnie użytkownik będzie mógł zdecydować się na wynajem pojazdu, poprzez kliknięcie przycisku *Rent*. Po kliknięciu przycisku, pojazd zniknie z listy i zostanie przypisany do [historii wypożyczeni użytkownika](#).

2.2.2. WYŚWIETLANIE HISTORII WYPOŻYCZEŃ

Widok strony:

The screenshot shows a web application interface for 'Rentals'. On the left is a dark blue sidebar with links: 'Home', 'View cars to rent', and 'Account'. The main content area has a header with 'Hello, Test!' and a 'Log-out' button. Below this is a table with two columns: 'Car Mark' and 'Car Model'. The first row shows 'Dodge' and 'Charger'. The table is currently empty of data rows.

Tutaj użytkownik zobaczy każdy poprzednio wypożyczony przez siebie pojazd w postaci tabeli. Informacje są zwracane dla użytkownika jako View utworzone w bazie, które wykorzystuje jego id aby wyświetlić jedynie te dane które go dotyczą; użytkownik może kliknąć na wypożyczenie, aby pozyskać więcej informacji na jego temat:

Dodge	Charger
Rent Date and Length: 19.06.2024 - 4	Paid Amount: 2600
Given Rating: 3	Submit Rating
Returned: False	Return Car

Użytkownikowi wyświetla się data i długość wypożyczenia, zapłacona kwota, [ocena pojazdu](#) którą przyznał dla danego wypożyczenia, oraz informacje dotyczące [zwrotu pojazdu](#).

2.2.2.1. OCENA POJAZDU

The screenshot shows a rating interface. It displays 'Given Rating: 3' next to a slider bar that ranges from 1 to 5. The slider is currently positioned at 3. To the right of the slider is a blue button labeled 'Submit Rating'.

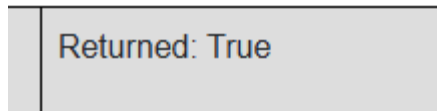
Użytkownik może zobaczyć jaką ocenę przyznał wypożyczeniu, (bazowo jest to 3), oraz dostaje możliwość przyznania mu oceny od 1 do 5 przy użyciu suwaka; po wybraniu wartości która mu odpowiada, użytkownik może kliknąć przycisk *Submit Rating*, aby zapisać swoją ocenę.

The screenshot shows the same rating interface as before, but the slider is now positioned at 5. The text 'Given Rating: 5' is displayed, and the 'Submit Rating' button remains visible.

2.2.2.2. ZWROT POJAZDU

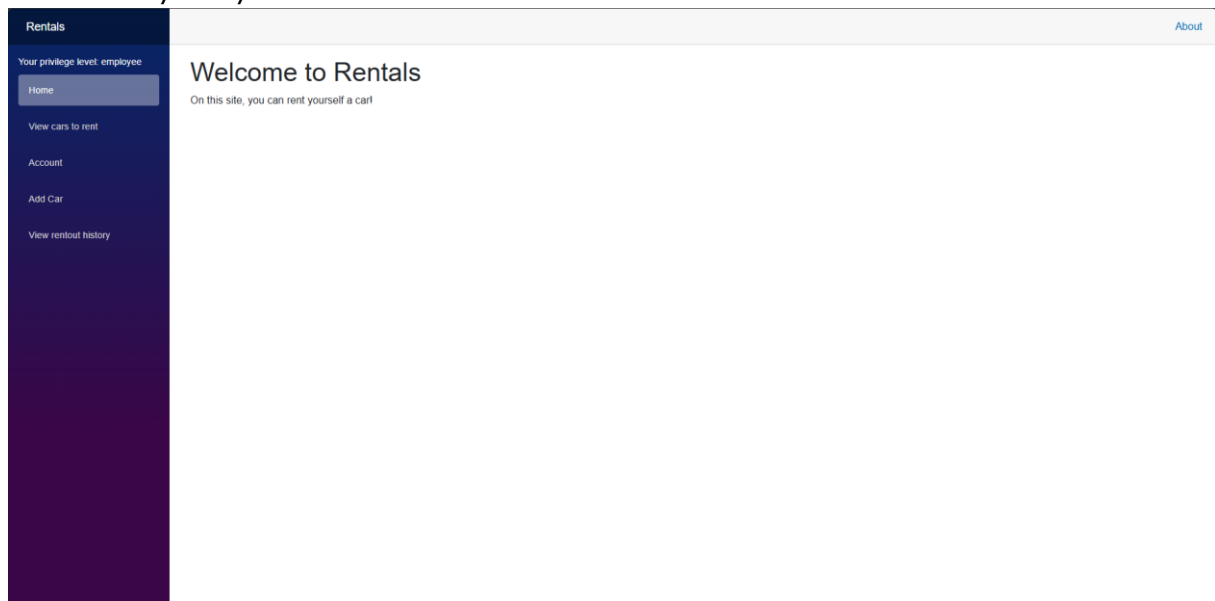


Użytkownik dostaje informacje czy pojazd który wypożyczył, został zwrócony; jeśli nie, może go zwrócić przy pomocy przycisku *Return Car*. Po jego kliknięciu pojazd zostaje oznaczony jako zwrócony, a przycisk znika.



