****

**Wydział Informatyki i Matematyki**

**Kierunek: Informatyka**

**Patryk Bąk**

**Nr albumu: 103312**

Projekt konfiguratora budowy samochodu osobowego wykorzystującego bazę danych części samochodowych

Praca inżynierska

napisana pod kierunkiem:

dr inż. Jacek Wołoszyn

Radom 2018

Projekt konfiguratora budowy samochodów osobowych wykorzystującego bazę danych części samochodowych

(streszczenie)

Praca przedstawia projekt oraz implementację aplikacji internetowej „Car-conf”, która pozwala na stworzenie własnej konfiguracji samochodu osobowego z części znajdujących się w bazie danych. Utworzono prostą wersję konfiguratora, będącą przystępnym narzędziem dla niedoświadczonego użytkownika. W pracy zostały opisane najpopularniejsze dostępne w Internecie narzędzia, a także cały proces tworzenia aplikacji oraz jej testowanie.

Słowa kluczowe: samochód osobowy, konfigurator, aplikacja internetowa, baza danych, części samochodowe.

**Project of passenger car building configurator working with car parts database**

**(abstract)**

The thesis presents the project and implementation of the Internet application "Car-conf", which allows you to create your own configuration of a car from parts located in the database. A simple version of the configurator was created, which is an accessible tool for unexperienced user. Thesis describes the most popular tools available on the Internet, as well as the entire process of application development and testing.

Keywords: passenger car, configurator, internet application, database, car parts.

# CEL I ZAKRES PRACY

Celem pracy było stworzenie aplikacji internetowej, która korzysta z bazy danych części samochodowych i pozwala na zbudowanie własnego samochodu. Przybliży ona potencjalnemu użytkownikowi podstawy budowy samochodów osobowych na przykładzie kilku typów nadwozi, pozwoli na poznanie poszczególnych elementów pojazdu wraz ze wskazaniem ich wyglądu i charakterystyki oraz zrozumienie zasady działania samochodu, a także wzbudzi zainteresowanie światem motoryzacji.

Zaproponowane rozwiązanie problemu opiera się na wskazaniu głównych i niezbędnych do działania pojazdu elementów, przedstawieniu ich wizualnie w celu łatwego przyswojenia wiedzy o temacie oraz możliwości aktywnego uczestniczenia w procesie tworzenia samochodu osobowego.

Aplikacja oferuje użytkownikowi możliwość zbudowania własnego pojazdu w prosty sposób, skupiając się tylko na najważniejszych elementach (w celu uproszczenia procesu budowy). Aby ułatwić wybór bądź przybliżyć wygląd danej części, dostępny jest podgląd wybranego elementu, jak również podgląd postępu budowy całego samochodu na obrazie 2D.

W pierwszym rozdziale opisano przegląd aplikacji (dostępnych konfiguratorów samochodów) oraz narzędzi dostępnych na rynku.

W drugim rozdziale przedstawiono oraz uzasadniono wybór konkretnych narzędzi w odniesieniu do założeń aplikacji.

Trzeci rozdział pracy przedstawia podstawy teoretyczne dotyczące zarówno tematyki motoryzacyjnej, jak i informatycznej. Został w nim opisany podział budowy samochodu, jak również podstawy technologii wykorzystanych do stworzenia aplikacji internetowej.

Rozdział czwarty zawiera opis struktury programu, pokazuje funkcje w nim zaimplementowane oraz zastosowane rozwiązania problemów. Wszystkie opisywane przykłady są opatrzone rzeczywistym kodem źródłowym występującym w programie.

Piąty rozdział poświęcono testowaniu aplikacji. Pokazuje on graficznie sposób jej działania, metodę radzenia sobie z ewentualnymi błędami oraz jej funkcje.

W szóstym rozdziale zawarto wnioski wynikające z przeprowadzonej analizy problemu. Zawierają pewne sugestie i możliwości usprawnienia aplikacji.

Spis treści

[CEL I ZAKRES PRACY 4](#_Toc507087959)

[WSTĘP 7](#_Toc507087960)

[1. PRZEGLĄD NARZĘDZI DOSTĘPNYCH NA RYNKU 11](#_Toc507087961)

[1.1. Konfiguratory samochodów 11](#_Toc507087962)

[1.1.1. Mercedes-Benz 11](#_Toc507087963)

[1.1.2. Seat 13](#_Toc507087964)

[1.1.3. Car visualizer 14](#_Toc507087965)

[1.2. Języki programowania 14](#_Toc507087966)

[1.3. Środowiska programistyczne (IDE) 15](#_Toc507087967)

[1.4. Systemy zarządzania bazą danych 16](#_Toc507087968)

[1.5. Programy do edycji grafiki 3D 16](#_Toc507087969)

[1.6. Edycja i tworzenie grafiki 2D 17](#_Toc507087970)

[1.7. Środowiska testowe 17](#_Toc507087971)

[2. KONCEPCJA ORAZ WYKORZYSTANE TECHNOLOGIE I PROGRAMY 18](#_Toc507087972)

[2.1. Założenia projektowe 18](#_Toc507087973)

[2.2. Wybór języków programowania oraz bibliotek 18](#_Toc507087974)

[2.3. Podejście graficzne 19](#_Toc507087975)

[2.4. Narzędzia do testowania aplikacji 19](#_Toc507087976)

[2.5. Uzasadnienie wyboru narzędzi 19](#_Toc507087977)

[3. PODSTAWY TEORETYCZNE 22](#_Toc507087978)

[3.1. Podstawy budowy samochodu 22](#_Toc507087979)

[3.2. Język HTML 22](#_Toc507087980)

[3.3. Kaskadowe arkusze stylów 25](#_Toc507087981)

[3.4. Bootstrap 26](#_Toc507087982)

[3.5. MySQL 27](#_Toc507087983)

[3.6. Język PHP 28](#_Toc507087984)

[3.7. JavaScript 31](#_Toc507087985)

[4. OPIS I TWORZENIE APLIKACJI 35](#_Toc507087986)

[4.1. Przygotowanie środowiska 35](#_Toc507087987)

[4.2. Tworzenie bazy danych 35](#_Toc507087988)

[4.2.1. Struktura 35](#_Toc507087989)

[4.2.2. Konfiguracja 37](#_Toc507087990)

[4.3. Serwer 38](#_Toc507087991)

[4.4. Aplikacja 38](#_Toc507087992)

[4.4.1. Struktura, wygląd i funkcjonalność 38](#_Toc507087993)

[4.4.2. Szkielet aplikacji 39](#_Toc507087994)

[4.4.3. Konfigurator 40](#_Toc507087995)

[4.4.4. Komunikacja z bazą danych 40](#_Toc507087996)

[4.4.5. Implementacja 41](#_Toc507087997)

[5. TESTOWANIE 62](#_Toc507087998)

[6. WNIOSKI 74](#_Toc507087999)

[BIBLIOGRAFIA 75](#_Toc507088000)

[SPIS LISTINGÓW 77](#_Toc507088001)

[SPIS RYSUNKÓW 78](#_Toc507088002)

[SPIS TABEL 79](#_Toc507088003)

# WSTĘP

Rozwój przemysłu samochodowego w ciągu ostatnich lat jest bardzo szybki. Świadczy o tym chociażby ciągle zwiększająca się liczba samochodów na drogach praktycznie we wszystkich krajach na świecie. Można powiedzieć, że w dzisiejszych czasach przeżyć bez posiadania własnego pojazdu jest bardzo ciężko, ponieważ środki komunikacji miejskiej i międzymiastowej głównie kursują w większych miastach. Posiadając własny samochód, można komfortowo przemieszczać się zarówno po ruchliwym mieście, jak i wyjeżdżając na wakacje.

Warto wspomnieć, że dzisiejsze aplikacje ułatwiają podróżowanie samochodem. Wystarczy wymienić chociażby Mapy Google, które są bardzo popularne i oferują mnóstwo funkcji. Pomogą wybrać najszybszą drogę do celu omijając korki, co nie byłoby możliwe, gdyby podróżować autobusem (publiczne środki transportu przeważnie poruszają się stałymi trasami). Dzięki temu można zaoszczędzić nieraz dużo czasu, który dziś jest wręcz na wagę złota.

Dlatego właśnie warto poznać bliżej te popularne „auta”, zajrzeć do ich środka i sprawdzić, co tak naprawdę znajduje się tam, gdzie tego nie widać. Eksploatacja pojazdu to jedna z najważniejszych rzeczy w czasie jego użytkowania. Należy dbać nie tylko o jego czystość, ale także warto kontrolować ilość i jakość płynów, paliwa czy badać zużycie poszczególnych komponentów, bez których nasz samochód nie będzie w stanie poruszać się bezpiecznie (bądź wcale). Niedopilnowanie tych istotnych rzeczy może skończyć się mandatem lub co gorsza wypadkiem drogowym.

Ponieważ motoryzacja to ciekawy temat, a w szczególności samochody osobowe, warto jest przybliżyć ten fascynujący świat potencjalnemu użytkownikowi. W Internecie można znaleźć mnóstwo konfiguratorów od firm zajmujących się produkcją i sprzedażą pojazdów, lecz brakuje tam najczęściej edukacyjnej wartości, a celem jest bardziej marketing i zachęcanie klienta do wybrania konkretnego modelu, niż pokazanie mu, z czego tak naprawdę jest zbudowany.

Historia samochodu zaczęła się w 1886 roku, kiedy „Carl Benz opatentował pojazd napędzany silnikiem spalinowym” [[41](#Tes03)]. Jako syn maszynisty kolejowego, z pewnością nie zdawał sobie sprawy, że „zapoczątkuje on produkcję miliardów samochodów na całym świecie”. Ten pierwszy pojazd przypominał bardziej konną bryczkę, niż rzeczywisty samochód. Posiadał on trzy koła, w dodatku każde z nich miało druciane szprychy i opony z twardej gumy. Jednostką napędową tego pojazdu był jednocylindrowy silnik spalinowy chłodzony wodą, osiągający moc zaledwie 0,88 konia mechanicznego. Napędzał on tylną oś (wówczas sztywną, czyli koła były połączone ze sobą jedną prostą osią). Pojazd osiągał prędkość maksymalną 16 km/h. Dzisiaj najprostszy samochód posiada moc co najmniej 40-50 km, a prędkości osiągane przez te najszybsze potrafią przekroczyć nawet 400 km/h. Sekretnym rozwiązaniem opracowanym przez Benza było umieszczone z tyłu wielkie koło zamachowe, które umożliwiało mu wprawienie w ruch silnika podczas jazd próbnych. Jak nietrudno się domyśleć, jego nazwisko zostało później częścią nazwy marki Mercedes-Benz, czyli jednej z najbardziej znanych dziś firm produkujących między innymi samochody sportowe, luksusowe i ciężarowe [[1](#Wik)].

W XIX wieku praktycznie wszystkie pojazdy oparte były na konstrukcji z silnikiem parowym (takim jak w lokomotywach), zatem wynalazek Benza był czymś niewątpliwie wyprzedzającym swoje czasy. Ciekawą innowacją było także wprowadzenie przez Nikolausa Augusta Otto spalinowego silnika czterosuwowego [[42](#Ver03)]. Pozwoliło to na bardziej wydajne wykorzystanie spalanego paliwa i zwiększenie płynności pracy silnika. Dziś wiadomo, że każdy silnik spalinowy zastosowany w nowym samochodzie posiada silnik czterosuwowy. Stąd wniosek, że był to bardzo ważny wynalazek. Oprócz silników spalinowych, współcześnie stosowane są także silniki wysokoprężne (Diesla), elektryczne oraz napędy hybrydowe (łączące kilka różnych źródeł napędu).

Twórcy pierwszych samochodów skupiali się na czysto funkcjonalnych aspektach swoich pojazdów, pomijając kwestie komfortu czy bezpieczeństwa ich użytkowników. Wraz z ewolucją tych rewolucyjnych maszyn, pojawiła się karoseria. Najprościej mówiąc, jest to zewnętrzna powłoka auta wykonana z blachy lub tworzywa sztucznego [[43](#Ver031)]. Można powiedzieć, że po części warunkuje ona to, jakie przeznaczenie będzie miał pojazd. Sportowe samochody będą miały bardziej opływową karoserię, o małym współczynniku oporu powietrza i z licznymi wlotami powietrza, które poprawiają jego aerodynamikę. Z kolei pojazdy przeznaczone do jazdy w trudniejszych warunkach, jak na przykład SUV-y czy pojazdy terenowe posiadają większą, bardziej masywną karoserię. W tego typu samochodach nie liczy się aerodynamika ani masa, lecz wytrzymałość konstrukcji oraz niezawodność [[39](#Pot10)]. Ewolucja pozwala cieszyć się wieloma rozmaitymi rodzajami nadwozi, jak na przykład nadwozia z zamkniętym dachem (coupé, sedan, limuzyna, hatchback, SUV, liftback, fastback, itp.), z otwartym dachem (kabriolet, targa, roadster), czy stworzone do zadań specjalnych (do przewozu osób czy większych ładunków, np. mikrovan, minivan, van, pick-up) [[33](#Blo16)].

Skąd zatem tyle różnych rodzajów nadwozi? Każde z nich ma konkretne przeznaczenie. Przykładowo, jeden z najczęściej występujących typów karoserii, czyli sedan, stworzony został z myślą o transporcie od jednej do maksymalnie pięciu osób wraz z niewielkim ładunkiem (bagażnik takiego auta będzie miał pojemność około 500 litrów). Autem o takim nadwoziu ciężko jest poruszać się po górzystych terenach, zwłaszcza tam, gdzie występuje mocno pofałdowany czy grząski teren. Do takich zadań stworzone są auta terenowe bądź SUV-y, czyli auta sportowo-użytkowe, które posiadają wyższe zawieszenie oraz najczęściej napęd na wszystkie cztery koła. Ewolucją sedana jest limuzyna, czyli nadwozie charakteryzujące się sporą długością, dzięki czemu oferuje więcej miejsca pasażerom wewnątrz pojazdu. Posiada też najczęściej różne gadżety poprawiające komfort jazdy i użytkowania pojazdu, jak na przykład elektrycznie sterowane zawieszenie czy kamera cofania.

Z kolei coupé, czyli typowo sportowe nadwozie, zostało stworzone dla entuzjastów motoryzacji i sportowych wrażeń. Samochód o takim nadwoziu będzie charakteryzował się dużą aerodynamiką, rozmiarem bardziej kompaktowym niż sedan, dzięki czemu oferuje najczęściej znacznie lepsze osiągi. Posiada dwoje drzwi, dlatego w 4-miejscowych coupé pasażerowie siadający z tyłu będą czuli się znacznie mniej komfortowo, niż w nadwoziu typu sedan. Bardzo często z powodu oszczędności masy i specyficznego kształtu, taki samochód będzie miał zdecydowanie mniejszy bagażnik. Dlatego auto z tym typem karoserii jest przeznaczone do szybkiej jazdy oraz do transportu nie więcej niż maksymalnie czterech osób (często tylko dwóch) z bardzo skromnym bagażem. Nadwozie to jest przede wszystkim wykorzystywane w wyścigach samochodowych, takich jak zawody DTM, czy FIA klasy GT1 i GT3, oraz stosowane w samochodach sportowych.

Popularnym typem nadwozia jest również hatchback, które jest wykorzystywane między innymi do produkcji aut miejskich. Są to małe pojazdy, stworzone z myślą o kierowcach, którzy cenią sobie małe rozmiary i wysoką ekonomię transportu. Taki samochód będzie lekki, zwrotny, a także będzie miał kompaktowe rozmiary, które ułatwiają zaparkowanie go na miejskim parkingu. Nadwozie to jest bardzo zbliżone do nadwozia typu sedan, ale posiada ścięty tył, co zmniejsza długość pojazdu. Niestety, zabieg ten zmniejsza również pojemność bagażnika, ale za to we wnętrzu auta pozostaje tyle samo miejsca. Wiele modeli samochodów posiada kilka wersji nadwozia do wyboru, w tym również coupé, sedan oraz hatchback. Z czasem powstał też pośredni rodzaj karoserii, czyli liftback. Jest to tak zwany sedan z lekko ściętym tyłem. Trudno opisać, w jakim celu został zastosowany taki zabieg. Najprawdopodobniej była to kwestia czysto stylistyczna i nie miała zbyt wiele wspólnego z funkcjonalnością auta.

