

Bezpieczeństwo Systemów Komputerowych

**Internetowy Serwis Samochodowy**

Wykonał: Robert Daraż

Cel projektu:

Celem projektu było stworzenie aplikacji internetowej ułatwiającej zarządzanie przyjętymi do naprawy samochodami poprzez prowadzenie dokumentacji online o każdym przyjętym aucie.

Wymagania funkcjonalne:

1.) Logowanie użytkownika.

2.) Wylogowanie użytkownika.

3.) Dodawanie nowych samochodów do systemu.

4.) Dodawanie części samochodowych.

5.) Resetowanie danych formularza.

6.) Usuwanie dodanych części samochodowych.

7.) Obliczanie kwoty brutto/netto wszystkich samochodów w systemie oraz podatku dochodowego.

8.) Usuwanie pojazdów z systemu.

9.) Edycja dodanych wcześniej samochodów w systemie.

10.) Wyświetlanie terminu dostarczenia oraz oddania samochodu wraz z automatycznym obliczaniem czasu serwisowania pojazdu.

Wymagania niefunkcjonalne:

1.) Walidacja pól formularzy przy dodawaniu/edycji samochodów w systemie.

2.) Komunikaty ostrzegawcze przy wprowadzaniu wrażliwych danych oraz po upływie terminu oddania samochodu.

3.) Ochrona przed atakami XSS oraz wstrzykiwaniem SQL.

4.) Walidacja przechodzenia między ekranami systemu.

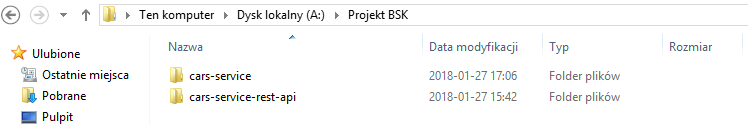
5.) Blokowanie ekranu systemu podczas ładowania danych.

6.) Kompatybilność ze środowiskami Native mobile/desktop, Web worker itp.

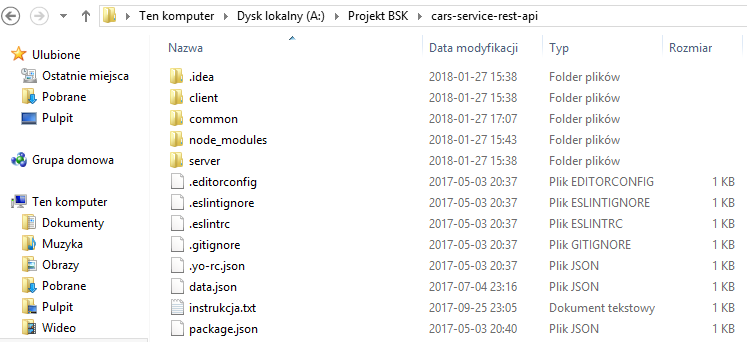
Uruchomienie aplikacji:

Aby uruchomić aplikację na komputerze, należy wykonać następujące kroki:

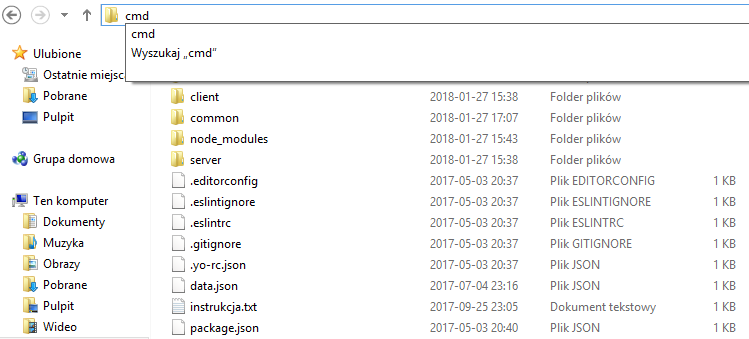
1.) Wypakowanie folderu z projektem „ProjektBSK.zip” na komputerze osobistym który zawiera dwa katalogi o nazwach „cars-service” i „cars-service-rest-api”.