2.3. PRACOWNIK

Widok bazowy strony:



Pracownik dostaje wszystkie możliwości Użytkownika, oraz możliwość [dodawania do bazy nowych pojazdów](#), jak również [wyświetlania ogólnej historii wypożyczeń](#). Jest również poinformowany o swoim poziomie dostępu.

2.3.1. DODAWANIE POJAZDU

Widok strony:

Rentals

Your privilege level: employee

Home

View cars to rent

Account

Add Car

View rental history

About

Add Car

Refresh

Found Cars: 8

Add New Car

Car Make: n/a

Car Model: n/a

Car Price: 0

Car Commodity 1: n/a

Price: 0

Car Commodity 2: n/a

Price: 0

Car Commodity 3: n/a

Price: 0

Add New Car

Tutaj użytkownik widzi ilość znalezionych pojazdów w bazie, możliwość odświeżenia tej informacji, oraz formularz danych do podania w celu dodania nowych pojazdów.

Add New Car

Car Make: n/a

Car Model: n/a

Car Price: 0

Car Commodity 1: n/a

Price: 0

Car Commodity 2: n/a

Price: 0

Car Commodity 3: n/a

Price: 0

Add New Car

Użytkownik proszony jest o podanie danych informacji w celu dodania nowego pojazdu do bazy; Wymaganyimi danymi są marka, model oraz cena za dzień, gdzie udogodnienia są opcjonalne:

Add New Car

Car Make:

Car Model:

Car Price:

Car Commodity 1: ▼

Price: 0

Car Commodity 2: ▼

Price: 0

Car Commodity 3: ▼

Price: 0

Użytkownik nie podał ceny, więc nie ma możliwości dodania pojazdu do bazy.

Add New Car

Car Make:

Car Model:

Car Price:

Car Commodity 1: ▼

Price: 0

Car Commodity 2: ▼

Price: 0

Car Commodity 3: ▼

Price: 0

Po wpisaniu prawidłowych danych, użytkownik może już dodać pojazd, aczkolwiek przed tym, może wybrać udogodnienia dla dodawanego pojazdu:

Add New Car

Car Make:

Car Model:

Car Price:

Car Commodity 1:

Price: 0

Car Commodity 2:

Price: 0

Car Commodity 3:

Price: 0

Select option

n/a

Scent (Ask in person)

Leather Upholstery

Smoked Windows

Alloy Wheels

Decorated Ceiling

Spare Wheel

Electric Car

Wyświetlane są one w postaci listy; w momencie wyboru jednego z nich, nie jest ono ponownie dostępne do wyboru.

Add New Car

Car Make:

Car Model:

Car Price:

Car Commodity 1:

Price: 5

Car Commodity 2:

Price: 0

Car Commodity 3:

Price: 0

Select option

n/a

Leather Upholstery

Smoked Windows

Alloy Wheels

Decorated Ceiling

Spare Wheel

Electric Car

Po wykonaniu wyboru udogodnień, użytkownik może kliknąć *Add New Car*, tym samym dodając nowy pojazd do bazy.

Add Car

Refresh

Found Cars: 9

Add New Car

Car Make:
Car Model:
Car Price:
Car Commodity 1:
Price: 5
Car Commodity 2:
Price: 100
Car Commodity 3:
Price: 0

Add New Car

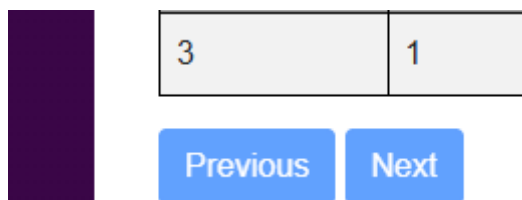
Widzimy że po kliknięciu przycisku wartość *Found Cars* wzrosła o 1, tym samym wiemy że auto zostało doane pomyślnie do bazy.