Warto wspomnieć także o często spotykanym typie nadwozia, zwanym SUV. Pojazdy o takim typie karoserii pojawiają się coraz częściej na polskich drogach i nie jest to kwestia przypadku. Łączą one zalety nadwozia typu sedan z czysto terenowym pojazdem, takim jak na przykład Mitsubishi Pajero Evolution. W zasadzie każdy pojazd o tym nadwoziu posiada napęd AWD (czyli napęd na wszystkie koła z możliwością ustawienia różnego stosunku mocy przenoszonej na przednią i tylną oś), dzięki czemu bardzo dobrze radzą sobie w terenie - jest to niezastąpiona zaleta podczas rodzinnego wyjazdu w wysokie góry. Posiadają także dużo miejsca dla pasażerów oraz pojemny bagażnik. Ekonomiczne silniki czynią te auta bardzo atrakcyjnym wyborem dla każdego kierowcy. W ostatnich latach bardzo mocno rozwinęła się idea napędów hybrydowych [[44](#Pra10)], stosowanych między innymi przez japońskie firmy, takie jak Toyota, Mazda czy Lexus. Pozwala to na stosowanie silników o coraz mniejszej pojemności, a więc coraz bardziej ekonomicznych.

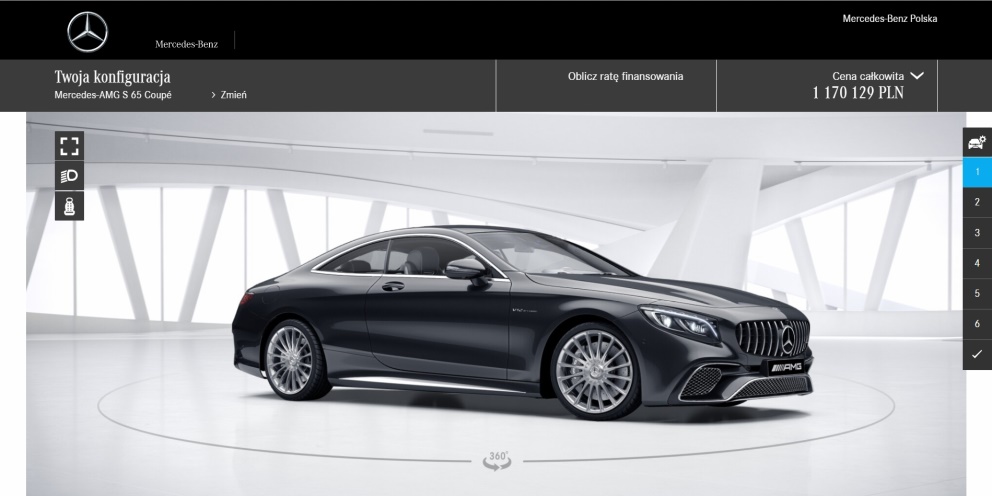
# PRZEGLĄD NARZĘDZI DOSTĘPNYCH NA RYNKU

Obecnie firmy produkujące samochody zazwyczaj mają własne witryny, na których dostępne są informacje o wszystkich modelach wytwarzanych przez odpowiednią markę. Na stronach tych największych producentów są dostępne również aplikacje internetowe, zwane konfiguratorami. Umożliwiają one odwiedzającemu stronę utworzenie własnej konfiguracji wybranego modelu samochodu, wraz z podaniem jego cech, danych technicznych oraz dostępnych opcji wyposażenia i gadżetów.

## Konfiguratory samochodów

### Mercedes-Benz

Przykładem takiej aplikacji może być konfigurator samochodów firmy Mercedes-Benz [[2](#Mer)]. Program ten umożliwia wybranie modelu z obecnej oferty (pozioma lista przewijana z obrazkami modelów), a następnie oferuje wybór wersji silnikowej spośród dostępnych wersji. Po wybraniu silnika, konfigurator wyświetla trójwymiarowy podgląd samochodu [] wraz z opcjami dla danego modelu.



Rys. 1: Konfigurator firmy Mercedes-Benz: wybrany model widnieje na dużym podglądzie, pod dostępnym widokiem opcje wyboru wyposażenia. Po lewej stronie opcje podglądu, po prawej lista wyboru kroków z podświetleniem obecnego [źródło: konfigurator samochodów Mercedes-Benz].

Pojazd na podglądzie jest modelem 3D powstałym prawdopodobnie poprzez wykonanie zdjęć w salonie w odstępie około 10 stopni między każdym ujęciem dookoła samochodu, stąd brak płynności przy obrocie modelem o 360 stopni. Do implementacji tego podglądu użyto głównie metody *Canvas* (wyświetlanie obrazków na stronie) oraz biblioteki WebGL (umożliwia ona stworzenie podglądu ze swobodnym obracaniem modelem). Oba te narzędzia są sterowane poprzez skrypty napisane w języku JavaScript. Opcje wyposażenia i dodatki są wyświetlane w postaci kafelków pod podglądem i posiadają w rogu mały kwadrat. Po jego kliknięciu zostaje wybrana dana opcja i część strony ładuje się od nowa bądź zanim to nastąpi wyświetla się dodatkowe okno umożliwiające wybór bardziej szczegółowych ustawień. Z boku strony wyświetlany jest pasek nawigacji pokazujący wszystkie kroki konfiguratora i umożliwia on przejście do dowolnej części konfiguracji. Podgląd zmienia się na każdym kroku, pokazując obecnie edytowane elementy samochodu (nadwozie, wnętrze, koła itp.).

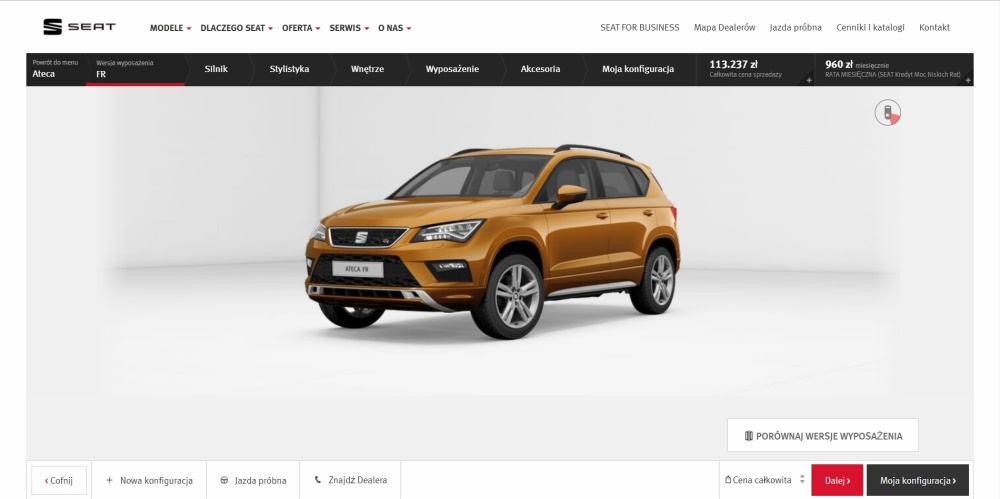
Za wygląd i układ elementów na stronie odpowiadają języki HTML oraz CSS. Użyto także bibliotek jQuery, AngularJS oraz JSON (ang. *JavaScript Object Notation*). Wygląd strony został prawdopodobnie zdefiniowany poprzez pliki CSS oraz JavaScript stworzone przez programistów na potrzeby witryny.

Pierwsza z wymienionych bibliotek, czyli jQuery, odpowiada za uproszczenie składni języka JavaScript i zwiększa jego możliwości. Natomiast AngularJS to framework, dzięki któremu można zrobić aplikację internetową działającą bez potrzeby odświeżania strony (ang. *Single Page Application*). JSON jest to lekki format wymiany danych komputerowych. JSON jest formatem tekstowym, bazującym na podzbiorze języka JavaScript [[3](#Wik16)]. Służy on m. in. do przekazywania danych do aplikacji opartych o AJAX (ang. *Asynchronous JavaScript And XML*). Z kolei AJAX jest to „technika tworzenia aplikacji internetowych, w której interakcja użytkownika z serwerem odbywa się bez przeładowywania całego dokumentu, w sposób asynchroniczny” [[4](#Wik17)]. Dzięki temu aplikacja nie zużywa dużych zasobów do zmiany zawartości strony, ponieważ przy wyborze opcji nie ładuje się cała strona, a tylko jej część.

Konfigurator sprawia wrażenie dość złożonej aplikacji, dużo jest elementów na stronie, natomiast układ wygląda na estetyczny i dopracowany. Optymalizacja działania jest na dobrym poziomie, nie zużywa zbyt dużej ilości zasobów urządzenia. Aplikacja działa także na urządzeniach mobilnych, takich jak tablety czy smartfony.

### Seat

Konfigurator marki Seat [[5](#Sea)] wydaje się nieco prostszy w konstrukcji. Elementy wyboru na stronie są nieruchome, a podgląd złożony jest z 24 obrazków, które wyświetlają się w miejscu podglądu w zależności od aktualnego przesunięcia obiektu [rys. 2]. Wszystkie te obrazki są plikami umieszczonymi na stronie render.seat-auto.pl w  formacie .jpeg.



Rys. 2: Konfigurator Seata. Po wyborze modelu wyświetla się podgląd samochodu z opcją wyboru wersji wyposażenia. Lista kroków dostępna na pasku nawigacji nad podglądem, poniżej widoku dostępne opcje. Na dole widoczny pasek oferujący dodatkowe funkcje, jest umieszczony zawsze w tym samym miejscu [źródło: konfigurator samochodów Seat].

Narzędzia użyte do wykonania powyższej aplikacji to głównie JavaScript oraz najprawdopodobniej PHP (w adresie strony obecny pytajnik oraz dowolny ciąg znaków). Konfigurator korzysta też z języka CSS, biblioteki jQuery oraz metod *Canvas* i *SVG* do wyświetlania grafiki. PHP służy do łączności programu z bazą danych SQL oraz przesyłania informacji między serwerem i bazą danych, a stroną.

### Car visualizer

Ciekawym konfiguratorem samochodów jest *Car visualizer* [[6](#car)]. Stworzona przez niezależne studio *+360 degree* aplikacja wyświetla w czasie rzeczywistym trójwymiarowe modele aut [rys. 3]. Jest ona oparta w głównej mierze na języku JavaScript i korzysta z biblioteki *three.js* [[7](#thr)], bazującej na wspomnianej już bibliotece WebGL.



Rys. 3: Konfigurator Car visualizer. Po wejściu na stronę konfiguratora i załadowaniu elementów wyświetla się trójwymiarowy model samochodu ze zmieniającym się widokiem kamery oraz zostaje odtworzona muzyka. W lewym górnym rogu widnieją opcje wyświetlania (powiększenie, widok z kamery, przejście do strony głównej). Na dole opcje wyboru modelu, kolor nadwozia i obręczy oraz opcje audio [źródło: konfigurator Car visualizer].

## Języki programowania

W opisanych wcześniej przykładach konfiguratorów głównym językiem użytym do stworzenia takiej aplikacji jest JavaScript, a korzysta on z dodatkowych bibliotek, takich jak jQuery, WebGL czy AngularJS dla zwiększenia funkcjonalności. Pojawił się również język PHP, który umożliwia nawiązanie połączenia z bazą danych SQL oraz przesyłanie i odbiór danych bez stosowania dodatkowych bibliotek czy narzędzi.

Mniej popularnym narzędziem, ale o równie dużych możliwościach, może być również zastosowanie języka Python oraz frameworka Django. Umożliwia to zrobienie dynamicznego serwisu internetowego, a także połączenia się z bazą danych (domyślnie jest to SQLite). Odbywa się to poprzez zdefiniowanie modeli oraz widoków dla bazy w odpowiednim pliku.

Niektóre aplikacje internetowe są programowane również przy pomocy narzędzia zwanego ASP.NET. Jest to „zbiór technologii opartych na frameworku zaprojektowanym przez firmę Microsoft” [[8](#htt)]. Pozwala on na tworzenie programów poprzez generowanie kodu HTML, CSS czy XML.

## Środowiska programistyczne (IDE)

Na rynku dostępnych jest wiele aplikacji i środowisk oferujących możliwość pisania kodu programu w dowolnym języku. Znając składnię języka i jego cechy oraz możliwości, można napisać kod źródłowy aplikacji nawet w prostym notatniku, który posiada każdy dzisiejszy system operacyjny. Niestety, minusem takiego narzędzia jest fakt, iż nie pomaga ono zbytnio programiście w pracy, jest mało intuicyjne i nie pozwala sprawdzić ewentualnych błędów w zapisie.

Jednym z prostszych i darmowych narzędzi do pisania kodu programu w języku JavaScript, PHP czy HTML może być Notepad++. Jest to rozbudowany notatnik, który oferuje podkreślenie składni oraz podpowiedzi funkcji w wybranym języku (może być to PHP, SQL, JavaScript czy HTML). Posiada także wbudowaną funkcję automatycznego zamykania apostrofów i znaczników, co ułatwia pracę z kodem. Minusem jest fakt, że nie potrafi sprawdzić programu pod względem błędów, zatem należy użyć do tego celu własnej przeglądarki internetowej.

Innym rozwiązaniem, choć płatnym, jest pakiet programów firmy JetBrains. Firma ta umożliwia skorzystanie z własnych środowisk programistycznych dedykowanych do konkretnych zastosowań. Programy te są bezpłatne dla studentów, zatem jest to dobra propozycja do nauki programowania, ponieważ oferuje ona wszelkie potrzebne pomoce dla początkującego programisty (na przykład podświetlenie składni, automatyczne wcięcia, kompilator w czasie rzeczywistym). Z interesujących aplikacji warto wymienić WebStorm, PhpStorm, czy PyCharm. Możliwości każdego z tych programów można rozszerzyć poprzez dodanie nowych bibliotek.

Pierwsza z nich jest dedykowana do programowania w językach HTML, CSS oraz JavaScript. Istnieje możliwość dodania dowolnej biblioteki bądź frameworka. Najpopularniejsze to jQuery, AngularJS czy Bootstrap.

Druga aplikacja służy do tworzenia programów głównie w języku PHP oraz łączenia się z serwerem (lokalnym bądź publicznym), a także z bazą danych. Oferuje także wsparcie dla innych języków, m. in. HTML, CSS, JavaScript.

PyCharm to środowisko programistyczne dla osób chcących pisać aplikacje w języku Python. Oferuje wiele narzędzi i szeroki zakres możliwości, który można również rozszerzyć poprzez dodatkowe biblioteki. Po dodaniu frameworka, na przykład Django, programista może wykonać serwis internetowy wraz z bazą danych w języku SQL. To narzędzie pozwala na połączenie tzw. *frontendu* i *backendu*, czyli strony wizualnej i funkcjonalnej aplikacji internetowej.

## Systemy zarządzania bazą danych

Jednym z zastosowanych narzędzi w powyższych programach jest baza danych oparta na języku SQL. Baza danych służy do przechowywania informacji oraz opisuje relacje pomiędzy danymi. Najpopularniejsze systemy baz danych oparte na tym języku to MySQL, Microsoft SQL Server oraz SQLite. Istnieją również bazy No-SQL, takie jak MongoDB, które nie są oparte na składni języka SQL, a także aplikacje, np. Microsoft Access.

Pierwszy system jest często używany z językiem PHP, ponieważ jest zawarty także w popularnym zintegrowanym pakiecie XAMPP [[9](#xampw)] dla wszystkich systemów (ang. *cross-platform Apache MySQL PHP Perl*) bądź LAMP, wyłącznie dla dystrybucji Linux. W takim pakiecie zawarte są wszystkie narzędzia potrzebne do tworzenia bazy danych, a także komunikacji z serwerem. Przy pomocy przeglądarki internetowej można dowolnie edytować taką bazę, jak również eksportować stworzoną do pliku SQL bądź importować dowolną inną bazę danych.

Microsoft SQL Server, który „jest platformą bazodanową typu klient-serwer” [[10](#Wik5)], jest przeznaczony na systemy Windows oraz Linux. Korzysta z technologii ASP.NET i jest w większości przeznaczony do komercyjnych zastosowań.