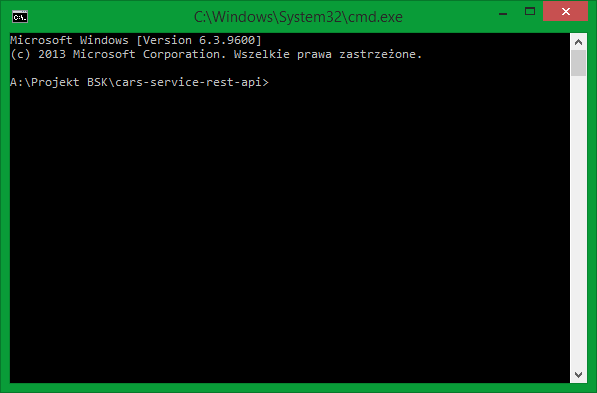


2.) Wejście do folderu z „cars-service-rest-api” w którym znajduje się serwer REST API.

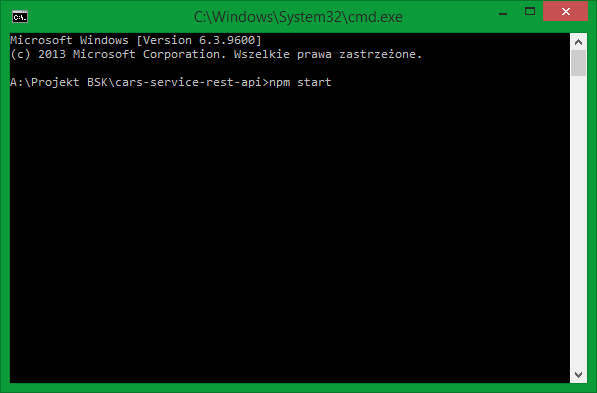


3.) Wejście do konsoli poleceń w folderze „cars-service-rest-api” poprzez wpisanie „cmd” w ścieżce do folderu.

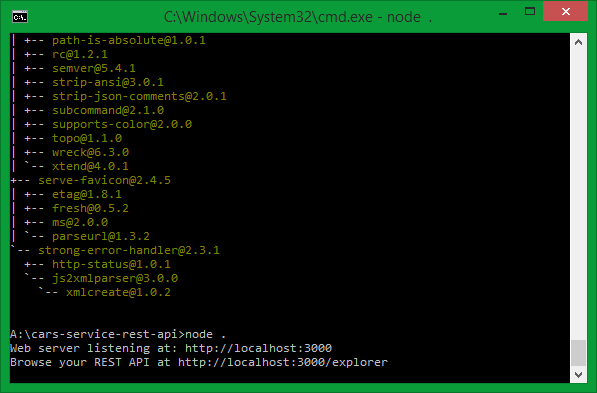




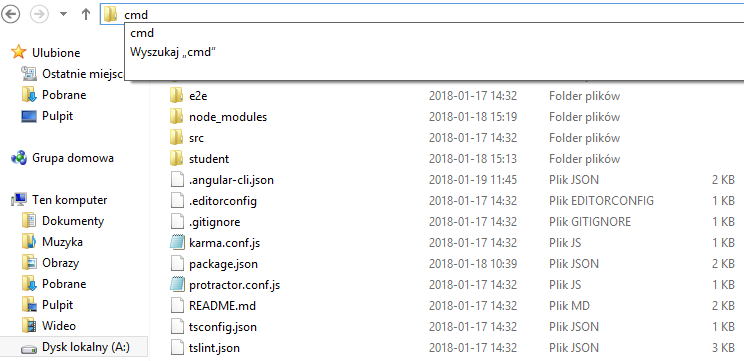
4.) Wpisanie do konsoli instrukcji „*npm start”*. Instalacja npm potrwa do maksymalnie pięciu minut w zależności od prędkości łącza internetowego.