2.3.2. OGÓLNA HISTORIA WYPOŻYCZEŃ

Widok strony:

Renting ID	Car ID	User ID	Car Price	Chosen Commodities	Commodities Full Price	Full Rent Price	Rent Date	Rent Length	Rent Rating
34	66	19	500	Leather Upholstery - 100, Spare Wheel - 500	600	2600	2024-06-19	4	5
33	66	17	500	Spare Wheel - 500	500	2000	2024-06-19	3	5
32	62	16	200	Scent (Ask in person) - 5, Electric Car - 700	705	1505	2024-06-19	4	5
31	62	16	200		0	600	2024-06-19	3	1
30	62	16	200		0	400	2024-06-19	2	5
29	63	13	100		0	200	2024-06-18	2	3
28	67	7	400	Electric Car - 700, Decorated Ceiling - 450	1150	2350	2024-06-18	3	5
27	62	6	200	Smoked Windows - 200, Electric Car - 700	900	1500	2024-06-18	3	5
26	66	12	500	Smoked Windows - 200, Spare Wheel - 500	700	2200	2024-06-18	3	3
25	66	12	500	Leather Upholstery - 100, Spare Wheel - 500	1600	1600	2024-06-18	2	5
24	66	0	500	Leather Upholstery - 100, Spare Wheel - 500	2600	2600	2024-06-18	4	2
23	63	0	100		200	200	2024-06-18	2	3
22	63	6	100		300	300	2024-06-18	3	5
21	63	0	100		300	300	2024-06-17	3	1
20	61	0	69		138	138	2024-06-17	2	3
19	64	0	60		367	367	2024-06-17	2	3

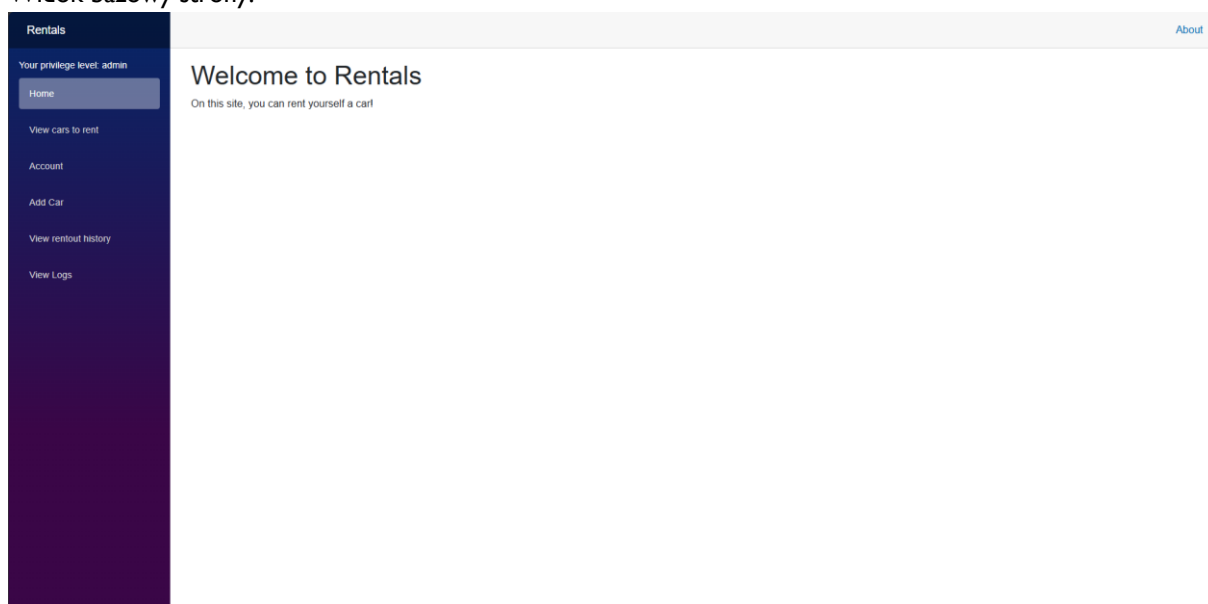
Użytkownikowi pokazuje się kolejno: ilość znalezionych wypożyczeń, przycisk odświeżający tą wartość i historia wszystkich uprzednio wykonanych wypożyczeń. Jest to tabela, która zwraca mu informacje dotyczące ID wypożyczenia, użytkownika oraz pojazdu, wybrane dla niego udogodnienia, pełna kwota wydana na udogodnienia, pełna cena wypożyczenia, jego data i długość, oraz ocena. Tabela ma limit rekordów na stronie równy 50; w przypadku przebicia limitu rekordów na stronę, użytkownik może przejść między kolejną i poprzednią stroną rekordów, przy użyciu przycisków na dole strony:



Przycisk *Previous*, pozwala wyświetlać poprzednią stronę historii; przycisk *Next*, następną. Przycisk *Previous*, jest obecnie wyłączony, gdyż znajdujemy się na pierwszej stronie wpisów. Przycisk *Next* jest wyłączony, gdyż ilość znalezionych wpisów nie jest większa od limitu wpisów na stronę.