## Programy do edycji grafiki 3D

Bardziej złożone i atrakcyjne wizualnie aplikacje, niepowiązane z komercyjnymi zastosowaniami, jak na przykład wymieniony wyżej konfigurator „Car visualizer”, wymagają obiektów w postaci modeli 3D. Ze znanych programów do edycji grafiki trójwymiarowej warto wymienić komercyjny program 3DS Max oraz Blender, czyli jego darmowy odpowiednik. Niektóre programy, jak na przykład Adobe Photoshop, oferują również możliwość edycji pliku w formacie .obj, ale jest to trudne z uwagi na dużą złożoność aplikacji, a także niezbyt dobrą optymalizację w przypadku dużej ilości elementów.

## Edycja i tworzenie grafiki 2D

Warto tutaj rozróżnić dwie główne gałęzie grafiki - grafikę wektorową oraz grafikę rastrową. Pierwsza z nich najczęściej odpowiada za tworzenie ikon, elementów interfejsu oraz ich estetyczny układ na stronie, a także na przykład figur geometrycznych. Programy do grafiki wektorowej to na przykład: Adobe Illustrator, CorelDRAW, Inkscape. Natomiast druga służy do edycji obrazków dwuwymiarowych, kolorów, stosowania efektów, filtrów i innych narzędzi. Programy do grafiki rastrowej to np. Adobe Photoshop, Corel Photo-Paint, GIMP.

## Środowiska testowe

W Internecie dostępnych jest kilka różnych narzędzi, które wspomagają testowanie. Najbardziej znane są środowiska programistyczne bądź w przypadku stron lub serwisów internetowych także przeglądarki internetowe. Do tych pierwszych należy zaliczyć pakiet programów firmy JetBrains (wbudowany kompilator), natomiast najpopularniejsze przeglądarki to Google Chrome, Mozilla Firefox oraz Opera.

Aplikacje zwane IDE (ang. *Integrated Development Environment*) mają często wbudowany kompilator, który pozwala na sprawdzenie błędów w tworzonym programie już na etapie pisania kodu. Nieraz takie środowiska analizują kod źródłowy w czasie rzeczywistym, informując programistę na bieżąco o możliwych błędach w zapisie bądź nieprawidłowych konstrukcjach. Do takich należą wszystkie aplikacje firmy JetBrains, a także np. firmy Microsoft. Umożliwiają one często nawiązanie połączenia z serwerem (nawet lokalnym), a więc i łączność z bazą danych.

Przeglądarki internetowe pozwalają na sprawdzenie napisanej strony internetowej tak, jak byłaby w rzeczywistości stroną na dowolnym serwerze. Plusem jest fakt, że przeglądarka otworzy każdą stronę, która wykorzystuje kod HTML, CSS czy JavaScript. Niestety, nie wyświetli poprawnie strony napisanej w języku PHP, jeśli nie ma ona zdefiniowanego serwera. Dotyczy to również zawartości, która pojawia się na stronie po odwołaniu się do pliku PHP, ponieważ język ten wymaga połączenia z serwerem (przeglądarka nie widzi kodu źródłowego, tylko wynik działania tego kodu). Jednakże jest to niezbędne narzędzie dla każdego programisty *frontendowego* czy *backendowego*.

# KONCEPCJA ORAZ WYKORZYSTANE TECHNOLOGIE I PROGRAMY

## Założenia projektowe

Tworzona aplikacja jest, zgodnie z tematem pracy, konfiguratorem budowy samochodów, wykorzystującym bazę danych części samochodowych. Oznacza to, że umożliwia ona zbudowanie pojazdu poprzez łączenie wybranych przez użytkownika elementów auta z bazy danych. Przyjęto, że jest to aplikacja internetowa, ze względu na dużą popularność stron i serwisów WWW. Za nazwę przyjęto „Car-conf”.

Program powinien być maksymalnie prosty w obsłudze i nieskomplikowany tak, aby każdy użytkownik, bez względu na wiedzę w zakresie motoryzacji czy informatyki, był w stanie stworzyć w nim swój samochód. Ponadto, aby uczynić go bardziej przyjaznym dla osób nieobeznanych ze światem motoryzacji, powinien posiadać podgląd części oraz ilustrować, który krok jest aktualnie realizowany. Dzięki temu aplikacja stanie się bardziej atrakcyjna wizualnie.

Dodatkowo, program powinien być napisany zgodnie z najnowszymi standardami w danym języku, czyli zawierać obsługę błędów oraz zużywać jak najmniej zasobów sprzętu, na którym program będzie uruchomiony. Musi także zawierać obsługę wyjątków oraz komunikaty o błędach i powiadomienia, informujące użytkownika o nieprawidłowym działaniu z jego strony bądź żądające jego potwierdzenia do wykonania akcji.

## Wybór języków programowania oraz bibliotek

Wykonana aplikacja internetowa została oparta na językach HTML5, CSS3 oraz PHP. Do zapewnienia logicznego układu elementów wykorzystuje ona popularny framework Bootstrap [[11](#Boo)]w wersji 3.3.7, wraz z bibliotekami jQuery oraz Popper.js, niezbędnymi do jego działania. Ponieważ aplikacja ta do pracy potrzebuje bazy danych, do tego celu stworzona została baza danych oparta na systemie MySQL w wersji 4.7.1, pracująca na lokalnym serwerze Apache zapewnionym przez aplikację XAMPP [[12](#XAM14)]. Do połączenia aplikacji z bazą danych wykorzystano język PHP w wersji 7.1.1, JavaScript oraz technologię AJAX. Aplikacja została zaprogramowana w środowisku IDE firmy JetBrains, w programie PhpStorm 2017.3 [[13](#Jet17)].

## Podejście graficzne

Do wykonania elementów graficznych (ikony) na stronie został użyty program Adobe Illustrator CC 2017 [[14](#Ado171)]. Szablon strony został pobrany z witryny W3Schools [[15](#W3S)]. Obrazki powstały poprzez wykonanie zrzutów ekranu w programie Blender [[16](#Ble)]w wersji 2.78, oraz ich edycji w programie Adobe Photoshop CC 2017 [[17](#Ado17)]. Obrazki części oraz pojazdów zostały pobrane z Grafiki Google [[18](#gra)]. Natomiast do graficznego układu pliku .pdf z utworzonej konfiguracji wybrano program Stimulsoft Designer [[19](#sti)].

## Narzędzia do testowania aplikacji

Do wykonania testów aplikacji użyto przeglądarki Google Chrome [[20](#Goo17)]w wersji 63.0.3239.84. Pozwala ona na uruchomienie trybu programisty, dzięki któremu można analizować błędy generowane przez kod HTML, CSS oraz JavaScript. Oferuje także możliwość sprawdzenia responsywności aplikacji poprzez zastosowanie zmiennej rozdzielczości okna, w którym otwarta jest strona. Bardzo pomocna okazała się edycja kodu w programie PhpStorm, dzięki debugerowi sprawdzającemu błędy w kodzie aplikacji w czasie rzeczywistym.

## Uzasadnienie wyboru narzędzi

Aplikacja do prawidłowego działania potrzebowała obiektowego języka programowania wysokiego poziomu, bazy danych wraz z serwerem, połączenia z bazą z pomocą wybranej technologii oraz elementów graficznych ilustrujących podgląd postępu pracy w aplikacji. Zdecydowano, że program będzie aplikacją internetową ze względu na rosnącą wciąż popularność Internetu oraz łatwiejszy dostęp szerokiego grona użytkowników do aplikacji.

W tym celu wygląd aplikacji został określony standardowo kodem HTML oraz CSS. Aby strony spełniały standardy wytyczone przez dzisiejsze trendy, użyto jednego z najbardziej popularnych i cenionych frameworków, czyli Bootstrapa. Biblioteki jQuery oraz Popper.js zostały zainstalowane dla większych możliwości rozwoju aplikacji w przyszłości, a także dla uzyskania pełnej funkcjonalności używanego frameworka. Dzięki temu narzędziu strony aplikacji są responsywne, zatem stronę będzie można uruchomić na ekranie o każdej rozdzielczości, ponadto wyglądają estetycznie oraz są proste w obsłudze i intuicyjne.

Baza danych jest oparta na systemie MySQL, ponieważ system ten jest prosty w obsłudze i pozwala na graficzną edycję danych wraz z podglądem zapytań kierowanych do bazy. Jako serwer lokalny został użyty jeden z najpopularniejszych serwerów, czyli Apache. Dla większej integracji całości oraz prostoty obsługi serwera, oba powyższe narzędzia zostały pobrane wraz z aplikacją XAMPP, pozwalającą na łatwe zarządzanie połączeniem z serwerem oraz z bazą danych.

Jako język programowania głównie został zastosowany język PHP, pozwalający na bezpieczne łączenie się z bazą danych. Wybór wydaje się logiczny, ze względu na zastosowanie systemu MySQL, który jest silnie zintegrowany z powyższym językiem. Język PHP do zastosowań wdrożonych w aplikacji jest prosty i logiczny w konstrukcji, pozwala na bardzo przystępne programowanie wielu funkcji witryny, jak również tworzenie sesji, dzięki czemu pozwala to użytkownikowi zapisać całą konfigurację, rozwiązując bardzo szybko problem przekazu informacji między stronami.

Aby połączenie bazy danych ze stroną było dynamiczne i nie wymagało odświeżania strony za każdym razem, użyto technologii AJAX. Dzięki specjalnemu zapisowi w języku JavaScript używającego pliku PHP, było możliwe wyświetlanie aktualnych informacji z bazy danych na stronie dla danej części. Jest to kolejna prosta technika, nie wymagająca zastosowania takich narzędzi jak Node.js czy AngularJS.

Wszystkie kluczowe strony zawarte w aplikacji, które odpowiadają za wyświetlanie wyników z bazy danych, wymagają odpowiednich plików w języku PHP. Zadanie to ułatwił program firmy JetBrains - PhpStorm. Dzięki temu narzędziu, edycja kodu w takich językach jak HTML, CSS czy PHP okazała się dużo prostsza. Podświetlanie składni odpowiednich tagów czy elementów zdecydowanie poprawia czytelność kodu, co wpływa pozytywnie na aplikację. Program oferuje również zapisywanie aplikacji na bieżąco oraz sprawdzanie kodu pod względem błędów w czasie rzeczywistym, co skraca czas testowania aplikacji w celu wychwycenia największych błędów.

Strony wymagają też odpowiednich elementów graficznych. W tym celu zostały zastosowane programy z pakietu Adobe: Illustrator oraz Photoshop, jak również program Blender, który posłużył do edycji grafiki 3D. Pierwszy program, pozwalający na tworzenie grafiki wektorowej, okazał się bardzo pomocny przy tworzeniu logo aplikacji występującego na stronie, czy też ikon przycisków dostępnych na stronie. Z kolei Photoshop został użyty do obróbki zrzutów ekranu uzyskanych przy pomocy programu Blender, w którym model samochodu w 3D został ukazany na obrazku 2D. Precyzja programu firmy Adobe pozwoliła na przycięcie wszystkich obrazków do takich samych rozmiarów. Ponadto umożliwił on umieszczenie obrazków części na przezroczystej warstwie alfa, co pomogło w realizacji pomysłu dotyczącego podglądu postępu.

# PODSTAWY TEORETYCZNE

## Podstawy budowy samochodu

Mówiąc o samochodzie osobowym, na myśl przychodzi najczęściej pojazd użytkowy o stosunkowo małych gabarytach (w porównaniu na przykład do samochodów ciężarowych), posiadający nadwozie i podwozie wraz z kołami (najczęściej cztery koła). O tym, jak rozbudowane potrafią być te dwa pierwsze elementy, świadczy chociażby artykuł z Wikipedii o budowie samochodu [[21](#Wik2)]. Na nadwozie składają się elementy zewnętrzne i wnętrza. Do tych pierwszych należą: karoseria, błotniki, zderzaki, wszelkiego rodzaju szyby, wycieraczki, dach, chłodnica, pokrywa komory silnika, maskownica (część nadwozia zakrywająca silnik) czy spoiler. Natomiast elementami wnętrza są wszystkie części, które otaczają kierowcę i pasażerów w samochodzie. Należą do nich m. in. deska rozdzielcza, tablica rozdzielcza, wszelkie przełączniki i regulatory, poduszki powietrzne, kierownica, fotele, lusterko wsteczne czy pasy bezpieczeństwa.

Jeśli chodzi o podwozie, jest tu znacznie więcej elementów. Można je podzielić na cztery zasadnicze grupy: układ napędowy i układ przeniesienia napędu, układ hamulcowy, układ kierowniczy i zawieszenie. Wszystkie te grupy określają części konieczne do sprawnego i prawidłowego działania pojazdu, warunkują jego zachowanie oraz możliwości na drodze, a także bezpieczeństwo, na które w ostatnim czasie kładzie się ogromny nacisk. Stąd wniosek, iż niełatwo jest przeciętnemu człowiekowi zrozumieć szczegółową budowę samochodu.

## Język HTML

W aplikacji użyto narzędzi i technologii wymienionych wcześniej. Podstawą do stworzenia aplikacji internetowej jest znajomość języka HTML. HTML, czyli skrót od *HyperText Markup Language*, jest podstawowym narzędziem do tworzenia każdej, nawet najprostszej witryny. Według Wikipedii, jest to „*hipertekstowy język znaczników, wykorzystywany do tworzenia dokumentów hipertekstowych*” [[22](#Wik3)].

Wszystkie strony programu oczywiście używają tego języka, choć nie jest on głównym narzędziem użytym w pracy. Za jego pomocą został stworzony formularz na stronach konfiguratora, dzięki któremu istnieje możliwość wyboru elementu przez użytkownika. Język ten umożliwił również przejrzysty i pomocny podział na sekcje w dokumencie przy użyciu tagu *„<div>”*. Jako swoisty korpus aplikacji, za jego pomocą jest możliwe także wyświetlanie wszelkich elementów na stronie - zarówno obrazków części samochodowych, jaki i tekstu informującego użytkownika o tym, co wybrał bądź gdzie skieruje go dany przycisk. Przyciski na stronie zostały opisane różnymi tagami, w zależności od funkcji. Przycisk, który umożliwia przejście do następnego kroku, posiada tag *„<input>”* (ze względu na to, że dotyczy formularza) wraz z określonymi właściwościami, natomiast pozostałe przyciski opisano tagiem *„<button>”*. Wyświetlanie obrazków zostało zrealizowane przy pomocy tagu *„<img>”*.

Wspomniane wcześniej tagi to charakterystyczny element tego języka [[35](#Lem16)]. Dokument z rozszerzeniem *.html* zaczyna się zwykle tagiem *„<!DOCTYPE html>”*, co oznacza, że domyślnie typ tego dokumentu to *html*. Następnie znajdują się tagi głównych sekcji dokumentu, czyli *„<head>”* (nagłówek strony), *„<body>”* (tak zwane ciało strony) oraz *„<footer>”* (stopka strony). Każdy z nich musi być zamknięty przed końcem dokumentu, który jest oznaczony poprzez zakończenie tagu html (*</html>*). Jak widać, tag kończący to po prostu ten sam tag, poprzedzony jedynie znakiem *slash* „/” wewnątrz znaczników *„<>”*. Tag *„<div>”* oznacza sekcję dokumentu wydzieloną przez programistę. Umożliwia to umieszczenie dowolnego elementu w konkretnym miejscu, nie powodując chaosu na stronie.

Formularz można utworzyć przy pomocy tagu *„<form>”*. Użyty typ formularzu w aplikacji to lista rozwijana z możliwością wyboru opcji, wywoływana poprzez tag *„<select>”*. Każda z opcji jest opisana tagiem *„<option>”* i posiada własną wartość. Zatem wewnątrz tagów można również określić właściwości konkretnego elementu na stronie. Dotyczy to również wywoływania funkcji ze skryptów, jak i innych plików.

Często spotyka się takie działania, na przykład przy umieszczaniu obrazków. Język wymaga do tego użycia tagu *„<img>”*. Jest to jeden z nielicznych tagów, który nie wymaga tagu zamykającego, jednak aby działał, należy koniecznie określić źródło obrazka (poprzez właściwość *src* wewnątrz tagu). Źródło jest to ścieżka dostępu do danego pliku. Może występować w postaci lokalnej (umieszczonej na dysku użytkownika) bądź w postaci adresu strony lub serwera, na których plik jest umieszczony. Zatem tag będzie wyglądał na przykład w ten sposób: *<img src=”C:\Users\User\Pictures”>,* lub tak: *<img src=”http://localhost/Pictures/”>.* Przy pomocy własności takich jak *class, style, margin, alt, width* czy *height* wewnątrz tagu można określić kolejno wywołaną klasę dla obrazka (na przykład na podstawie frameworka Bootstrap), styl obrazka, margines na stronie, podpis obrazka oraz jego rozmiary: szerokość i wysokość. Własności te należy umieścić za ścieżką do obrazka i oddzielić spacją, a wartości umieścić w cudzysłowie.