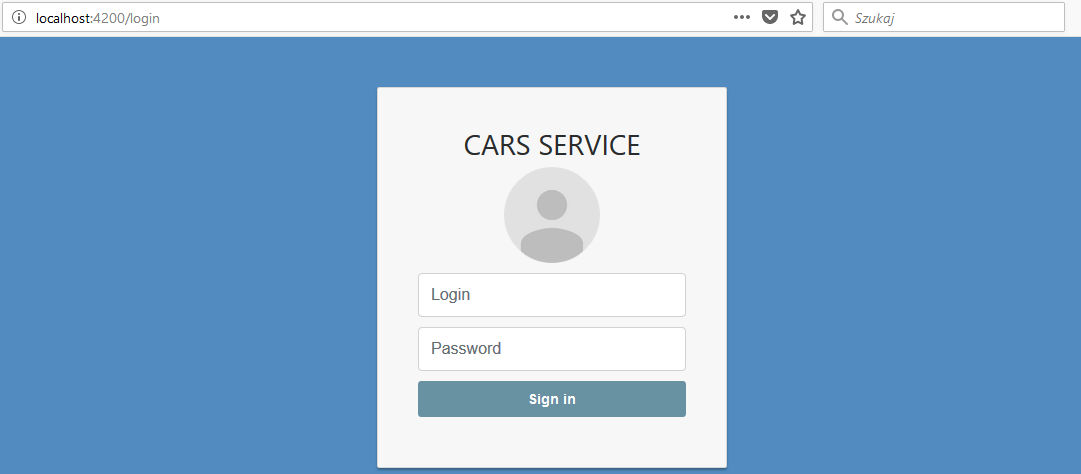
5.) Po poprawnej instalacji *npm*, należy wpisać instrukcję „*node .”* jak poniżej, która zakończy się wyświetleniem odsyłacza do serwera web api.



6.) Po uruchomieniu serwera (należy nie zamykać konsoli z web api), należy przejść do folderu „*cars-service”* i uruchomić konsolę.



7.) Po uruchomieniu konsoli w folderze aplikacji, należy wpisać komendę „*ng serve -o”* która spowoduje, że aplikacja uruchomi się automatycznie w domyślnej przeglądarce internetowej gdzie wyświetli się ekran logowania.



Opis interfejsu aplikacji

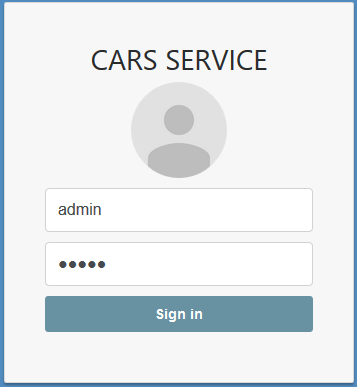
**Logowanie do aplikacji**

Pierwszym ekranem po uruchomieniu aplikacji jest zaimplementowany system logowania. Po wpisaniu nieprawidłowego loginu oraz hasła użytkownik nie otrzyma dostępu do panelu głównego. Aplikacja została rozszerzona o tzw. guard’a dzięki któremu z panelu logowania, nie można przejść do innych ekranów np. po wpisaniu *localhost:4200/cars*  - nastąpi wtedy automatyczne przekierowanie do logowania.