2.4. ADMINISTRATOR

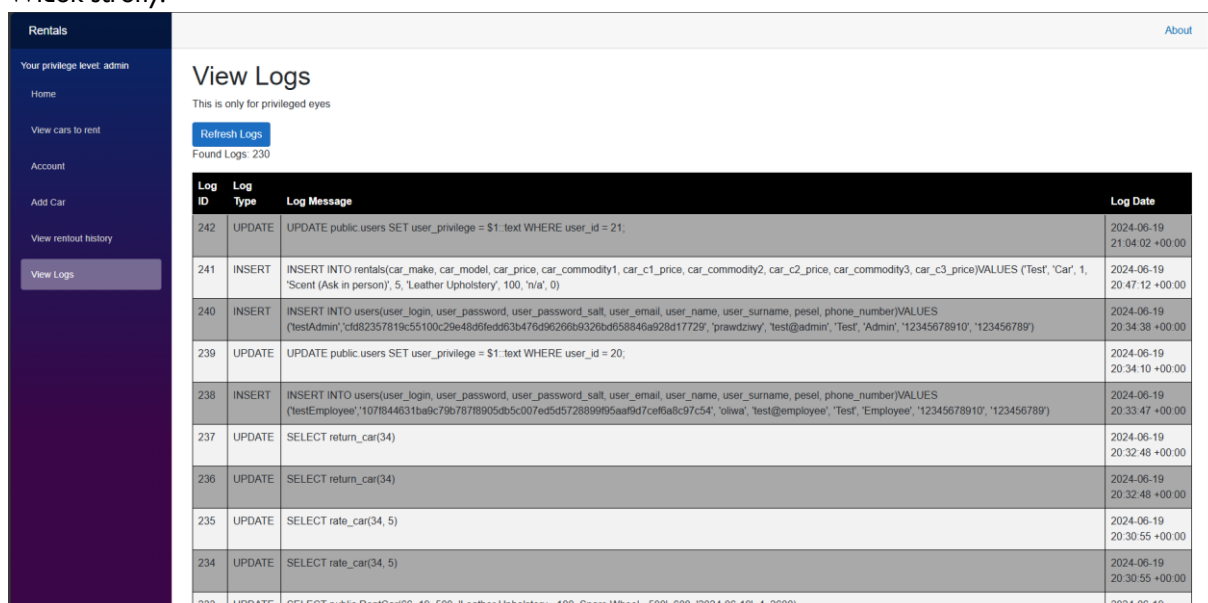
Widok bazowy strony:



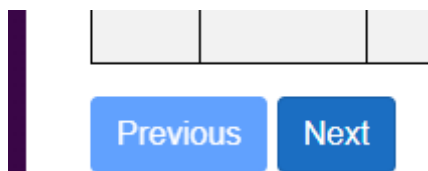
Administrator posiada wszystkie zdolności pracownika, oraz dostęp do [historii zmian](#). On również, jak pracownik ma podany swój stopień uprzywilejowania.

2.4.1. HISTORIA ZMIAN

Widok strony:



Użytkownikowi pokazuje się kolejno: ilość znalezionych wpisów, przycisk do odświeżania tej liczby, oraz historia wszystkich zmian. Jest to tabela która zwraca mu: id wpisu, typ wpisu, na podstawie typu operacji wykonanej w bazie, kwerenda która wykonała te zmiany, oraz data operacji. Tabela ma limit rekordów na stronie równy 50; w przypadku przebicia limitu rekordów na stronę, użytkownik może przejść między kolejną i poprzednią stroną rekordów, przy użyciu przycisków na dole strony:



Przycisk *Previous*, pozwala wyświetlać poprzednią stronę historii; przycisk *Next*, następną. Przycisk *Previous*, jest obecnie wyłączony, gdyż znajdujemy się na pierwszej stronie wpisów. Po kliknięciu przycisku *Next*, widzimy kolejną stronę tablicy (możemy to rozpoznać po numerze najwyższego wpisu):

Rentals

Your privilege level: admin

Home

View cars to rent

Account

Add Car

View rentout history

View Logs

View Logs

This is only for privileged eyes

Refresh Logs

Found Logs: 230

Log ID	Log Type	Log Message	Log Date
192	INSERT	SELECT public.RentCar(62, 16, 200, 'Scent (Ask in person)' - 5, Electric Car - 700', 705, '2024-06-19', 4, 1505)	2024-06-19 16:20:58 +00:00
191	UPDATE	SELECT rate_car(30, 5)	2024-06-19 16:18:20 +00:00
190	UPDATE	SELECT rate_car(30, 5)	2024-06-19 16:18:20 +00:00
189	UPDATE	SELECT rate_car(31, 1)	2024-06-19 16:18:17 +00:00
188	UPDATE	SELECT rate_car(31, 1)	2024-06-19 16:18:17 +00:00
187	UPDATE	SELECT return_car(31)	2024-06-19 16:18:07 +00:00
186	UPDATE	SELECT return_car(31)	2024-06-19 16:18:07 +00:00