Warto również wspomnieć o tagu *„<a>”*, który odpowiada za umieszczanie na stronie hiperłącza. Za pomocą takiego zapisu, można na przykład umieścić łącze do innej podstrony, co jest wykorzystywane praktycznie na każdej stronie w sieci. Podobnie jak wcześniej, znacznik ten posiada unikalne atrybuty, jednak wymaga tagu zamykającego *„</a>”*. Najczęściej wykorzystywane to *href* oraz *target*. Pierwszy z nich opisuje adres strony bądź ścieżkę dostępu do pliku, który jest powiązany z hiperłączem. Natomiast atrybut *target* pozwala na ustalenie charakteru otwarcia łącza. Dzięki temu można na przykład otworzyć stronę w nowej karcie przeglądarki internetowej, bądź w nowym oknie (*target = ”\_blank”*). Tekst wpisany pomiędzy zakończeniem tagu *„<a>” a* tagiem zamykającym będzie tekstem, który zostanie wyświetlony na stronie. Po kliknięciu w taki tekst uruchomi się hiperłącze. Na przykład zapis *<a href = ”http://localhost/index.html”>strona główna</a>* wyświetli napis „strona główna”, który przy kliknięciu przekierowuje na stronę *index.html*. Zatem zapis hiperłącza, które pozwala przejść do konkretnej strony i otwiera się w nowej karcie, będzie wyglądał następująco: *<a href = ” http://localhost/index.html” target = ”\_blank”></a>*. Należy wspomnieć, iż w podanym przypadku zamiast tekstu zostanie wyświetlony link podany jako atrybut tagu *„<a>”*.

Ważnym elementem wyglądu jest odpowiednie formatowanie tekstu. W większości można zrobić to przy pomocy CSS, ale w języku HTML dostępne jest kilka typów nagłówków oraz tekst akapitowy. Można stosować nagłówki (zgodnie z numeracją, największy oznaczony numerem 1) *„<h1>”*, *„<h2>”* i *„<h3>”*, a także tekst akapitowy przy pomocy tagu *„<p>”*. Oddzielanie tekstu, jak również elementów na stronie w pionie można opisać tagiem *„<br />”*, który nie wymaga tagu zamykającego.

Przy pomocy tagu *„<style>”* możliwe jest umieszczenie własnego stylu dla danej strony, choć w praktyce do tego używa się osobnego pliku CSS, deklarując go tagiem *„<link>”* w sekcji *„<head>”*, na przykład *<link rel="stylesheet" href = "style.css">.* Jednak czasem istnieje potrzeba zdefiniowania osobnego stylu dla jednego elementu na tylko jednej stronie. W takim przypadku powyższy tag będzie pomocny, ponieważ używa on języka CSS, opisanego w następnym podrozdziale.

Do wyświetlania informacji o wybranej części dla użytkownika użyto formy tabelki. Jest to również sposób na uporządkowanie niektórych danych w pionie. Tabelkę opisuje się znacznikiem *„<table>”,* a wewnętrznymi tagami można określić jej strukturę. Przy pomocy tagu *„<tr>”* określa się nowy wiersz, a znacznikiem *„<td>”* nową kolumnę. Każdy z tych tagów oczywiście musi być zamknięty, tzn. wewnątrz tagu *„<tr>”* należy po otwarciu zamknąć tag *„<td>”* i na odwrót. Wewnątrz znaczników *„<tr>”* i *„<td>”* umieszcza się zawartość pojedynczej komórki tabeli.

Warto wspomnieć o jeszcze jednym tagu. Używając metody *Canvas* do rysowania grafiki na stronie, należy wydzielić obszar przeznaczony dla grafiki w kodzie języka HTML. Wykonuje się to poprzez umieszczenie w nim tagu *„<canvas>”* oraz określeniu odpowiednich atrybutów tego obszaru wewnątrz podanego znacznika.

## Kaskadowe arkusze stylów

„Kaskadowe arkusze stylów (ang. Cascading Style Sheets, w skrócie CSS) to język służący do opisu formy prezentacji (wyświetlania) stron WWW” [[23](#Wik13)]. Krótko mówiąc, jest to język odpowiadający za wygląd większości dzisiejszych stron internetowych, używany przy tworzeniu każdej bardziej złożonej witryny. Dzięki możliwości, jaką daje HTML, kod tego języka można umieścić w kodzie strony, jak i w osobnym pliku w formacie .css. To ostatnie rozwiązanie jest wykorzystywane praktycznie wszędzie, ze względu na czytelność kodu strony.

Podobieństwem w składni tego języka jest możliwość odniesienia się do konkretnych sekcji dokumentu, takich jak *„<head>”*, *„<body>”* czy *„<footer>”*, a także do poszczególnych jej elementów, na przykład nagłówków, akapitów, obrazków czy tabel. Odbywa się to poprzez odwołanie się do odpowiedniego znacznika bądź sekcji, na przykład *body {}*. Wyraz *body* jest tutaj selektorem, a cały zapis oznacza deklarację. Klamry oznaczają początek i koniec edycji stylu danego elementu, zatem wszystkie atrybuty należy określić pomiędzy nimi.

Aby przeglądarka internetowa wiedziała, jak powinny być wyświetlane poszczególne elementy strony, należy je określić w kodzie CSS używając właściwości. Odbywa się to poprzez wpisanie odpowiednich właściwości wraz z wartościami wewnątrz deklaracji, według schematu: *selektor { właściwość: wartość; }*, gdzie właściwość jest elementem stylu, który jest edytowany, a wartość określa konkretnie pozycję elementu bądź jego styl (na przykład margines bądź styl czcionki). Tak jak w wielu językach, pojedynczą instrukcję należy zakończyć średnikiem.

## Bootstrap

Bootstrap to „framework CSS, rozwijany przez programistów Twittera, wydawany na licencji MIT” [[24](#Wik15)]. Zawiera on zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Został wykorzystany jako baza do rozbudowy stylu CSS w programie.

Bootstrap posiada wiele zdefiniowanych klas, które ułatwiają tworzenie przejrzystego i intuicyjnego interfejsu na stronach internetowych [[40](#Spu13)]. Klasy te mogą dotyczyć zarówno elementów strony (na przykład tagów HTML), jak i poszczególnych właściwości tych elementów (przykładowo atrybuty tagów). W aplikacji zostało wykorzystane kilka podstawowych narzędzi, takich jak podział strony na 12 rzędów (ang. *rows*, klasa *row*), klasy dotyczące tagów (*container*, *container-fluid*, *navbar*, *navbar-toggle*, itp.), atrybutów (*img-responsive*, *collapse*). Odpowiadają one za funkcjonalność i estetykę interfejsu użytkownika oraz strony.

Warto wspomnieć, że jedną z największych zalet tego frameworka jest wbudowana responsywność. Oznacza to, że niezależnie od wielkości ekranu, na którym będzie otwarta strona aplikacji, użyte narzędzia będą dzielić zawartość strony w ten sposób, aby jej układ był możliwie jak najlepszy dla danej rozdzielczości. Dzięki temu aplikację można otworzyć zarówno na komputerze PC z dużym ekranem, jak i mniejszym laptopie, tablecie czy nawet smartfonie.

Selektor może odwoływać się również do nagłówków (na przykład *h1*), tekstu akapitowego, tabelki bądź klasy. Klasa umożliwia odwołanie się do konkretnego elementu w dokumencie, a więc umożliwia określenie stylu dla elementu oznaczonego nazwą tej klasy. Jest to przydatne w przypadku, gdy nie ma potrzeby zmieniać stylu na przykład każdej tabelki w danym pliku CSS, a jedynie pewną tabelkę na jednej bądź kilku stronach. Można to zrobić zapisując jako selektor nazwę danej klasy po kropce. Przykładowo, mając znacznik *„<div>”* z atrybutem klasy *container-fluid*, można również zamieścić w nim własną nazwę klasy, na przykład „bg-1”, do której będzie można odwołać się w pliku CSS. Zapis takiego znacznika będzie wyglądał następująco: *<div class = ”container-fluid bg-1”>*, natomiast odwołanie w pliku CSS: *.bg-1 {}*. Dzięki takiej operacji możliwe jest określenie stylu dla znacznika *„<div>”* określonego klasą „bg-1”.

## MySQL

Jednym z systemów zarządzania bazą danych opartą na języku SQL jest MySQL. Według Wikipedii jest to „wolnodostępny system zarządzania relacyjnymi bazami danych” [[25](#mysql)]. Jeśli chodzi o jego instalację, najprościej jest skorzystać z odpowiedniego pakietu. Zależnie od systemu operacyjnego będzie to XAMPP bądź LAMP. Taki pakiet zawiera zestaw narzędzi pomocny przy administrowaniu serwera, a także zarządzaniu bazą danych. Posiada również interpreter języka PHP, dzięki czemu można przy jego pomocy skonfigurować na przykład serwer lokalny, który będzie komunikował się ze stroną poprzez ten język.

Najważniejszym narzędziem dla bazy danych w tym pakiecie jest *phpMyAdmin*. Jest to „narzędzie służące do łatwego zarządzania bazą danych MySQL, napisane w języku PHP” [[26](#phpmy)]. Dzięki niemu można przy pomocy przeglądarki internetowej tworzyć i dowolnie modyfikować taką bazę, używając przy tym przyjaznego użytkownikowi interfejsu graficznego. Takie rozwiązanie powoduje, że wystarczy podstawowa wiedza z zakresu języka SQL, aby używać go do nawet dość skomplikowanych operacji.

Mimo, iż *phpMyAdmin* generuje kod zapytań w zależności od działań użytkownika, nie wszystkie operacje da się wykonać przy pomocy samego interfejsu. Warto więc znać podstawy języka SQL. Dzięki temu znacznie poszerza się możliwości tworzenia relacyjnych baz danych.

Narzędzie oferuje konsolę, gdzie można wpisywać kod w języku SQL (czyli kierować zapytania do bazy) [[36](#Lis171)]. Należy zauważyć, że każda taka pojedyncza instrukcja musi kończyć się znakiem średnika. Po akceptacji kodu serwer sprawdza, czy polecenia da się wykonać. Jeśli tak, to działanie zostaje zrealizowane. Może być to stworzenie nowej tabeli, modyfikacja już istniejącej, dodanie rekordu, aktualizacja dowolnej kolumny bądź usunięcie każdego elementu bazy. Najczęściej jednak zapytania związane są z wyświetlaniem odpowiednich danych po ustaleniu warunków.

Najbardziej podstawowym zapytaniem jest *SELECT*. Zaznacza ono te dane, które spełniają podane później warunki i wyświetla je. Konstrukcja prostego zapytania tego typu to na przykład: „*SELECT \* FROM tabela*;”. Oznacza ono, że użytkownik chce wyświetlić wszystkie rekordy z tabeli o nazwie *tabela*. Istotną uwagą jest fakt, że nazwy tabel powinny być pisane jednym ciągiem (tzn. bez spacji). Pomiędzy *SELECT*, a*FROM* znajduje się miejsce na wpisanie nazw kolumn. Symbol gwiazdki oznacza, że polecenie dotyczy wszystkich kolumn z danej tabeli bądź tabel. Zarówno nazwy kolumn, jak i tabel powinny być oddzielane przecinkami. Do polecenia można dodać warunek, na przykład: „*SELECT \* FROM tabela WHERE ‘id’ = 1;*”. Zostaną wówczas wybrane tylko te rekordy, które mają wartość 1 w kolumnie *id*.

## Język PHP

Kolejnym językiem, który odpowiada za łączenie aplikacji z bazą danych, a także jest głównym językiem użytym do jej stworzenia, jest PHP. Nazwa jest „skrótem” od PHP Hypertext Preprocessor. Jest językiem dedykowanym do komunikacji z bazą danych na serwerze. Według Wikipedii, to „interpretowany skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym” [[27](#Wik4)].

Podstawową zaletą języka PHP jest to, że kod wykonywany jest po stronie serwera [[38](#Nix15)]. Oznacza to, że przeglądarka widzi jedynie wynik jego działania, czyli to, co zwraca plik po wykonaniu kodu. Niemożliwe jest zatem podejrzenie kodu strony napisanej w PHP, można jedynie wywnioskować jego użycie po adresie WWW. Dlatego często wykorzystuje się go m. in. do formularza rejestracji oraz przy zapisywaniu haseł. Język ten jest odpowiednikiem technologii ASP.NET, bowiem przy pomocy obu narzędzi można nawiązać połączenie z serwerem.

W PHP zostały napisane wszystkie strony dotyczące bezpośrednio konfiguratora, jak i pliki odpowiadające za pobieranie informacji z bazy danych oraz dostarczanie ich w czasie rzeczywistym. Kod języka PHP, aby mógł zostać umieszczony w pliku, należy deklarować specjalnym tagiem *„<?php”*. Jeśli w pliku dodatkowo znajduje się kod HTML, tag ten powinien być zamknięty tagiem *„?>”* i umieszczony wewnątrz jednej z głównych sekcji dokumentu (*head* lub *body*).

Bardzo często jest tak, że plik posiada wyłącznie kod PHP i jest wykorzystywany przez inny plik bądź element strony. Wynika to z faktu, że nieraz wystarczy samo nawiązanie połączenia z bazą danych oraz skierowanie do niej odpowiedniego zapytania, aby osiągnąć zamierzony cel. Pierwszym takim plikiem może być kilka linijek kodu odpowiedzialnych za zadeklarowanie czterech zmiennych wykorzystywanych przy połączeniu np. z bazą danych. Zmienna w języku PHP deklarowana jest przy pomocy znaku „$” przed jej nazwą. Na przykład po określeniu *$host*, *$db\_user*, *$db\_password*, *$db\_name* w takim pliku, można później wykorzystać te zmienne przy ustanawianiu połączenia z bazą danych odwołując się do niego przy pomocy funkcji *include* i podając nazwę pliku. Należy oczywiście pamiętać, że tutaj, podobnie jak w języku CSS, koniec pojedynczej instrukcji jest określany poprzez znak średnika.

Warto od początku zabezpieczyć program na wypadek błędów. W przypadku błędu połączenia z serwerem, taki zapis spowoduje wyświetlenie odpowiedniego komunikatu zamiast zawieszać działanie strony. Instrukcja *try...catch* jest jedną z podstawowych deklaracji dobrego programu w każdym języku. W przypadku PHP, taki zapis wygląda następująco: *try { instrukcje } catch ( Exception $e ){ wyjątek }*, gdzie *Exception* oznacza wyjątek, czyli numer i nazwę błędu, który wystąpił, a zmienna *e* przechowuje informacje o błędzie, który wystąpił. Zmienna taka jest później wykorzystana w miejscu *wyjątek*. W nawiasach klamrowych zapisane są instrukcje, które są wykonywane w określonych przypadkach. Tutaj warunkiem jest wystąpienie błędu, a więc kiedy połączenie jest udane, następuje wykonanie instrukcji w bloku *try*. W przeciwnym razie zostanie wykonane to, co zapisano w bloku *catch*. Przykładowa deklaracja w tym miejscu wygląda tak: *echo ‘Informacja deweloperska’.$e;*. Ten zapis spowoduje wyświetlenie się podanego napisu wraz z numerem błędu i jego nazwą w języku angielskim.

Należy zauważyć, że w poprzednim zapisie wykorzystano dwie rzeczy. Pierwsza to funkcja *echo*, która powoduje wypisanie tekstu z zachowaniem odpowiednich znaczników (na przykład języka HTML) oraz zmiennych, przy odpowiednim wykorzystaniu apostrofów i cudzysłowu (apostrof jest silniejszy od cudzysłowu). Drugim jest konkatenacja, która umożliwia połączenie kilku elementów w jeden zapis (na przykład tekst, znaczniki oraz zmienne) przy pomocy znaku kropki. Wykorzystuje się to przy zastosowaniu różnych elementów w jednej linijce tekstu. Funkcja *echo* pozwala na określenie wyniku działania programu i daje możliwość wyświetlenia go na stronie. W przypadku, gdy w zapisie pojawiają się różne zmienne, wartości czy atrybuty, warto wewnątrz cudzysłowu zastosować apostrof i w nim umieścić element, który nie powinien być traktowany przez interpreter jako tekst.