Aby zalogować się poprawnie, należy podać następujące dane:

Login -> „**admin**”

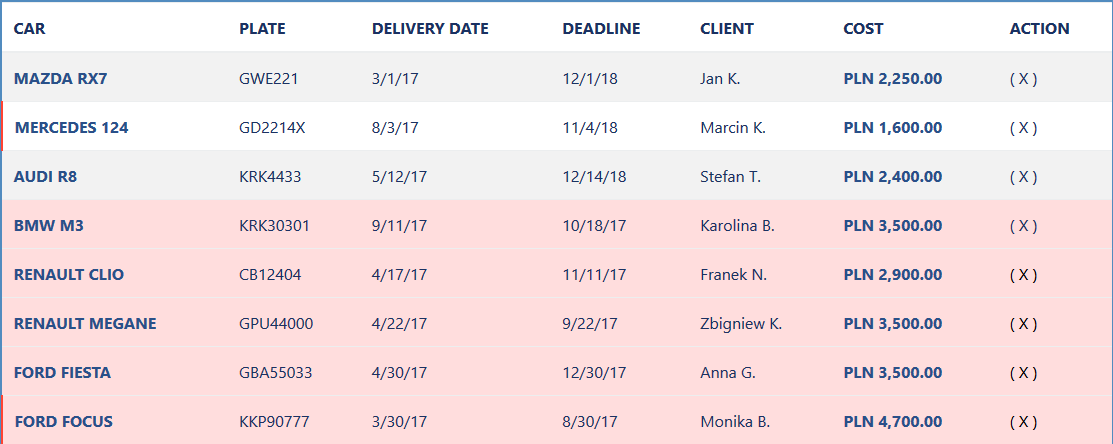
Password -> „**admin**”



Ekran główny aplikacji

Ekran główny jest zbudowany z kilku modułów.

1.) Najważniejszym z nich jest lista samochodów w serwisie:



Moduł ten zawiera listę wszystkich samochodów dodanych do serwisu. Jak widać na wyżej załączonym obrazku, znajdują się tam kolorystyczne walidacje informujące o stanach dla indywidualnych pojazdów.



Czerwony pionowy pasek po lewej stronie rekordu oznacza iż samochód jest całkowicie zepsuty. O tej informacji można się dowiedzieć klikając w dany rekord.



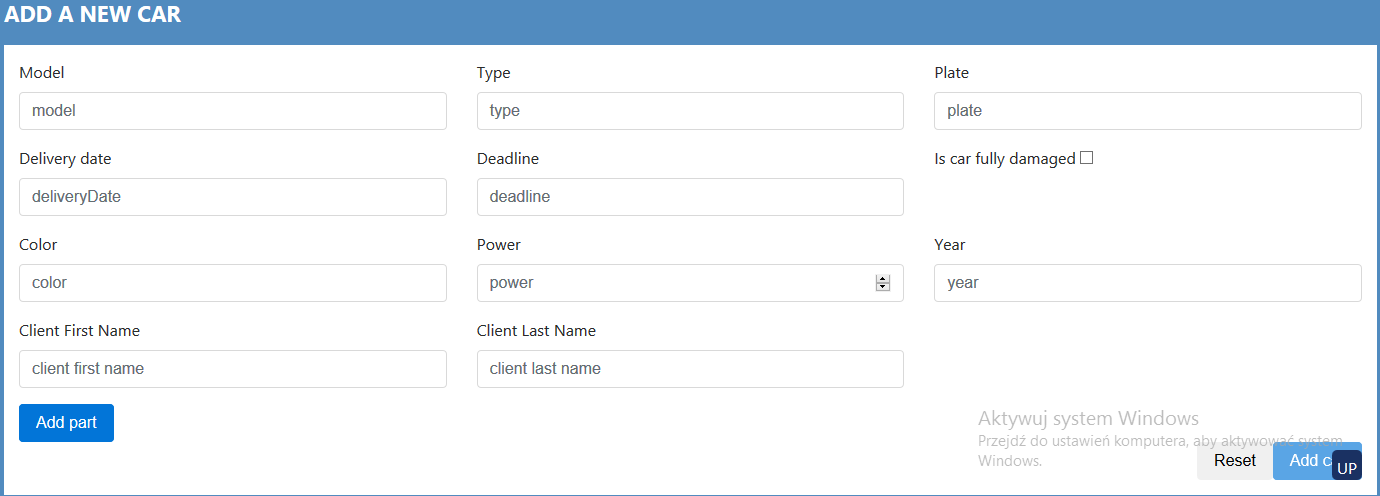
Czerwone tło rekordu oznacza iż samochód jest w serwisie mimo iż minął już termin w którym powinien być oddany klientowi.