Teraz, po wciśnięciu odblokowanego już przycisku *Previous*, wracamy na poprzednią stronę:



Rentals

Your privilege level: admin

Home

View cars to rent

Account

Add Car

View rentout history

View Logs

View Logs

This is only for privileged eyes

Refresh Logs

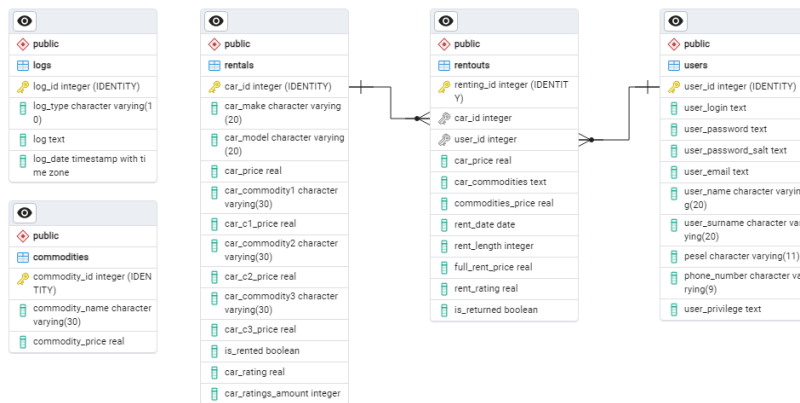
Found Logs: 230

Log ID	Log Type	Log Message	Log Date
242	UPDATE	UPDATE public.users SET user_privilege = \$1::text WHERE user_id = 21;	2024-06-19 21:04:02 +00:00
241	INSERT	INSERT INTO rentals(car_make, car_model, car_price, car_commodity1, car_c1_price, car_commodity2, car_c2_price, car_commodity3, car_c3_price)VALUES ('Test', 'Car', 1, 'Scent (Ask in person)', 5, 'Leather Upholstery', 100, 'n/a', 0)	2024-06-19 20:47:12 +00:00
240	INSERT	INSERT INTO users(user_login, user_password, user_password_salt, user_email, user_name, user_surname, pesel, phone_number)VALUES ('testAdmin', 'cd92357819c55100c29e48d9fedd53b478d9626669328bd658846a928d17729', 'prawdziwy', 'test@admin', 'Test', 'Admin', '12345678910', '123456789')	2024-06-19 20:34:38 +00:00
239	UPDATE	UPDATE public.users SET user_privilege = \$1::text WHERE user_id = 20;	2024-06-19 20:34:10 +00:00
238	INSERT	INSERT INTO users(user_login, user_password, user_password_salt, user_email, user_name, user_surname, pesel, phone_number)VALUES ('testEmployee', '107f844631ba9c79b787f8905db5c007ed5d5728899f95aafdf7cef0a8c97c54', 'oliwa', 'test@employee', 'Test', 'Employee', '12345678910', '123456789')	2024-06-19 20:33:47 +00:00
237	UPDATE	SELECT return_car(34)	2024-06-19 20:32:48 +00:00
236	UPDATE	SELECT return_car(34)	2024-06-19 20:32:48 +00:00
235	UPDATE	SELECT rate_car(34, 5)	2024-06-19 20:30:55 +00:00
234	UPDATE	SELECT rate_car(34, 5)	2024-06-19 20:30:55 +00:00
233	UPDATE	SELECT public.RentCar(66, 19, 500, 'Leather Upholstery - 100, Spare Wheel - 500', 600, '2024-06-19', 4, 2600)	2024-06-19

3. STRUKTURA BAZY DANYCH

3.1. SCHEMAT RELACJI

Poniższy obraz prezentuje graficznie schemat relacyjny bazy.



Skrócony opis:

Schemat składa się z pięciu tablic:

logs, która przechowuje wpisy zmian w bazie; zbudowana jest z kolumn:

log_id; INT; jest to klucz podstawowy typu liczbowego o auto inkrementacji

log_typ; VARCHAR(10); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera dane dotyczące typu wpisu

log; TEXT; jest to typ tekstowy bez limitu znaków; w zamyśle zawiera dane informacyjne wpisu

log_date; TIMESTAMPZ; jest to typ daty, uwzględniający strefę czasową; zawiera dane dokonania wpisu

commodities, która przechowuje możliwe dla pojazdów udogodnienia; zbudowana jest z kolumn:

commodity_id; INT; jest to klucz podstawowy typu liczbowego o auto inkrementacji

commodity_name; VARCHAR(30); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera nazwę udogodnienia