Kiedy wymagane jest połączenie z bazą danych, można użyć pliku opisanego wcześniej. Trzeba jednak wiedzieć, jaka funkcja odpowiada za takie połączenie. Może to wyglądać w ten sposób: *$polaczenie = new mysqli($host, $db\_user, $db\_password, $db\_name)*. Widać tutaj funkcję *mysqli*, która kieruje żądanie do serwera używając wcześniej zadeklarowanych zmiennych (jeśli znajdują się one w innym pliku, należy przed tym zapisem użyć funkcji *include*).

Po ustanowieniu takiego połączenia, można wysyłać zapytania. Wystarczy zdefiniować pewną zmienną, która przechowa odpowiedni tekst zawierający zapytanie w języku SQL, zrozumiały dla bazy danych. Na przykład:

*$sql = $polaczenie->query('SELECT \* FROM `cookies`')*

Wykorzystano tu zmienną *polaczenie*, aby wykonać podane zapytanie. Znajduje się tutaj również funkcja *query*, dzięki której program wie dokładnie, jaka jest jego treść. Jeśli jednak wystąpiłby jakiś błąd w wyniku wysłania tego zapytania, warto zabezpieczyć tę instrukcję dodatkowym zapisem: *throw new Exception($polaczenie->error)*. Wyrażenie *throw new Exception* oznacza „rzuć nowym wyjątkiem”, czyli strona wyświetli w ten sposób komunikat o błędzie z połączeniem. Pozwoli to programiście na zdiagnozowanie problemu z programem. Trzeba pamiętać też o tym, że po każdym wykonaniu instrukcji należy zamknąć połączenie z bazą danych. Przykład:

*$rezultat->close(); mysqli\_close($polaczenie);* oznacza, że zmienna *rezultat* przechowująca wynik zapytania z bazy danych została zamknięta, a zaraz potem połączenie z MySQL.

Ważną cechą używaną w języku PHP jest możliwość tworzenia sesji. Sesja, czyli „obiekt, zapamiętujący przez pewien czas na serwerze szczegóły dotyczące połączenia z klientem” [[28](#sesja)], jest przydatnym narzędziem do zapisywania działań użytkownika. Uruchomienie sesji na stronie odbywa się poprzez instrukcję *session\_start()*, co daje również możliwość stosowania tzw. zmiennych sesyjnych. Zmienna taka różni się od zwyczajnej tym, że jest zapisywana w sesji i może być przywołana również na innej stronie. Przykładowa deklaracja zmiennej sesyjnej: *$\_SESSION['s'] = 1*. Nazwa tej zmiennej to *s*, ale odwołanie do niej odbywa się poprzez pełną nazwę *$\_SESSION['s']*.

Istnieje również możliwość pozyskania zmiennej poprzez odwołanie się do elementu strony, która korzysta z pliku PHP. Może to być na przykład wartość opcji z listy *„<select>”* znajdującej się w formularzu HTML. Jeśli zostanie zdefiniowany atrybut *name* w znaczniku listy, na przykład: *<select name=”pole\_wyboru”>*, to wtedy w pliku PHP można przypisać zmiennej wartość aktualnie wybranej opcji: *$p = $\_POST['pole\_wyboru']*. Zapis *POST* oznacza pozyskanie tej wartości metodą POST, która jest używana głównie do przesyłania danych z formularzy na stronie. Istnieje również metoda GET, która umieszcza dane w postaci parametr=wartość w adresie strony. Gdyby zaszła potrzeba uzyskania zmiennej wykorzystywanej przez kod JavaScript w ramach jednej sesji, można to zrobić w ten sposób: *$q = intval($\_GET['q'])*. Została tu wykorzystana funkcja *intval.* Służy ona do przekształcenia zmiennej do typu integer (czyli zwraca całkowitą wartość zmiennej).

Ważnym elementem każdego języka programowania są instrukcje warunkowe. W PHP można wyróżnić dwie główne: *If* oraz *switch*. Konstrukcja pierwszej to *If (warunek) { instrukcje }*, natomiast druga wygląda tak: *switch (zmienna) { case wartość 1: instrukcja; break; case wartość 2: instrukcja; break; ... }*. Pierwsza to podstawowa instrukcja warunkowa, wykorzystywana wszędzie tam, gdzie są przynajmniej dwa przypadki (na przykład obsługa błędów). Za to instrukcja *switch* najlepiej nadaje się wtedy, gdy program docierając do jakiegoś momentu, musi przeanalizować więcej przypadków w zależności od jednego warunku.

Inną istotną rzeczą są pętle. Wyróżnia się głównie pętle typu *while* i *for*. Pierwsza ma konstrukcję: *while (warunek) { instrukcja }*. Dopóki *warunek* jest prawdziwy, wykonywana jest *instrukcja*. Natomiast pętla *for* wygląda tak: *for ( wartość początkowa; wartość końcowa; krok) { instrukcja }*. Pętla wykonywana jest określoną ilość razy. Zaczyna od *wartość początkowa* i dodaje do niej *krok* po każdym wykonaniu *instrukcja*, aż do osiągnięcia *wartość końcowa*.

## JavaScript

Jeden z najbardziej popularnych języków skryptowych, dzisiaj używany do tworzenia praktycznie każdej witryny. JavaScript jest to „skryptowy język programowania, stworzony przez firmę Netscape, najczęściej stosowany na stronach internetowych” [[29](#js)]. Charakterystyczną cechą tego języka jest fakt, że odpowiada on za stronę funkcjonalną witryny internetowej. Za jego pomocą można modyfikować i usuwać zawartość dowolnej części strony. W przeciwieństwie do języka PHP, kod języka JavaScript jest wykonywany głównie po stronie przeglądarki i może być wyświetlony dla użytkownika.

Kod w tym języku można umieścić zarówno w kodzie HTML, jak i PHP. Odbywa się to poprzez wyznaczenie pola dla języka skryptowego, czyli umieszczenie tagu *„<script>”* wraz z zamykającym go *„</script>”*. Pomiędzy tymi znacznikami można definiować dowolne instrukcje w JavaScript. Popularnym działaniem jest również umieszczenie odniesienia do pliku z rozszerzeniem .js przez zapis: *<script src="adres do pliku"></script>*, gdzie w miejscu „adres do pliku” umieszcza się ścieżkę dostępu. Istnieje również możliwość odwołania się do adresu strony internetowej. Jeśli strona ta zawiera plik o takim rozszerzeniu, wówczas można zawrzeć go w dokumencie i korzystać z jego funkcji. Przykładowy zapis:

*<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"> </script>* oznacza odniesienie się do pliku frameworka Bootstrap w wersji 3.3.7. Przy pomocy takiego zabiegu można przyspieszyć działanie aplikacji, ponieważ plik ten znajduje się na innym serwerze i strona nie musi go wczytywać. Wszystkie te deklaracje należy najlepiej umieszczać w sekcji *head* dokumentu HTML. Można to zrobić również na końcu dokumentu lub sekcji *body*.

Funkcje w JavaScript definiuje się podobnie jak w innych językach: *function nazwafunkcji(parametr) { instrukcje }.* W miejscu *nazwafunkcji* należy wpisać własną nazwę tworzonej funkcji - będzie to nazwa, przy pomocy której można ją wywołać. Natomiast *parametr* to zmienna, która jest wynikiem działania tej funkcji. Aby można było wykonać działanie, często niezbędna będzie deklaracja zmiennych. Przykładowo: *var xhttp* oznacza zdefiniowanie zmiennej *xhttp* nieznanego typu. Typ zmiennej określany jest dopiero w momencie przypisywania jej wartości bądź wykorzystania w jakimś celu. Instrukcje warunkowe wyglądają tak samo jak w języku PHP. Występują tu instrukcje *If* oraz *switch*.

Często wykorzystywaną cechą JavaScript jest możliwość przypisania funkcji do elementu na stronie, na przykład przycisku. Wiąże się to z obsługą zdarzeń. Kiedy użytkownik dokonuje pewnej akcji na stronie związanej z określonym elementem, może to wywołać funkcję zdefiniowaną w kodzie języka skryptowego. Jednym z takich zdarzeń jest *onclick*. Wywołuje ono przypisaną do przycisku funkcję poprzez kliknięcie go. Przykład: *<input type = 'submit' value = 'Dalej' onclick = 'sprawdz()' />*. Po kliknięciu tego przycisku, zdarzenie wywołuje funkcję *sprawdz*.

Przydatnym narzędziem przy pracy na przykład z językiem PHP jest AJAX [[34](#Bri11)]. Wymaga ono dodatkowego pliku uwzględnionego na stronie. Można to zrobić podając link: *<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">* w sekcji *head* dokumentu HTML. Pozwala na połączenie się z bazą danych wykorzystując na przykład plik PHP i dynamiczną zmianę zawartości strony. Doskonale sprawdza się przy wyświetlaniu informacji z bazy danych o wybranej pozycji z listy. Przykład zastosowania funkcji [Listing 1] wykorzystującej AJAX:

function showPart(str) {

var xhttp;

if (str === "") {

document.getElementById("podpowiedź").innerHTML = "brak";

return;

}

xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState === 4 && this.status === 200) {

document.getElementById("podpowiedź").innerHTML = this.responseText;

}

};

xhttp.open("GET", "sync.php?q="+str, true);

xhttp.send();

}

Listing 1: Wykorzystanie technologii AJAX [źródło: opracowanie własne].

Zaczynając od początku, zdefiniowana zostaje funkcja *showPart* z parametrem *str*. W niej zostaje zadeklarowana zmienna *xhttp*. Następnie program sprawdza warunek, czy zmienna *str* jest pustym łańcuchem znaków. Jeśli tak, to wtedy element na stronie o *id* równym *podpowiedź* zostanie zmieniony na „brak”. Do zmiennej *xhttp* zostaje przypisany nowy element *XMLHttpRequest*. Dla zdarzenia *onreadystatechange* jest zdefiniowana funkcja. Zdarzenie to bada, kiedy dany element jest gotowy na wysłanie żądania do pliku PHP. Następnie sprawdza warunek: *this.readyState === 4* oznacza pełną gotowość na odpowiedź po wysłaniu żądania, natomiast *this.status === 200* oznacza, że status żądania ma numer 200 (czyli „OK”). Właściwość *readyState* określa stan połączenia na podstawie jednego z pięciu stanów [[30](#ajax)]:

0: żądanie nie zostało zainicjowane

1: nawiązano połączenie z serwerem

2: żądanie zostało otrzymane

3: przetwarzanie żądania

4: żądanie zostało wykonane i odpowiedź jest gotowa

Aby funkcja została zrealizowana, muszą zostać spełnione oba warunki. Zatem żądanie musi zostać wysłane, odpowiedź gotowa, a status żądania musi mieć numer 200. Wtedy element o *id* równym *podpowiedź* zostanie zmieniony na wynik działania instrukcji zawartych w pliku *sync.php*. Po tym następuje otwarcie tego pliku, uzyskanie danych metodą GET oraz dopisanie zmiennej *str*, która podtrzymuje działanie funkcji na stronie, a następnie wysłanie tego wyniku na stronę poprzez funkcję *send()*.

# OPIS I TWORZENIE APLIKACJI

## Przygotowanie środowiska

Do rozpoczęcia pracy z projektem należy przygotować odpowiednio skonfigurowane środowisko. Ponieważ przeglądarka Chrome jest narzędziem wykorzystywanym na co dzień, jej instalacja nie była konieczna. Zainstalowano aplikację XAMPP. Następnie został zainstalowany i skonfigurowany program PhpStorm (serwer lokalny). Do tworzenia i obróbki grafiki 2D zainstalowane zostały programy z pakietu Adobe: Photoshop oraz Illustrator, natomiast do grafiki 3D program Blender.

## Tworzenie bazy danych

Aplikacja do pracy wymaga bazy danych, z której będą pobierane dane o częściach samochodowych. Zanim zacznie się tworzyć aplikację, należy najpierw określić, co aplikacja ma pokazywać użytkownikowi po wybraniu dostępnej opcji.

### Struktura

Baza danych została utworzona przy pomocy aplikacji XAMPP. Zawiera ona osiem tabel niepowiązanych ze sobą relacjami, aby maksymalnie uprościć jej strukturę. Każda z siedmiu tabel zawiera elementy, do których odwołuje się aplikacja w konkretnym kroku konfiguratora. Wszystkie są zapisane tym samym wzorem, aby ułatwić implementację zapytań do bazy danych w języku PHP.

Pierwsza tabela [Tabela 1] zawiera rodzaje nadwozia i została wykorzystana w pierwszym kroku konfiguratora.

|  |  |  |
| --- | --- | --- |
| ID | Rodzaje\_nadwozia | Obrazek |
| 1 | Coupe | coupe.jpg |
| 2 | Sedan | sedan.jpg |

Tabela 1: Pierwsza tabela z bazy danych zawierająca rodzaje nadwozia

[źródło: opracowanie własne].

W drugiej [Tabela 2] umieszczono różne podwozia:

|  |  |  |
| --- | --- | --- |
| ID | Podwozia | Obrazek |
| 1 | BMW series 3 | bmw\_series\_3.jpg |
| 2 | Audi S4 | audi\_s4.jpg |

Tabela 2: Druga tabela zawierająca podwozia [źródło: opracowanie własne].

Trzecia tabela [Tabela 3] dotyczy silników:

|  |  |  |
| --- | --- | --- |
| ID | Silniki | Obrazek |
| 1 | BMWS60 | bmws60.jpg |
| 2 | Audi V8 | audi\_v8.jpg |

Tabela 3: Trzecia tabela zawierająca silniki [źródło: opracowanie własne].

Czwarta [Tabela 4] zawiera skrzynie biegów:

|  |  |  |
| --- | --- | --- |
| ID | Skrzynie\_biegow | Obrazek |
| 1 | Tiptronic C60 | tiptronic\_c60.jpg |
| 2 | 6-biegowa ręczna | 6\_biegowa\_reczna.jpg |

Tabela 4: Czwarta tabela zawierająca skrzynie biegów [źródło: opracowanie własne].

Piąta [Tabela 5] dotyczy różnych wnętrz:

|  |  |  |
| --- | --- | --- |
| ID | Wnetrza | Obrazek |
| 1 | Alcantara brązowa | wnetrze1.jpg |
| 2 | Alcantara biała | wnetrze2.jpg |

Tabela 5: Piąta tabela zawierająca wnętrza [źródło: opracowanie własne].

Szósta tabela [Tabela 6] zawiera rodzaje karoserii:

|  |  |  |
| --- | --- | --- |
| ID | Karoserie | Obrazek |
| 1 | Coupe1 | coupe.jpg |
| 2 | Sedan1 | sedan.jpg |

Tabela 6: Szósta tabela zawierająca karoserie [źródło: opracowanie własne].

Natomiast siódma [Tabela 7] przechowuje koła:

|  |  |  |
| --- | --- | --- |
| ID | Kola | Obrazek |
| 1 | Koła1 | kola1.jpg |
| 2 | Koła2 | kola2.jpg |

Tabela 7: Siódma tabela zawierająca koła [źródło: opracowanie własne].

Ostatnia tabela służy do zapisywania danych o częściach, które wybrał użytkownik [Tabela 8]:

|  |  |
| --- | --- |
| ID | Wybrane\_czesci |
| 1 | Coupe |
| 2 | BMW series 3 |
| 3 | BMWS60 |
| 4 | Tiptronic C60 |
| 5 | Alcantara brązowa |
| 6 | Coupe1 |
| 7 | Koła1 |

Tabela 8: Tabela „cookies”, w której zapisywane są dane części wybranych przez użytkownika. Domyślnie tabela ta jest pusta, a elementy dodawane przez program, pokazano tu przykładową kompletną konfigurację pojazdu [źródło: opracowanie własne].

Każda tabela oprócz ostatniej posiada trzy kolumny. Pierwsza zawiera numer ID części i zarazem jest jej identyfikatorem. Druga, zatytułowana rodzajem części, zawiera nazwę elementu wyświetlaną w aplikacji. Natomiast trzecia kolumna zawiera nazwy obrazków elementów wykorzystywane przy wyświetlaniu podglądu wybranej części. W każdej z siedmiu tabel umieszczono po 2 części, umożliwiając tym samym wybór dla użytkownika.