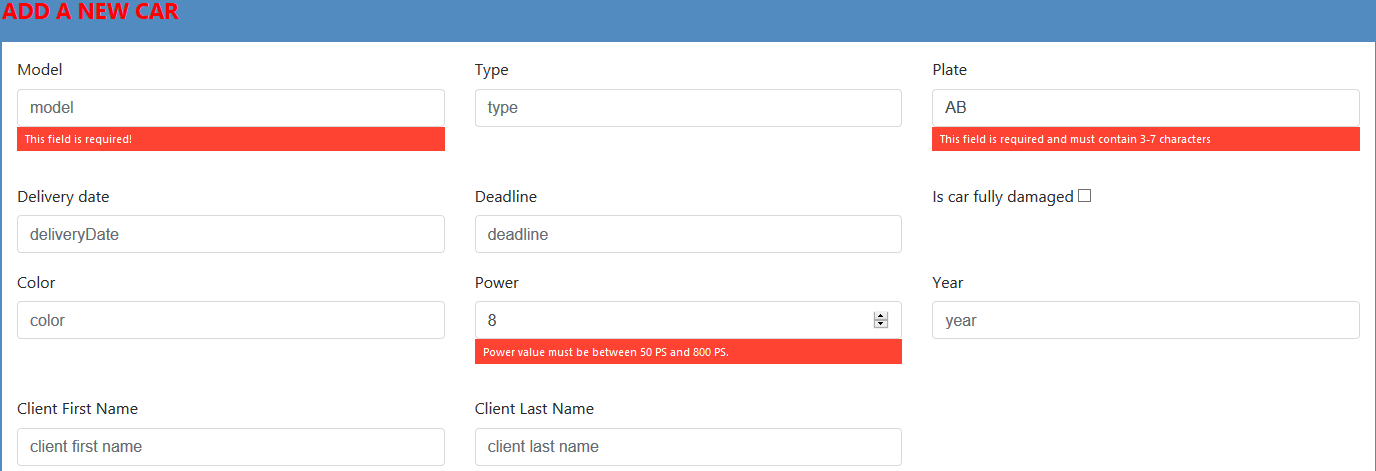
Rekord w obrazku powyżej informuje iż samochód *Ford Focus* jest całkowicie zepsuty oraz jest trzymany w serwisie mimo minięcia daty jego oddania.

Za pomocą przycisku  możemy usuwać auta z systemu.

2.) Poniżej listy samochodów znajduje się moduł który umożliwia dodawanie nowych samochodów.



Moduł ten jest rozszerzony o walidacje danych na formularzu.



Za pomocą przycisku *Add part* można dodawać części które wymagają naprawy w samochodzie wraz z ich kosztem montażu i zakupu.

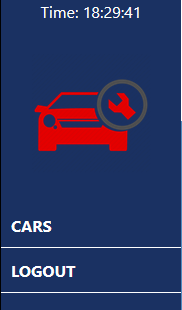


Koszt tych części jest przeliczany w aplikacji na kwotę brutto oraz netto (suma ze wszystkich aut w systemie). Jest także wyliczany podatek dochodowy.