commodity_price; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje cenę udogodnienia

rentals, która przechowuje pojazdy do wypożyczenia; składa się z kolumn:

car_id; INT; jest to klucz podstawowy typu liczbowego o auto inkrementacji

car_make; VARCHAR(20); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera markę pojazdu

car_model; VARCHAR(20); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera model pojazdu

car_price; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje cenę wynajmu pojazdu za dzień

car_commodity1 (/2/3); VARCHAR(30); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera jedno z udogodnień pojazdu

car_cl (/2/3)_price; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje cenę danego udogodnienia

is_rented; BOOL; jest to typ logiczny, który zwraca prawdę lub fałsz; w zamyśle przechowuje informację o tym czy auto jest wypożyczone

car_rating; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje ocenę pojazdu

car_ratings_amount; INT; jest to typ liczbowy; w zamyśle przechowuje ilość ocen pojazdu

rentouts, która przechowuje wszystkie poprzednie wypożyczenia; składa się z kolumn:

renting_id; INT; jest to klucz podstawowy typu liczbowego o auto inkrementacji

car_id; INT; jest to klucz obcy typu liczbowego, pobierający z *rentals.car_id*; jest to relacja typu jeden (*rentals.car_id*) do wielu (*rentouts.user_id*)

user_id; INT; jest to klucz obcy typu liczbowego, pobierający z *users.user_id*; jest to relacja typu jeden (*users.user_id*) do wielu (*rentouts.user_id*)

car_price; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje cenę bazową wypożyczanego pojazdu

car_commodities; TEXT; jest to typ tekstowy bez limitu znaków; w zamyśle zawiera informacje dotyczące wybranych udogodnień podczas tworzenia wypożyczenia

commodities_price; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje sumę cen wybranych udogodnień

rent_date; DATE; jest to podstawowy typ daty; w zamyśle przechowuje informacje dotyczącą daty wykonania wypożyczenia

rent_length; INT; jest to typ liczbowy; w zamyśle przechowuje informację dotyczącą długości wynajmu pojazdu

full_rent_price; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje pełną cenę wypożyczenia

rent_rating; REAL; jest to typ liczbowy, pozwalający na wartości po przecinku; w zamyśle przechowuje ocenę wypożyczenia

is_returned; BOOL; jest to typ logiczny, który zwraca prawdę lub fałsz; w zamyśle przechowuje informację o tym czy wypożyczone auto zostało zwrócone

users, która przechowuje dane użytkowników; składa się z kolumn:

user_id; INT; jest to klucz podstawowy typu liczbowego o auto inkrementacji

user_login; TEXT; jest to typ tekstowy bez limitu znaków; w zamyśle zawiera login do konta użytkownika

user_password; TEXT; jest to typ tekstowy bez limitu znaków; w zamyśle zawiera zaszyfrowane hasło konta użytkownika

user_password_salt; TEXT; jest to typ tekstowy bez limitu znaków; w zamyśle zawiera sól użytą do zaszyfrowania hasła do konta użytkownika

user_email; TEXT; jest to typ tekstowy bez limitu znaków; w zamyśle zawiera email użytkownika

user_name; VARCHAR(20); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera imię użytkownika

user_surname; VARCHAR(20); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera nazwisko użytkownika

pesel; VARCHAR(11); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera numer PESEL użytkownika

phone_number; VARCHAR(9); jest to typ tekstowy o określonym limicie znaków; w zamyśle zawiera numer telefonu użytkownika

user_privilege; TEXT; jest to typ tekstowy bez limitu znaków; w zamyśle zawiera poziom dostępu użytkownika

3.2. BUDOWA PRZES SQL

Poniższy kod prezentuje budowanie pustej bazy przy użyciu języka SQL:

```
BEGIN;
```

```
CREATE TABLE IF NOT EXISTS public.commodities
```

```
(
```

```
    commodity_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1  
MINVALUE 1 MAXVALUE 999 CACHE 1 ),
```

```
    commodity_name character varying(30) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
```

```
    commodity_price real NOT NULL DEFAULT 0,
```

```

        CONSTRAINT commodities_pkey PRIMARY KEY (commodity_id)
    )
WITH (
    OIDS = FALSE
)
CREATE TABLE IF NOT EXISTS public.logs
(
    log_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1
MAXVALUE 99999999 CACHE 1 ),
    log_type character varying(10) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
    log_text COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
    log_date timestamp with time zone NOT NULL DEFAULT '2024-01-01 12:00:00+00'::timestamp with time
zone,
    CONSTRAINT logs_pkey PRIMARY KEY (log_id)
)
WITH (
    OIDS = FALSE
)

CREATE TABLE IF NOT EXISTS public.rentals
(
    car_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1
MAXVALUE 99999999 CACHE 1 ),
    car_make character varying(20) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
    car_model character varying(20) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
    car_price real NOT NULL DEFAULT 1,
    car_commodity1 character varying(30) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
    car_c1_price real NOT NULL DEFAULT 0,
    car_commodity2 character varying(30) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
    car_c2_price real NOT NULL DEFAULT 0,
    car_commodity3 character varying(30) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,