### Konfiguracja

Każda tabela ma klucz główny, zwany ID, a także porównywanie napisów określone jako utf8\_polish\_ci, aby baza obsługiwała polskie znaki. Ze względu na uproszczenie działania, zrezygnowano z wprowadzania relacji między tabelami, a program odwołuje się do konkretnej tabeli przy pomocy zapytań w języku SQL.

## Serwer

Wybrany został język PHP oraz pakiet XAMPP, baza oraz aplikacja do wymiany danych wykorzystują serwer lokalny Apache. Zanim w przeglądarce zostanie uruchomiona aplikacja, należy go uruchomić oraz włączyć obsługę bazy danych MySQL. Został on skonfigurowany jako serwer lokalny w programie PhpStorm.

## Aplikacja

Rdzeniem programu jest aplikacja internetowa, która korzystając z bazy danych, pozwala użytkownikowi na tworzenie pojazdu.

### Struktura, wygląd i funkcjonalność

Program posiada kilka stron napisanych w języku HTML oraz osiem stron w języku PHP, które odpowiadają za pracę konfiguratora. Komunikacja z bazą danych odbywa się za pośrednictwem odrębnych plików.

Strona główna (*index.html*) wita użytkownika w serwisie oraz prezentuje mu cel i działanie konfiguratora, wraz z obrazkami ilustrującymi opisy. Na górze znajduje się pasek nawigacji, przy pomocy którego można przejść do głównej części aplikacji, a także dowiedzieć się więcej o programie (*o\_aplikacji.html*) bądź wyświetlić stronę z danymi kontaktowymi twórcy witryny (*kontakt.html*).

Plik *conf.php* jest pierwszą stroną w konfiguratorze, która prosi o wybranie rodzaju nadwozia pojazdu, a więc podstawę do budowy samochodu. Dzięki temu konfigurator wie, jaki rodzaj auta użytkownik chce stworzyć. Po lewej stronie dostępny jest podgląd tworzonego samochodu, natomiast po prawej stronie znajduje się lista rozwijana zawierająca nazwy części pobrane z bazy danych. Po wybraniu dowolnego elementu z listy wyświetla się tabelka z informacjami o nim, a poniżej obrazek pokazujący jego wygląd. Pole to aktualizuje się przy każdej zmianie wybranej opcji. Jeśli żadna nie zostanie wybrana (pierwsza opcja), wówczas strona wyświetli podpowiedź informująca użytkownika o konieczności wybrania innego pola z listy.

Kolejne strony konfiguratora pod względem wyglądu różnią się jedynie podpowiedzią oraz opcjami na liście i informacjami importowanymi z bazy danych. Istotny fragment programu stanowią trzy pliki napisane w języku PHP, odpowiedzialne za połączenie oraz wymianę danych z bazą. Są to: *connect.php*, *lista.php*, *sync.php* oraz *dataupdate.php*. Podsumowanie tworzonego samochodu, czyli plik *sum.php*, wyświetla dane z tabelki „cookies” o częściach, które zostały wybrane przez użytkownika. Dane te można zapisać do pliku w formacie pdf, włącznie z podglądem stworzonego auta, bądź usunąć przechodząc do strony głównej (poprzez plik *resetcookies.php*) i akceptując komunikat.

Za wyświetlanie komunikatów, informacji o błędzie oraz przekierowanie do strony głównej, a także wyświetlanie tabelki oraz podglądu tworzonego auta odpowiadają osobne pliki napisane w języku JavaScript. Do tych plików odwołuje się każda strona konfiguratora. Oczywiście zapewniają one również płynne działanie strony oraz frameworka Bootstrap. Każdy z nich posiada zdefiniowaną funkcję wraz z odpowiednimi zmiennymi. Niektóre z tych funkcji odwołują się także do jednego z trzech wyżej wymienionych plików (najczęściej *sync.php*).

Wygląd stron aplikacji oraz układ elementów jest utrzymywany w porządku dzięki frameworkowi, a także własnemu plikowi CSS (*style1.css*). Odpowiednio dobrane proporcje oraz zdefiniowane klasy siatki (ang. *grid*) umożliwiają umieszczenie elementów z prawej strony poniżej podglądu, dzięki czemu na ekranach smartfonów aplikacja jest czytelna i pozwala na taką samą funkcjonalność, jak w wersji na komputery PC.

### Szkielet aplikacji

Podstawą rozmieszczenia elementów na stronie zajmuje się kod HTML. Wszystkie strony aplikacji zawierają bardzo podobny układ, który został nieco zmodyfikowany na potrzeby konfiguratora. Wygląd oraz działanie zostały określone poprzez framework Bootstrap, osobny plik CSS oraz pliki JavaScript.

Na stronie głównej został umieszczony pasek nawigacji oraz trzy elementy *container-fluid*, a każdy z nich zawiera tekst bądź obrazki. Ostatni został podzielony na trzy kolumny, które umożliwiły umieszczenie trzech ilustracji ilustrującymi działanie konfiguratora wraz z opisami. Na dole strony znajduje się stopka, która podaje podstawowe informacje o twórcy witryny, roku jej utworzenia oraz o zastrzeżeniu praw autorskich.

### Konfigurator

Plik *conf.php*, do którego można przejść prosto ze strony głównej, oraz pliki od *conf2.php* do *conf7.php*, są jej zmodyfikowanymi szablonami. Zawierają one pasek nawigacji, dwa elementy *container-fluid*, z czego drugi jest podzielony na dwie kolumny o różnej szerokości (kolumny te zmieniają wielkość w zależności od rozmiaru ekranu urządzenia) oraz stopkę. W nagłówku strony umieszczono odwołanie do odpowiednich plików JavaScript, które wpływają na funkcjonowanie strony. Jest tu link do pliku AJAX z serwera Google oraz linki do odpowiednich plików frameworka Bootstrap. Takie odwołanie powoduje, że strona ładuje się szybciej, ponieważ plik znajduje się na innym serwerze. Dopiero potem strona wczytuje plik CSS, aby wcześniejsze pliki nie zmodyfikowały ustalonego stylu. Znajdują się tutaj również wszystkie pliki skryptowe odpowiedzialne za uruchamianie się odpowiednich funkcji konfiguratora, w tym funkcjonalność przycisków (*powrót.js*) czy wyświetlanie tabelki (*tabelka.js*).

W lewej kolumnie drugiego elementu *container-fluid* znajduje się obrazek ilustrujący podgląd aktualnego kroku. W prawej, węższej kolumnie znajduje się lista z polami wyboru, wyświetlana przy pomocy pliku *sync.php*. Plik ten jest wywoływany funkcją zdefiniowaną w pliku *tabelka.js*, wykorzystującego AJAX do dynamicznej zmiany zawartości strony (zapobiega to konieczności odświeżania strony po wybraniu innego elementu z listy). Zasada działania tego pliku zostanie omówiona w następnym podrozdziale.

Każda ze stron konfiguratora zawiera zdefiniowaną w języku PHP zmienną sesyjną *s*, która jest deklarowana przy ładowaniu się strony. Dzięki temu program wie, które dane z bazy powinien pobrać. Jest to numer kroku (od 1 do 7), w którym obecnie jest użytkownik.

### Komunikacja z bazą danych

Bardzo ważną część aplikacji stanowią wymienione wcześniej pliki PHP odpowiedzialne za komunikację z bazą (*connect.php*, *lista.php*, *sync.php*, *database.php*). To one pobierają dane i wyświetlają je na stronie, a także zmieniają zawartość tabelki zapisującej postępy pracy w konfiguratorze.

Aby zmniejszyć rozmiar aplikacji oraz przyspieszyć jej działanie, pliki *lista.php*, *sync.php* oraz *database.php* obsługują wszystkie kroki konfiguratora, jednak są odpowiedzialne za różne funkcje. Wspomniana wcześniej zmienna sesyjna *s*, która zostaje zapamiętana przez serwer, odgrywa tutaj kluczową rolę. Za jej pomocą zaimplementowana została deklaracja *switch*, dzięki której można było wygodnie zapisać siedem przypadków w ramach jednej funkcji. Ta deklaracja używa wartości zmiennej *s*, aby wybrać odpowiedni warunek z tej listy oraz wykonać przypisane do niego instrukcje. Te pliki zostaną omówione bardziej szczegółowo w następnych podrozdziałach.

### Implementacja

Konfigurator składa się z siedmiu stron napisanych w języku PHP - każda z nich odwołuje się do innych plików, m. in. do tych odpowiedzialnych za komunikację z bazą danych. Na początku dokumentu (plik *conf.php*) zawarty jest zapis [Listing 2]:

<?php  
 session\_start();  
 $\_SESSION['s'] = 1;  
?>

Listing 2: Start sesji i zmienna *s* [źródło: opracowanie własne].

Oznacza on start sesji oraz deklarację zmiennej sesyjnej *s* i ma ona wartość 1, ponieważ jest to pierwszy krok konfiguratora. Stąd inne pliki, do których ten się odwołuje wiedzą, który przypadek jest właściwy. Dotyczy to plików *lista.php*, *sync.php* oraz *dataupdate.php*. W każdym z tych plików zawarta jest jedna lub kilka instrukcji warunkowych *switch*, które korzystają z tej zmiennej.

Na pasku nawigacji występuje pewna różnica pomiędzy stroną konfiguratora a pozostałymi stronami aplikacji. Kiedy użytkownik korzysta z głównej części programu, prawdopodobnie nie chciałby, aby postępy jego pracy nie zostały zapisane przez przejście do innej strony poprzez przypadkowe kliknięcie. Zostało to rozwiązane przy pomocy zapisu [Listing 3] wewnątrz listy zawierającej elementy menu:

<li><a href="o\_aplikacji.html" target="\_blank">O aplikacji</a></li>

Listing 3: Otwarcie linku w nowej karcie [źródło: opracowanie własne].

Należy zwrócić uwagę na atrybut *target* - właściwość *\_blank* oznacza, że po kliknięciu napisu „O aplikacji” znajdującego się na pasku, użytkownik zamiast przejść do tej strony w tej samej karcie, otworzy nową. Dzięki temu można swobodnie poruszać się po programie bez obawy o utratę danych. Dodatkowo na stronach znajdują się ikony, które po kliknięciu przenoszą użytkownika do konkretnego kroku. Przyciski te nie działają na stronie głównej oraz pierwszej stronie konfiguratora.

Układ stron konfiguratora został zaprojektowany przy pomocy frameworka Bootstrap. Oferowane przez niego zdefiniowane klasy oraz responsywność doskonale dbają o estetyczny układ elementów na stronie. Jeden z jej fragmentów, czyli drugi *container-fluid*, posiada podział [Listing 4] na trzy kolumny:

<div class="row">  
<div class="col-xs-12 col-sm-10 col-md-8 col-lg-6">

(...)

<div class="col-xs-12 col-sm-10 col-md-3 col-lg-4">

(...)

<div class="col-xs-0 col-sm-0 col-md-1 col-lg-2">

Listing 4: Podział na kolumny według siatki Bootstrapa [źródło: opracowanie własne].

Dwie z nich zawierają podgląd pojazdu oraz listę wyboru. Natomiast trzecia kolumna [Listing 5] utrzymuje właściwy układ tych elementów. Jest to swoisty „prawy margines”:

<div class="col-xs-0 col-sm-0 col-md-1 col-lg-2">

Listing 5: Margines jako odrębna kolumna [źródło: opracowanie własne].

Widoczne zapisy atrybutu *class*, czyli *col-xs*, *col-sm*, *col-md*, *col-lg* odpowiadają za różne rozmiary ekranu urządzenia. To pozwala na dopasowanie zawartości do jego wielkości, dzięki czemu układ strony jest zawsze przejrzysty i czytelny.

W jednej z tych kolumn znajduje się zapis [Listing 6]:

<?php  
 include "lista.php";  
?>

Listing 6: Deklaracja wykorzystania pliku z listą [źródło: opracowanie własne].

Oznacza on odwołanie się do pliku *lista.php*, odpowiedzialnego za wyświetlanie listy wyboru części. Pozwala to zachować porządek w kodzie strony.

Wszystkie strony konfiguratora wyglądają prawie tak samo - różni je jedynie wartość zmiennej *s*. Wcześniej wspomniany plik, czyli *lista.php*, zawiera pierwszą instrukcję *switch*, która wykorzystuje tę zmienną. Zapis [Listing 7] tej deklaracji wygląda następująco:

switch ($\_SESSION['s']){  
 case 1:  
 //Rodzaje nadwozia  
  
 $\_SESSION['nazwatab'] = "rodzaje\_nadwozia";  
 $\_SESSION['nazwapodpowiedz'] = "rodzaj nadwozia";  
 break;  
 case 2:

...

case 7:  
 //Koła  
  
 $\_SESSION['nazwatab'] = "kola";  
 $\_SESSION['nazwapodpowiedz'] = "rodzaj kół";  
 break;  
}

Listing 7: Instrukcja warunkowa *switch* obsługująca kroki konfiguratora w pliku *lista.php*

[źródło: opracowanie własne].

Widać tu siedem przypadków (*case* 1-7) i w każdym z nich następuje inna deklaracja zmiennych sesyjnych *nazwatab* oraz *nazwapodpowiedz*. Pierwsza zmienna przechowuje nazwę kolumny w bazie danych, w której jest zapisana nazwa części, natomiast druga odpowiada za wyświetlanie właściwego tekstu w miejscu podpowiedzi. Dzięki zmieniającej się wartości zmiennej *s*, program wie, który przypadek jest właściwy w danym momencie.

Podstawowym ograniczeniem jest konieczność wybrania rodzaju nadwozia. Jest to baza do budowy każdego samochodu i określa ona, jaki rodzaj części będzie pasował do tworzonego auta. Poprzez warunek [Listing 8] program uniemożliwia użytkownikowi wybranie jakiejkolwiek części bez wybrania rodzaju nadwozia:

$\_SESSION['n'] = 1;

if ($\_SESSION['s']>1) {

$rezultat = $polaczenie->query("SELECT \* FROM `cookies` WHERE `id` = 1");

if ($rezultat->num\_rows!=0) {

} else {

unset($\_SESSION['n']);

}

}

Listing 8: Warunek określający istnienie zmiennej *n* [źródło: opracowanie własne].

Na początku deklarowana jest zmienna sesyjna *n*. Następnie instrukcja warunkowa bada wartość zmiennej przyjmującej wartość kroku. Jeśli użytkownik nie znajduje się na pierwszej stronie konfiguratora, program przy pomocy wysłania zapytania do bazy danych sprawdza, czy istnieje rekord w tabelce *cookies* o numerze *id* równym 1. Jeśli tak, oznacza to, że rodzaj nadwozia został wybrany i aplikacja nie podejmie żadnej akcji, w przeciwnym wypadku zmienna *n* zostaje usunięta.

Pod instrukcją zawarty jest zapis [Listing 9]:

if (isset ($\_SESSION['n'])) {

...

} else {

echo '<span style="color: red;">Błąd: Nie wybrano rodzaju nadwozia!</span>';

}

Listing 9: Instrukcja uniemożliwiająca wybranie części bez określenia rodzaju nadwozia

[źródło: opracowanie własne].

Warunek ten sprawdza, czy została utworzona zmienna *n*. Jeśli taka zmienna istnieje, to program wykonuje dalsze instrukcje. W przeciwnym wypadku wypisuje tekst błędu zamiast listy wyboru (nie można wybrać części).

Po tych deklaracjach znajduje się formularz [Listing 10], w którym następnie umieszczono listę wyboru:

echo "<form action = 'dataupdate.php' method = 'post'>";  
echo "<select id = 'pole' name = 'pole\_wyboru' onchange = 'showPart(this.value)'>";  
echo "<option value = '0' selected>" . "Wybierz $nazwapodpowiedz" . "</option>";

Listing 10: Formularz z listą wyboru części [źródło: opracowanie własne].