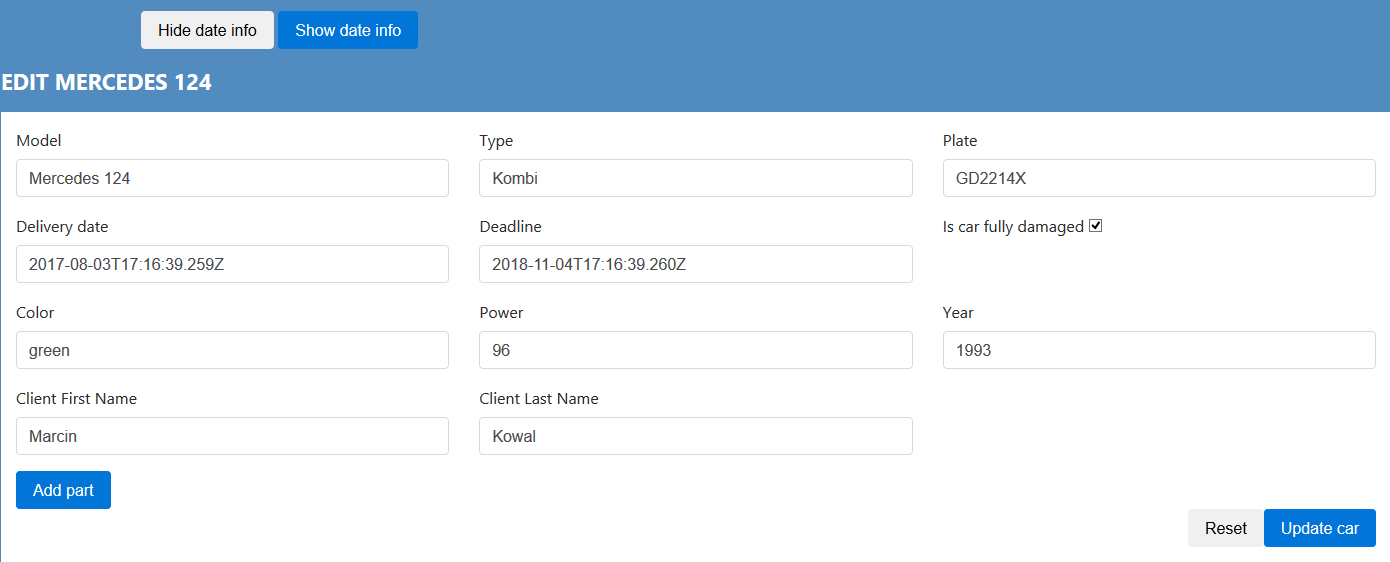
Możemy również usuwać części korzystając z przycisku .

3.) Po lewej stronie panelu głównego widoczny jest pasek nawigacji pokazujący aktualny czas i logo. Poprzez nawigację, możliwe jest wylogowanie z systemu.



Ekran szczegółów pojazdu

Aby wejść w szczegóły pojazdu, należy kliknąć w odpowiedni rekord na ekranie głównym.



Moduł ten umożliwia zmianę wprowadzonych wcześniej danych, a także dodanie/usunięcie lub edycję części do naprawy. Możemy także skorzystać z przycisku „*Reset”* aby wyczyścić cały formularz.

Poprzez przycisk „*Show date info*” wyświetlimy daty dostarczenia samochodu oraz jego odbioru.

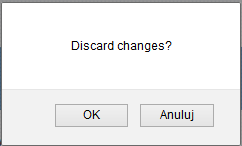


Gdy klikniemy w przycisk „*Check elapsed days”* w nagłówku aplikacji otrzymamy informację o tym jak długo pojazd znajduje się w serwisie.



Walidacje

W systemie zostały zastosowane walidacje dla pól formularza co zostało zaprezentowane przy omawianiu panelu głównego. Dodana została także walidacja przy przechodzeniu na inny ekran w momencie, gdy przerwaliśmy uzupełnianie formularza. Otrzymujemy wtedy komunikat ostrzegawczy o utracie danych.