```

```

car_c3_price real NOT NULL DEFAULT 0,
is_rented boolean NOT NULL DEFAULT false,
car_rating real NOT NULL DEFAULT 0,
car_ratings_amount integer NOT NULL DEFAULT 0,
CONSTRAINT "Rentals_pkey" PRIMARY KEY (car_id)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

```

```

ALTER TABLE IF EXISTS public.rentals
    OWNER to ---;

```

```

CREATE TRIGGER log_delete
    AFTER DELETE
    ON public.rentals
    FOR EACH STATEMENT
    EXECUTE PROCEDURE public.log_query();

```

```

CREATE TRIGGER log_insert
    AFTER INSERT
    ON public.rentals
    FOR EACH STATEMENT
    EXECUTE PROCEDURE public.log_query();

```

```

CREATE TRIGGER log_update
    AFTER UPDATE
    ON public.rentals
    FOR EACH STATEMENT

```



```
EXECUTE PROCEDURE public.log_query();
```

```
CREATE TABLE IF NOT EXISTS public.rentouts
```

```
(
```

```
    renting_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1  
MINVALUE 1 MAXVALUE 999999999 CACHE 1 ),
```

```
    car_id integer NOT NULL DEFAULT 0,
```

```
    user_id integer NOT NULL DEFAULT 0,
```

```
    car_price real NOT NULL DEFAULT 0,
```

```
    car_commodities text COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
```

```
    commodities_price real NOT NULL DEFAULT 0,
```

```
    rent_date date NOT NULL DEFAULT '2024-01-01'::date,
```

```
    rent_length integer NOT NULL DEFAULT 0,
```

```
    full_rent_price real NOT NULL DEFAULT 0,
```

```
    rent_rating real NOT NULL DEFAULT 3,
```

```
    is_returned boolean NOT NULL DEFAULT false,
```

```
    CONSTRAINT rentouts_pkey PRIMARY KEY (renting_id),
```

```
    CONSTRAINT fk_rentouts_car_id FOREIGN KEY (car_id)
```

```
        REFERENCES public.rentals (car_id) MATCH SIMPLE
```

```
        ON UPDATE CASCADE
```

```
        ON DELETE CASCADE
```

```
        NOT VALID,
```

```
    CONSTRAINT fk_rentouts_user_id FOREIGN KEY (user_id)
```

```
        REFERENCES public.users (user_id) MATCH SIMPLE
```

```
        ON UPDATE CASCADE
```

```
        ON DELETE CASCADE
```

```
        NOT VALID
```

```
)
```

```
WITH (
```

```
    OIDS = FALSE
```

)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.rentouts

OWNER to ---;

CREATE TRIGGER log_delete

AFTER DELETE

ON public.rentouts

FOR EACH STATEMENT

EXECUTE PROCEDURE public.log_query();

CREATE TRIGGER log_insert

AFTER INSERT

ON public.rentouts

FOR EACH STATEMENT

EXECUTE PROCEDURE public.log_query();

CREATE TRIGGER log_update

AFTER UPDATE

ON public.rentouts

FOR EACH STATEMENT

EXECUTE PROCEDURE public.log_query();

CREATE TABLE IF NOT EXISTS public.users

(

user_id integer NOT NULL GENERATED ALWAYS AS IDENTITY (INCREMENT 1 START 0 MINVALUE 0 MAXVALUE 999999999 CACHE 1),

user_login text COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,

user_password text COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,

```

user_password_salt text COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
user_email text COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
user_name character varying(20) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
user_surname character varying(20) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
pesel character varying(11) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
phone_number character varying(9) COLLATE pg_catalog."default" NOT NULL DEFAULT 'n/a'::text,
user_privilege text COLLATE pg_catalog."default" NOT NULL DEFAULT 'user'::text,
CONSTRAINT users_pkey PRIMARY KEY (user_id)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

```

```

ALTER TABLE IF EXISTS public.users
    OWNER to ---;

```

```

CREATE TRIGGER log_delete
    AFTER DELETE
    ON public.users
    FOR EACH STATEMENT
    EXECUTE PROCEDURE public.log_query();

```

```

CREATE TRIGGER log_insert
    AFTER INSERT
    ON public.users
    FOR EACH STATEMENT
    EXECUTE PROCEDURE public.log_query();

```

```

CREATE TRIGGER log_update

```

```

AFTER UPDATE

ON public.users

FOR EACH STATEMENT

EXECUTE PROCEDURE public.log_query();

ALTER TABLE IF EXISTS public.rentouts

ADD CONSTRAINT fk_rentouts_car_id FOREIGN KEY (car_id)

REFERENCES public.rentals (car_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

NOT VALID;

ALTER TABLE IF EXISTS public.rentouts

ADD CONSTRAINT fk_rentouts_user_id FOREIGN KEY (user_id)

REFERENCES public.users (user_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

NOT VALID;

END;

```

3.3. FUNKCJE

Baza zawiera kilka własnych funkcji, które zostały opisane poniżej, jako opis zadania funkcji oraz jako kod w języku pnpqsql.