Jest to formularz znany z języka HTML, a umieszczony w PHP przy pomocy funkcji *echo*. Widać tutaj, że po zatwierdzeniu danych odwołuje się on do pliku *dataupdate.php*, który z kolei odpowiada za przesłanie danych do bazy. Identyfikator *id* tej listy został określony jako „pole”, natomiast atrybut *onchange* ma za zadanie zmieniać zawartość elementu *div* znajdującego się pod listą w zależności od wartości pola. Funkcja *showPart* jest odpowiedzialna za wyświetlanie właściwych danych z bazy o wybranej części i zawiera ją plik *tabelka.js*. Natomiast zapis *this.value* odwołuje się do obecnej wartości wybranego pola. Jest tu również pierwsza opcja o wartości 0, która jest niejako podpowiedzią w samej liście. Zmienna *nazwapodpowiedz* wskazuje, jaki rodzaj części użytkownik może wybrać.

Następnie program łączy się z bazą danych i deklaruje niezbędne zmienne [Listing 11]:

$rezultat = $polaczenie->query("SELECT \* FROM $nazwatab");  
if (!$rezultat) throw new Exception($polaczenie->error);

if ($nazwatab == "karoserie") {

$\_SESSION['nazw'] = $polaczenie->query("SELECT \* FROM `cookies` WHERE `id` = 1");

$\_SESSION['nazw'] = $\_SESSION['nazw']->fetch\_row();

...

}else {

for ($i=1;$i<=$rezultat->num\_rows + 1;$i++){

$rezultat = $polaczenie->query("SELECT \* FROM $nazwatab WHERE `id` = '$i'");

if (!$rezultat) throw new Exception($polaczenie->error);

$wiersz = $rezultat->fetch\_row();

echo "<option value = ".$i.">"."$wiersz[1]"."</option>";

}

}

Listing 11: Deklaracja zmiennej przechowującej nazwy karoserii [źródło: opracowanie własne].

Należy zwrócić uwagę na istotne ograniczenie. W praktyce niemożliwym jest wybranie na przykład rodzaju nadwozia typu sedan, a następnie dobranie karoserii typu coupé. Dlatego zmienne te zostaną wykorzystane później do filtrowania listy przy wyborze karoserii. Na liście wyświetlany będzie tylko element, który pasuje do danego rodzaju nadwozia.

Trudnym zadaniem było napisanie we właściwy sposób pętli, która będzie generowała nowe pola wyboru na podstawie wyników zapytań do bazy danych. Po zaczerpnięciu wiedzy z literatury [[37](#Lis17)], zostało to rozwiązane w ten sposób [Listing 12]:

for ($i=1;$i<=$rezultat->num\_rows + 1;$i++){  
  
 if ($\_SESSION['nazw'][1] == "Coupe"){

$rezultat = $polaczenie->query("SELECT \* FROM $nazwatab WHERE `id` = 1");

if (!$rezultat) throw new Exception($polaczenie->error);

$wiersz = $rezultat->fetch\_row();

echo "<option value = ".$i.">"."$wiersz[1]"."</option>";

} else {

$rezultat = $polaczenie->query("SELECT \* FROM $nazwatab WHERE `id` = 2");

if (!$rezultat) throw new Exception($polaczenie->error);

$wiersz = $rezultat->fetch\_row();

echo "<option value = ".$i.">"."$wiersz[1]"."</option>";

}  
}

...  
echo "</select>";

Listing 12: Pętla *for* odpowiadająca za tworzenie opcji wyboru [źródło: opracowanie własne].

Przywołanym fragmentem kodu jest pętla *for*, która odpowiada za generowanie pól wyboru w liście. Wewnątrz zostały określone pewne instrukcje warunkowe, dzięki którym lista w kroku wyboru karoserii wyświetla tylko te pola, które są właściwe (zależnie od wyboru rodzaju nadwozia). Zmienna *i* została tu zadeklarowana jako swoisty licznik pętli, ale również określa kolejne wartości utworzonych pól. Dzięki temu w kolejnych plikach będzie można się odwołać do wartości takiej opcji, aby zidentyfikować tę konkretną. Dobrze ilustruje to tabelka [Tabela 9] przedstawiona poniżej:

|  |  |  |
| --- | --- | --- |
| 1. ID (0) | Rodzaje\_nadwozia(1) | Obrazek (2) |
| (1) 1 (0) | Coupe (1) | coupe.jpg (2) |
| (2) 2 (0) | Sedan (1) | sedan.jpg (2) |

Tabela 9: Zaznaczenie wartości używanych przez pętlę *for* na przykładzie tabeli z bazy danych. W nawiasach podano numery indeksów - po lewej stronie indeks wiersza, po prawej kolumny

[źródło: opracowanie własne].

Zapytanie w pętli również używa tej zmiennej, lecz w celu ustalenia indeksu wiersza, który baza ma wyświetlić (pogrubione pola tabelki - numer indeksu zgadza się z wartością w kolumnie ID). Z niego zostaje uzyskany jedynie pojedynczy wiersz z kolumny, który następnie jest wypisywany jako opcja poprzez zmienną *wiersz* z indeksem 1. Zmienna ta zapisuje wartości wszystkich kolumn w jednym wierszu, a funkcja *fetch\_row­*, której używa, pozwala na odwołanie się do każdej z nich przy pomocy właściwego indeksu (zaznaczony w tabelce zygzakiem wiersz, łącznie z pogrubionym polem). Są one numerowane od 0, a ponieważ szukana kolumna to druga z kolei, zatem jej indeksem będzie 1. Dzięki temu otrzymuje się pojedynczą wartość z właściwej kolumny. Widać tutaj, że zmienna *i* została użyta aż trzy razy, a wszystkie wartości zgadzają się ze sobą. Pozwala to przyspieszyć działanie programu, zwłaszcza w przypadku istnienia wielu części w bazie, ponieważ program zapamiętuje jedną zmienną zamiast trzech.

Pod listą wyświetlana jest podpowiedź [Listing 13], jeśli wybrane pole ma wartość równą 0 (czyli nie wybrano żadnej części):

echo "<div id = 'podpowiedź'>" . "<h2>" . "Wybierz $nazwapodpowiedz, aby wyświetlić informacje: " . "</h2>" . "</div>";

Listing 13: Podpowiedź wyświetlana pod listą [źródło: opracowanie własne].

Jest to nagłówek *h2* w elemencie *div*. Zastosowano tutaj, tak samo jak w pętli, konkatenację wyrażeń, aby połączyć znaczniki, zmienne i ciągi znaków w jeden tekst, zrozumiały dla interpretera języka PHP. Element *div* posiada identyfikator *id* o nazwie „podpowiedź”. Został on wykorzystany w pliku skryptowym *tabelka.js*, aby wskazać funkcji miejsce, w którym ma pojawić się tabelka z danymi o części. Pełni ona także rolę informacyjną dla użytkownika.

Natomiast pod podpowiedzią znajduje się przycisk [Listing 14]:

echo "<input type = 'submit' value = 'Dalej' onclick = 'sprawdz()' />";

Listing 14: Przycisk „Dalej” [źródło: opracowanie własne].

Po kliknięciu przesyła dane do serwera oraz przechodzi do następnej strony konfiguratora. Przycisk wywołuje również funkcję *sprawdz*, która znajduje się w pliku *błąd.js*. Odpowiada ona za wyświetlanie informacji o błędzie, kiedy użytkownik kliknie „Dalej” w momencie, gdy w liście wybrana jest opcja o wartości 0 (żadna część nie została wybrana).

Aby wyświetlić dane o części w formie tabelki, zarówno plik *lista.php*, jak i każda strona konfiguratora korzystają z funkcji zawartej w *tabelka.js*. Plik ten wykorzystuje technologię AJAX do dynamicznej zmiany zawartości strony, w tym przypadku tekstu na tabelkę i na odwrót. Funkcja [Listing 15] wygląda następująco:

function showPart(str) {  
 var xhttp;  
 if (str === "") {  
 document.getElementById("podpowiedź").innerHTML = "";  
 return;  
 }  
 xhttp = new XMLHttpRequest();  
 xhttp.onreadystatechange = function() {  
 if (this.readyState === 4 && this.status === 200) {  
 document.getElementById("podpowiedź").innerHTML = this.responseText;  
 }  
 };  
 xhttp.open("GET", "sync.php?q="+str, true);  
 xhttp.send();

}

Listing 15: Funkcja *showPart* korzystająca z technologii AJAX [źródło: opracowanie własne].

Wcześniej wspomniana funkcja *showPart* o wyjściowym parametrze *str* używa pliku *sync.php* do pobrania danych z bazy. Najpierw definiowana jest zmienna *xhttp*, następnie instrukcja warunkowa. W przypadku, gdy zmienna *str* jest identyczna z pustym łańcuchem znaków (czyli "") zostaje wyświetlona podpowiedź. Właściwa podpowiedź została zdefiniowana w pliku *lista.php*, natomiast tutaj pełni ona rolę elementu zmienianego przez funkcję *showPart*. Funkcja *return* zwraca wartość zmiennej *str*, co usuwa zawartość elementu o *id* równym „podpowiedz”. Jeśli zmienna *str* nie jest pusta (tj. w każdym innym przypadku), wtedy zostaje wykonana pewna sekwencja kroków. Najpierw zostaje stworzony nowy element *XMLHttpRequest*, a następnie definiowana jest funkcja, która jest gotowa do wywołania, gdy otrzyma odpowiedź od serwera. Serwerem w tym przypadku będzie serwer lokalny, skonfigurowany przy pomocy XAMPP oraz PhpStorm. Następnie zostaje wysłane żądanie do pliku *sync.php*. Występuje tutaj także kolejna zmienna: jest to zmienna *q*, która przybiera wartość aktualnie wybranej opcji z listy. Została ona wykorzystana także w powyższym pliku. Wówczas metodą *GET* poprzez *sync.php* zostaje zmodyfikowany już wcześniej wspomniany element z *id* „podpowiedz”. Zapis „"sync.php?q="+str” oznacza, że funkcja odwołuje się do pliku *sync.php* oraz definiuje parametr *q*, a zmienna *str* podtrzymuje zamianę pola „podpowiedz” (co oznacza, że pole to nie zmieni się, dopóki nie zmieni się parametr *q*).

W przypadku pliku *connect.php* [] wykonywane jest jedynie połączenie z bazą danych:

<?php

$host = "localhost";

$db\_user = "root";

$db\_password = "";

$db\_name = "car\_conf";

Listing 16: Deklaracja zmiennych używanych do połączenia z serwerem

[źródło: opracowanie własne].

Instrukcja ta została zapisana w jednym miejscu, aby nie powtarzać tego samego w kodzie pozostałych. Wówczas wystarczy odwołać się do *connect.php*, aby nawiązać połączenie z bazą. Oczywiście *hostem* jest tutaj serwer lokalny, a użytkownik będzie miał uprawnienia administratora. Nazwa bazy danych to po prostu *car\_conf*.

Plik *sync.php* na początku łączy się z bazą danych przy pomocy wcześniej wspomnianego pliku [Listing 17]:

<?php  
  
include "connect.php";

Listing 17: Deklaracja wykorzystania pliku *connect.php* [źródło: opracowanie własne].

Został on uodporniony na ewentualne błędy spowodowane problemami z połączeniem przy pomocy instrukcji *try…catch* [Listing 18]:

try {  
 $polaczenie = new mysqli($host, $db\_user, $db\_password, $db\_name);  
 if ($polaczenie->connect\_errno != 0) {  
 throw new Exception(mysqli\_connect\_errno());  
 } else {

...

}  
}  
  
catch(Exception $e){  
 echo '<span style="color: red;">Błąd serwera. Przepraszamy za utrudnienia.</span>';  
 echo '<br />Informacja deweloperska: '.$e;  
}  
?>

Listing 18: Instrukcja *try...catch* użyta do łapania wyjątków [źródło: opracowanie własne].

W przypadku wystąpienia takiego błędu, wyświetli się komunikat napisany czerwonym tekstem wraz z informacją dla dewelopera (można go wyłączyć w razie potrzeby przy pomocy funkcji *mysqli\_report*).

Następnie deklarowana jest zmienna *q* [Listing 19], która poprzez funkcję zawartą w pliku *tabelka.js* zapisuje wartość obecnie wybranego pola z listy wyboru (licząc od 0):

$q = intval($\_GET['q']);

Listing 19: Deklaracja zmiennej *q* [źródło: opracowanie własne].

Zmienna ta uzyskiwana jest przy pomocy metody *GET* z wcześniej wspomnianej funkcji *showPart*, która przyjmuje aktualną wartość wybranego pola z listy.

Po tym następuje połączenie z bazą *ajax\_demo* [Listing 20], która umożliwia zmianę elementu na stronie konfiguratora bez potrzeby odświeżania strony:

mysqli\_select\_db($polaczenie, "ajax\_demo");

Listing 20: Połączenie z serwerem przy użyciu technologii AJAX [źródło: opracowanie własne].

Pod nim znajduje się instrukcja *switch* [Listing 21], która w tym przypadku deklaruje zmienne typu *string*, które pojawią się w konkretnym miejscu zapytań do bazy danych:

switch ($\_SESSION['s']) {  
 case 1:  
 //Rodzaje nadwozia  
  
 $nazwakol = "Nazwa\_nadwozia";  
 $nazwaelem = "Typ nadwozia: ";  
 $nazwapodpowiedz = "rodzaj nadwozia";  
 break;  
 case 2:

...

case 7:

//Kola

$nazwakol = "Nazwa\_kol";  
 $nazwaelem = "Rodzaj kół: ";  
 $nazwapodpowiedz = "rodzaj kół";  
 break;  
  
}

Listing 21: Instrukcja warunkowa *switch* w pliku *sync.php* [źródło: opracowanie własne].

Ponownie instrukcja ta posiada siedem przypadków. Zatem w zależności od wartości zmiennej *s* zmieniać się będą zarówno szukana kolumna (zmienna *nazwakol*), jak i nazwa kolumny wyświetlanej tabelki (*nazwaelem*) oraz podpowiedź (*nazwapodpowiedz*).

Definiowana jest również jeszcze jedna zmienna [Listing 22]:

$nazwatab = $\_SESSION['nazwatab'];

Listing 22: Deklaracja zmiennej *nazwatab* [źródło: opracowanie własne].

Jest to zmienna sesyjna uzyskana z poprzedniej instrukcji *switch* znajdującej się w pliku *lista.php*, która przyjmuje nazwę aktualnie używanej kolumny z bazy danych. Tutaj również została ona zapisana pod nazwą *nazwatab*, aby uprościć formułę zapytań języka SQL w programie.

Następnie sprawdzany jest warunek [Listing 23]:

if ($\_SESSION['s'] == 6) {

if ($\_SESSION['nazw'][1] == "Coupe"){

$sql = "SELECT \* FROM $nazwatab WHERE `id` = 1";

} else {

$sql = "SELECT \* FROM $nazwatab WHERE `id` = 2";

}

} else {

$sql = "SELECT \* FROM $nazwatab WHERE `id` = '" . $q . "'";

}

Listing 23: Warunek określający ograniczenie dla wyboru karoserii [źródło: opracowanie własne].

Tutaj również należy wprowadzić ograniczenie dotyczące karoserii. Wartość 6 oznacza dla zmiennej sesyjnej *s* krok wyboru tego elementu. Warunek dotyczy zapytania wysyłanego do bazy danych. Jeśli użytkownik nie wybiera karoserii, numer z kolumny *id* jest wybierany na podstawie wartości zmiennej *q*. W przeciwnym wypadku zostaje wybrana wartość odpowiednia dla wcześniej określonego rodzaju nadwozia.

Jak widać, w warunku tym zawarta jest pewna zmienna *q*. Jest to uzyskana wcześniej przy pomocy metody GET zmienna przyjmująca wartość obecnie wybranej opcji z listy. W tym przypadku podając jako wartość *q*, program uzyska dostęp do krotki, która zawiera informację o dokładnie tej części, która została wybrana przez użytkownika, ponieważ numer z kolumny ID oraz wartość pola zgadzają się ze sobą. Ponadto poprzednia zmienna oferuje również możliwość dostępu do każdej kolumny, zatem aplikacja może poruszać się swobodnie po każdej tabeli w bazie danych, kierując do niej odpowiednie zapytanie [Listing 24]:

$rezultat = mysqli\_query($polaczenie, $sql);

Listing 24: Zmienna przechowująca rezultat zapytania wysłanego do bazy

[źródło: opracowanie własne].