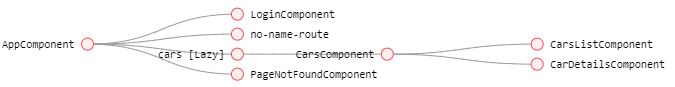
System jest także odporny na wstrzykiwanie SQL oraz ataki XSS poprzez zastosowanie odpowiednich referencji do drzewa DOM które nie odwołują się do jego elementów bezpośrednio zapewniając bezpieczeństwo.

Aby dane wczytały się w poprawnym formacie, został zaimplementowany spinner który blokuje ekran do momentu załadowania odpowiedniego event’u (przejścia między ekranami).

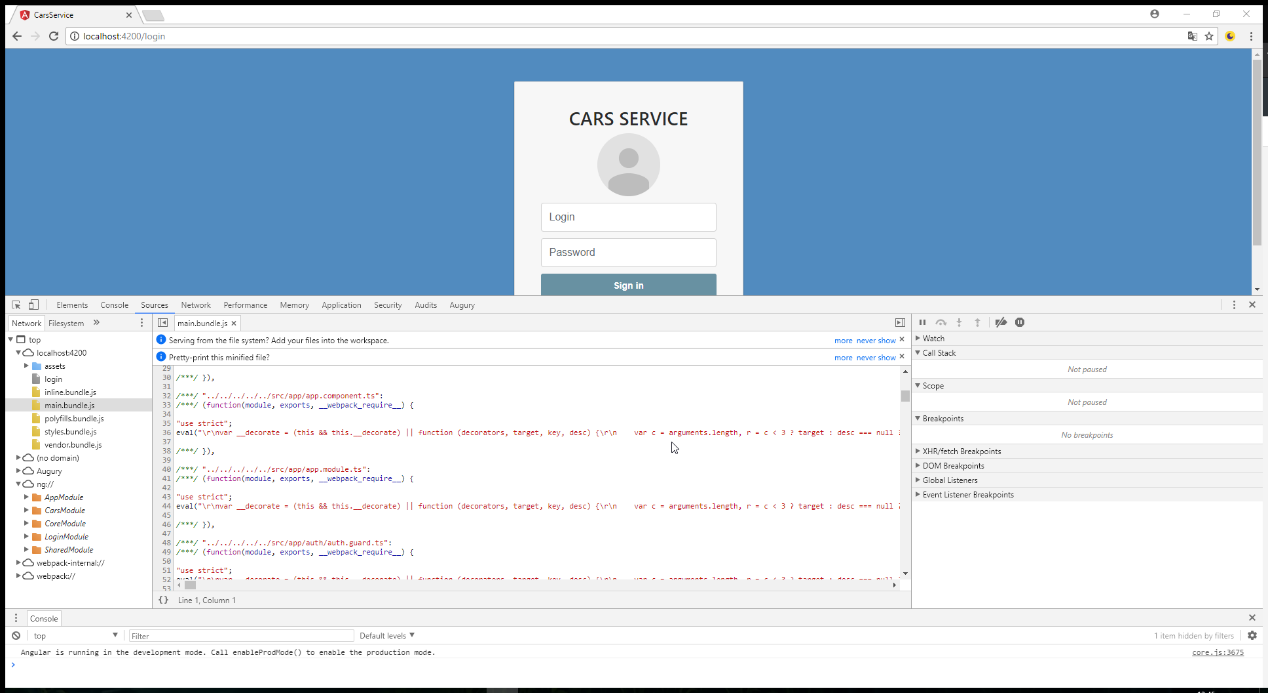


Lazy-loading

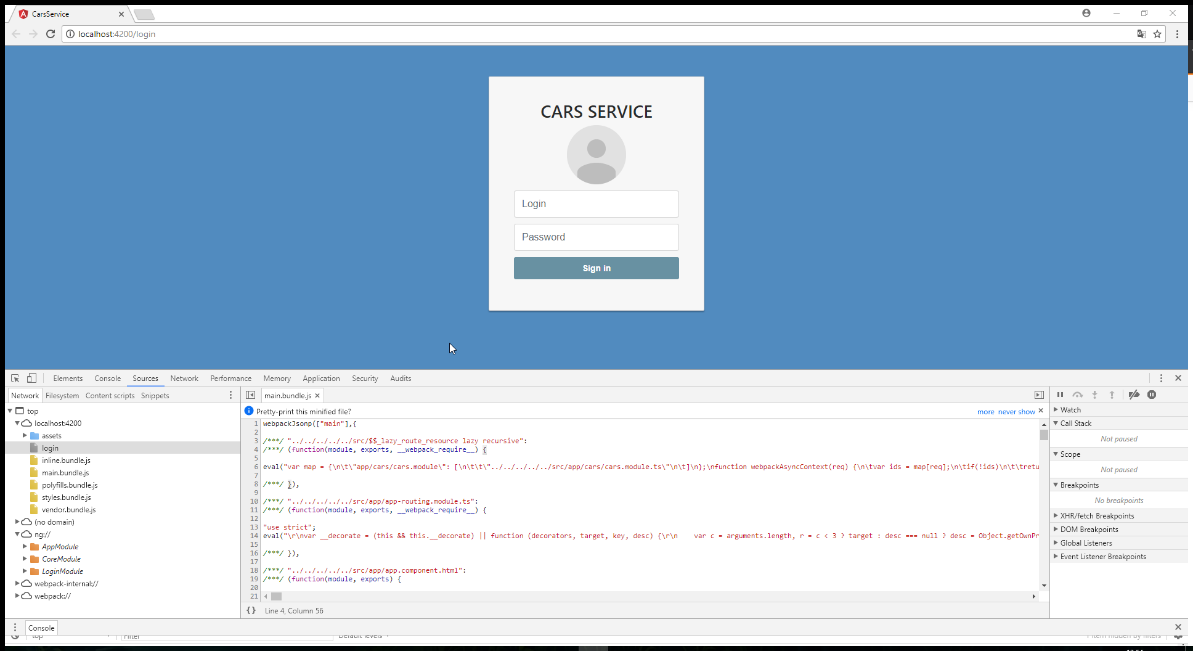
Aby aplikacja działała szybciej przy większej ilości danych, wykorzystałem lazy-loading do ładowania modułów w momencie gdy są one potrzebne. Dzięki rozszerzeniu dla przeglądarki Chrome - Augury można zaobserwować schemat ładowania modułów:



Przed zastosowaniem lazy-loadingu już przy wyświetleniu ekranu logowania, automatycznie ładowane były wszystkie modułu jak poniżej:



Po zastosowaniu lazy-loadingu:



Jak widać na załączonych powyżej obrazkach, przy uruchomieniu aplikacji bez lazy-loadingu, były ładowane wszystkie moduły czyli: *AppModule*, *CarsModule*, *CoreModule*, *LoginModule* oraz *SharedModule*.

Przy zastosowaniu lazy-loadingu ładowane są tylko te których potrzebujemy czyli *AppModule*, *CoreModule* oraz *LoginModule*. Dopiero po zalogowaniu „doładowywane” są pozostałe moduły przez co aplikacja ładuje się zdecydowanie szybciej.

Dodatkowe feature’y

W celu poprawy czytelności zastosowałem w aplikacji tzw. ScrollTop dzięki któremu po zjechaniu na dół strony (powyżej 150px od górnej krawędzi strony), w prawym dolnym roku pojawia się ikona  po kliknięciu której jesteśmy przenoszeni na samą górę strony.

Wykorzystana technologia

Do stworzenia aplikacji wykorzystałem produkt Google’a czyli z dedykowanym command-line oraz framework Angular który jest jednym z najlepszych jeśli chodzi o tworzenie front’end.

Do zaciągania danych wykorzystałem architekturę REST API, a dokładniej serwer loopback-cli umożliwiający komunikację client-server.

Kod aplikacji pisałem w programie Visual Studio Code.