3.3.1. PROCEDURY SKŁADOWE

Pierwszą funkcją jest rentcar; jej celem jest stworzenie nowego [wypożyczenia](#), oraz zaznaczenie wybranego do wypożyczenia pojazdu, jako wypożyczonego.

```

CREATE OR REPLACE FUNCTION public.rentcar(

    carid_ integer,

    userid_ integer,

    carprice_ real,

    carcommodities_ text,

    commoditiesprice_ real,

    rentdate_ date,

```

```

        rentlength_ integer,

        fullrentprice_ real)

RETURNS void

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

AS $BODY$

BEGIN

    INSERT INTO rentouts(car_id, user_id, car_price, car_commodities, commodities_price, rent_date,
rent_length, full_rent_price)

    VALUES(carId_, userId_, carPrice_, carCommodities_, commoditiesPrice_, rentDate_, rentLength_,
fullRentPrice_);

    UPDATE rentals

    SET is_rented = true

    WHERE car_id = carId_;

END;

$BODY$;

```

Następną jest `rate_car`; pozwala ona na [przyznanie oceny](#) dla wypożyczonego pojazdu z [poziomu historii wypożyczenia](#), policzenie ilości przyznanych aucie ocen, oraz wyliczenie średniej z tych ocen.

```

CREATE OR REPLACE FUNCTION public.rate_car(

    p_rentout_id integer,

    p_rating real)

RETURNS void

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

AS $BODY$

    DECLARE retrieved_car_id INTEGER;

BEGIN

```

```

UPDATE rentouts
SET rent_rating = p_rating
WHERE renting_id = p_rentout_id;

SELECT car_id
INTO retrieved_car_id
FROM rentouts
WHERE renting_id = p_rentout_id;

IF retrieved_car_id IS NOT NULL THEN
UPDATE rentals
SET car_rating = (
SELECT AVG(rentouts.rent_rating)
FROM rentouts
WHERE rentouts.car_id = retrieved_car_id
),
car_ratings_amount = (
SELECT COUNT(*)
FROM rentouts
WHERE rentouts.car_id = retrieved_car_id
)
WHERE rentals.car_id = retrieved_car_id;
END IF;
END;
$BODY$;

```

Ostatnią z funkcji jest `return_car`, która pozwala na zaznaczenie auta w [historii wypożyczeń](#) jako [zwróconego](#).

```

CREATE OR REPLACE FUNCTION public.return_car(
    p_renting_id integer)
RETURNS void

```

```

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

AS $BODY$

    DECLARE retrieved_car_id INT;

BEGIN

    UPDATE rentouts

    SET is_returned = true

    WHERE renting_id = p_renting_id;


    SELECT car_id INTO retrieved_car_id

    FROM rentouts

    WHERE renting_id = p_renting_id;


    IF retrieved_car_id IS NOT NULL

        THEN

            UPDATE rentals

            SET is_rented = false

            WHERE car_id = retrieved_car_id;

        END IF;

END;

$BODY$;

```

3.3.2. WYZWALACZE

Baza posiada jeden wyzwalacz. Jego celem jest tworzenie logów dla bazy.

```

CREATE OR REPLACE FUNCTION public.log_query()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

```

```

AS $BODY$
DECLARE   query_type TEXT;
BEGIN
IF TG_OP = 'INSERT'
    THEN
        query_type := 'INSERT';
ELSIF TG_OP = 'SELECT'
    THEN
        query_type := 'SELECT';
ELSIF TG_OP = 'UPDATE'
    THEN
        query_type := 'UPDATE';
ELSIF TG_OP = 'DELETE'
    THEN
        query_type := 'DELETE';
ELSE
        query_type := 'OTHER';
END IF;

INSERT INTO logs (log_type, log, log_date)  VALUES (query_type, current_query(), NOW());

RETURN NULL; END;

$BODY$;

ALTER FUNCTION public.log_query()
    OWNER TO ---;

```

(baza danych była pozyskiwana zewnątrznie; oficjalny właściciel został wykreślony w celach bezpieczeństwa)