Zmienna *rezultat* przechowuje natomiast wynik tego zapytania (czyli to, co baza danych wyświetliła po wykonaniu wiersza zapisanego jako *sql*). Do jego uzyskania wykorzystuje wcześniej zdefiniowaną zmienną *polaczenie*.

Poniżej znajduje się instrukcja warunkowa [Listing 25]:

if (".$q." <> 0) {

...  
} else {  
 echo "<h2>Wybierz $nazwapodpowiedz, aby wyświetlić informacje: </h2>";  
}

Listing 25: Instrukcja warunkowa dotycząca wyświetlania podpowiedzi pod listą

[źródło: opracowanie własne].

Wewnątrz tej deklaracji zapisano część kodu odpowiedzialna za wyświetlanie tabelki. Jeśli warunek zostanie spełniony, tj. zmienna *q* przybierze wartość różną od 0 (czyli zostanie wybrana część), wtedy pojawia się tabelka, natomiast w przeciwnym razie zostaje wyświetlona podpowiedź. Dzięki zmiennej *nazwapodpowiedz* znajdującej się w instrukcji *switch*, program automatycznie zmienia nazwę części tak, aby podpowiedź pasowała do strony. Pojawia się tylko wtedy, gdy nie została wybrana żadna część z bazy.

Tabelka [Listing 26] wewnątrz tej deklaracji wygląda następująco:

$wiersz = $rezultat->fetch\_row();  
echo "<table>";  
 echo "<tr>";  
 echo "<td>" . "ID Elementu:" . "</td>" . "<td>" . $wiersz[0] . "</td>";  
 echo "</tr>";  
 echo "<tr>";  
 echo "<td>" . $nazwaelem . "</td>" . "<td>" . $wiersz[1] . "</td>";  
 echo "</tr>";  
echo "</table>";  
echo "<br /><br />";  
echo "<img src = \"media/".$wiersz[2]."\" ... >";

Listing 26: Zapis tabelki, która wyświetla się po wybraniu opcji z listy

[źródło: opracowanie własne].

Należy zauważyć, że zastosowano tu konkatenację, aby odpowiednio rozmieścić elementy tabelki i sprawić, że jej układ będzie czytelny. Najpierw deklarowana jest zmienna *wiersz* - jest tam przechowywana macierz 1x4, która odpowiada pojedynczemu wierszowi z tabelki. Wykorzystując wcześniej zdefiniowaną zmienną *rezultat*, która przechowuje wynik zapytania, przy pomocy funkcji *fetch\_row* porządkuje zgromadzone dane i przypisuje każdemu elementowi tej macierzy odpowiedni indeks. Zatem indeks 0 będzie oznaczał pierwszą kolumnę tego wiersza, indeks 1 drugą i tak dalej. Jak opisano to wcześniej, tabelka w bazie zawiera informacje o wszystkich częściach z danego rodzaju, a zmienna *wiersz* przechowuje wszystkie, ale tylko o jednej części (zmienna *q* określa, o której części informacje należy wyświetlić). Układ wyświetlanej na stronie tabelki jest pionowy, zatem posiada ona jedynie dwie kolumny: w jednej zawarta jest nazwa kolumny z bazy danych, w drugim jej wartość. Przy pomocy odpowiedniej wartości indeksu wyświetlana jest jedynie odpowiednia informacja w danym miejscu, czyli pojedyncza kolumna zmiennej *wiersz*. W trzeciej kolumnie przechowywana jest nazwa pliku: jest to obrazek, który daną część ilustruje. Aby zwiększyć czytelność układu, obrazek ten znajduje się pod tabelką, w odpowiednim odstępie.

Na koniec zaleca się zamknięcie połączenia [Listing 27] z bazą danych:

$rezultat->close();  
mysqli\_close($polaczenie);

Listing 27: Zamknięcie połączenia z bazą danych [źródło: opracowanie własne].

Dzięki temu można uniknąć błędów związanych z serwerem.

Plik *dataupdate.php* wygląda podobnie, z tą różnicą, że deklaracja *switch* [Listing 28] definiuje zmienne jako nazwy konkretnych wartości z bazy danych:

$k = $\_SESSION['s'];  
$p = $\_POST['pole\_wyboru'];  
switch ($k) {  
  
 //krok 1: rodzaje\_nadwozia  
  
 case 1:  
  
 //Lista w kroku 1  
  
 switch ($p) {  
  
 case 0:  
 header('Location: conf.php');  
 exit();  
 break;  
  
 case 1:  
 $nazwaczesci = "Coupe";  
 break;  
  
 case 2:  
 $nazwaczesci = "Sedan";  
 break;  
 }  
 break;

...

//krok 7: kola  
  
 case 7:  
  
 //Lista w kroku 7  
  
 switch ($p) {  
  
 case 0:  
 $nazwaczesci = "";  
 header('Location: conf7.php');  
 exit();  
 break;  
  
 case 1:  
 $nazwaczesci = "Koła1";  
 break;  
  
 case 2:  
 $nazwaczesci = "Koła2";  
 break;  
 }  
 break;  
}

Listing 28: Instrukcja warunkowa *switch* w pliku *dataupdate.php* [źródło: opracowanie własne].

Jest to dość długa instrukcja, ale ponownie obsługuje ona siedem przypadków. Dla ułatwienia zapisu, zmienna sesyjna *s* została zadeklarowana jako zmienna *k*, natomiast przy pomocy metody POST uzyskano wartość opcji z listy wyboru i zapisano jako zmienną *p* (odpowiednik wcześniejszej zmiennej *q*). Ponieważ istnieje niewiele opcji wyboru (w każdej tabelce jest jedynie dwie części), wewnątrz każdego przypadku została określona kolejna instrukcja typu *switch*, tym razem warunkowana wartością zmiennej *p*. W przypadku istnienia większej ilości części, wygodniej byłoby zamiast takiej instrukcji zadeklarować zmienną *wiersz* i posłużyć się indeksem przy zapisie zapytania.

Następnym elementem jest skierowanie odpowiedniego zapytania [Listing 29] do bazy:

$sql = $polaczenie->query("SELECT \* FROM `cookies` WHERE `id` = '$k'");  
if ($sql) {  
 if($sql->num\_rows == 0){  
 $polaczenie->query("INSERT INTO `cookies` (`id`, `wybrane\_czesci`) VALUES ('$k', '$nazwaczesci')");  
 } else {  
 $polaczenie->query("UPDATE `cookies` SET `wybrane\_czesci` = '$nazwaczesci' WHERE `cookies`.`ID` = '$k'");  
 }

...

} else {  
 throw new Exception($polaczenie->error);  
}

Listing 29: Instrukcja warunkowa kierująca zapytania do bazy danych

[źródło: opracowanie własne].

Najpierw deklarowana jest zmienna *sql* - przechowuje ona wynik zapytania o wszystkie elementy z tabeli „cookies”. Następnie po spełnieniu odpowiedniego warunku podejmuje kolejne działania. Zapis „if ($sql)” oznacza „jeśli połączenie z bazą jest udane”. Po jego weryfikacji program sprawdza, czy w tabeli istnieje zero wierszy o konkretnym *id* (czyli sprawdza, czy została w tym miejscu zapisana jakaś część). Jeśli tak, to wtedy zostaje skierowane polecenie *insert* i zdefiniowane dwie wartości w odpowiednich dla nich kolumnach. Jeśli nie, to wykonane zostaje polecenie *update*, czyli aktualizacja istniejącego wiersza o nową wartość, która jest nazwą części.

Następnie sprawdzany jest warunek [Listing 30]:

if ($\_SESSION['s'] == 1){

$\_SESSION['car'] = $nazwaczesci;

}

Listing 30: Deklaracja zmiennej przechowującej wybrany rodzaj nadwozia

[źródło: opracowanie własne].

Po tym wszystkim program przechodzi przez jeszcze jedną instrukcję *switch* [Listing 31]:

switch ($k){  
 case 1:  
 header('Location: conf2.php');  
 break;  
 case 2:

...

case 7:  
 header('Location: sum.php');  
 break;  
}

Listing 31: Instrukcja *switch* odpowiadająca za przekierowanie użytkownika do kolejnej strony [źródło: opracowanie własne].

Tutaj następuje przekierowanie do kolejnej strony, zależnie znów od wartości zmiennej *k*. Jest to zapisane jako osobna instrukcja, aby nie powtarzać kierowania zapytania do bazy danych w każdym przypadku, ponieważ i tak odpowiednie zmienne dbają o to, by zapytanie było właściwe. Po wszystkim oczywiście połączenie z bazą zostaje zamknięte.

Uwagę może zwracać ostatni warunek - przekierowanie do pliku *sum.php*. Jest to strona konfiguratora wyświetlająca zawartość tabelki „cookies” przy pomocy odpowiedniego zapytania do bazy danych, czyli podsumowanie działań użytkownika w konfiguratorze.

Najpierw wyświetlany jest obrazek. Zależnie od wyboru użytkownika, określany jest na podstawie instrukcji [Listing 32]:

<?php

if (isset($\_SESSION['car'])){

if ($\_SESSION['car'] == "Coupe") {

echo '<img src = "media/coupe.jpg" ... >';

}

if ($\_SESSION['car'] == "Sedan") {

echo '<img src = "media/sedan.jpg" ... >';

}

} else {

echo '<img src = "media/step6.png" ... >';

}

?>

Listing 32: Instrukcja określająca wyświetlanie właściwego obrazka samochodu

[źródło: opracowanie własne].

Jeśli użytkownik wybrał jeden z dostępnych rodzajów nadwozi, to program wyświetli odpowiedni obrazek. W przeciwnym wypadku pojawi się standardowy obrazek roboczy.

Po zastosowaniu instrukcji *try...catch* oraz wstępnym połączeniu z bazą, zostaje zdefiniowana zmienna [Listing 33]:

$rezultat = $polaczenie->query("SELECT \* FROM `cookies`");

Listing 33: Zmienna przechowująca zawartość tabelki „cookies” [źródło: opracowanie własne].

Zapisuje ona rezultat zapytania o wszystkie wiersze z tabeli „cookies”. Następnie zostaje utworzona tabela [Listing 34] wraz z nagłówkami kolumn, w takim samym układzie, jak ta wyświetlająca dane o częściach:

echo "<table>  
 <tr>  
 <th>ID Elementu:</th>  
 <th>Nazwa elementu:</th>  
 </tr>";

...

echo "</table>";

Listing 34: Tabelka podsumowująca konfigurację samochodu [źródło: opracowanie własne].

Wewnątrz znajduje się pętla *while* [Listing 35], co usprawnia generowanie kolejnych wierszy, maksymalnie do siedmiu (tyle, ile jest części w pojedynczej konfiguracji samochodu):

while ($rzad = mysqli\_fetch\_array($rezultat)) {  
  
 echo "<tr>";  
 echo "<td>" . $rzad['ID'] . "</td>";  
 echo "<td>" . $rzad['Wybrane\_czesci'] . "</td>";  
 echo "</tr>";  
  
}

Listing 35: Pętla *while* dodająca każdą wybraną część do tabelki [źródło: opracowanie własne].

Jako warunek tej pętli określono zmienną *rząd*, która zwraca kolejne wiersze z tabelki „cookies”. Dzięki funkcji *mysqli\_fetch\_array* możliwe jest wywołanie konkretnych kolumn z danego wiersza po nazwie kolumny istniejącej w bazie danych. Pierwsza z nich to „ID”, czyli identyfikator części - na przykład dla rodzaju nadwozia będzie to 1, a dla kół ostatni, czyli 7. Natomiast druga, czyli „Wybrane\_czesci”, to nazwa każdej części importowana z tabelki, w której ta część się znajduje. Taki zapis wystarczy, aby za każdym razem stworzyć faktyczne podsumowanie działań użytkownika, niezależnie od tego, czy wybrał on część w każdym kroku konfiguratora, czy przeszedł od razu do tej strony. Na koniec następuje zamknięcie połączenia z bazą.

Ponieważ jest to wszystko zrealizowane na osobnej stronie, *sum.php* posiada także układ oraz wszystkie pliki JavaScript tak, jak inne strony konfiguratora. Oznacza to, że pod tabelką znajdują się również przyciski posiadające odpowiednie funkcje.

Przycisk [Listing 36] odpowiadający za zapisanie istniejącej konfiguracji wygląda tak:

<input type="submit" value="Zapisz" />

Listing 36: Przycisk „Zapisz” [źródło: opracowanie własne].

Natomiast przycisk [Listing 37], przy pomocy którego można wrócić do strony głównej, jest opisany w ten sposób:

<input type="submit" value="Wróć do strony głownej" onclick="wroc()"/>

Listing 37: Przycisk „Wróć do strony głównej” [źródło: opracowanie własne].

Widać tutaj funkcję *wroc* wywoływaną poprzez zdarzenie *onclick* - jest ona związana z plikiem *powrót.js*.

Warto się przyjrzeć również plikom pomocniczym. Są to w większości pliki posiadające pojedyncze funkcje, do których te strony się odwołują. Odpowiadają one za obsługę błędów czy przekierowanie użytkownika do właściwej strony.

Pierwszy z nich to *błąd.js*. Jak sama nazwa wskazuje, zawiera on funkcję [Listing 38], która zapobiega nieodpowiednim działaniom ze strony użytkownika:

function sprawdz() {  
 var wartosc = document.getElementById("pole").value;  
 if (wartosc === "0"){  
 document.getElementById("blad").innerHTML = "<div id=\"blad\" class=\"alert alert-danger\">Błąd: Nie wybrano żadnej części!</div>";  
 }  
}

Listing 38: Funkcja sprawdzająca, czy wybrana została część z listy [źródło: opracowanie własne].

Funkcja nazywa się *sprawdz* i jest przypisana do przycisku „Dalej” znajdującym się na stronach od *conf.php* do *conf7.php*. Zdefiniowano tu zmienną *wartość*, która odwołuje się do elementu o *id* równym „pole”, czyli do listy wyboru części. Kiedy wybrana opcja z listy ma wartość 0 (czyli kiedy nie została wybrana żadna część), funkcja zmieni element *div* o *id* równym „blad”, który znajduje się pod tekstem oznaczającym podpowiedź. Zostanie wtedy wyświetlony komunikat o błędzie: „Błąd: Nie wybrano żadnej części!”, umieszczony na czerwonym tle. Informuje on użytkownika, że zanim przejdzie do następnej strony, powinien najpierw wybrać część z listy.

Kolejnym plikiem jest *podsumowanie.js*. Zawarta tu funkcja [Listing 39] pozwala przejść do podsumowania:

function sum() {  
 window.location.replace("sum.php");  
}

Listing 39: Funkcja przekierowująca do strony z podsumowaniem [źródło: opracowanie własne].

Odwołuje się ona do pliku *sum.php* i jest przypisana do przycisku „Przejdź do podsumowania” na tych samych stronach konfiguratora.

Przycisk „Zapisz” pozwala zapisać konfigurację pojazdu w formacie pliku .pdf. Funkcja *zapisz* znajduje się w pliku *zapisz.js* [Listing 40] i wygląda następująco:

function zapisz() {

var report = new Stimulsoft.Report.StiReport();

report.loadFile("reports/Report.mrt");

report.renderAsync(function () {

var data = report.exportDocument(Stimulsoft.Report.StiExportFormat.Pdf);

Object.saveAs(data, "Konfiguracja.pdf", "application/pdf");

})

}

Listing 40: Plik zapisujący konfigurację [źródło: opracowanie własne].

Jak widać, został tu użyty program Stimulsoft Designer do zaplanowania graficznego układu pliku. Odbywa się to poprzez połączenie z bazą danych i pobranie informacji z tabelki *cookies*.

Ostatnim plikiem, który jest wykorzystany przez te strony, jest *powrót.js*. Jest on przypisany do przycisku na stronie podsumowania - „Przejdź do strony głównej”. Przycisk ten obsługuje funkcja *wroc* [Listing 41]:

function wroc() {  
  
 var xml = new XMLHttpRequest();  
 if (confirm("Zamierzasz wrócić do strony głównej. Czy na pewno chcesz to zrobić? (Wprowadzone dane będą wyczyszczone!)") === true) {  
 xml.open("GET", "resetcookies.php");  
 xml.send();  
 window.location.replace("index.html");  
 }  
}

Listing 41: Funkcja używająca pliku *resetcookies.php* [źródło: opracowanie własne].

Ponieważ wymagana jest tutaj komunikacja z bazą, aby zmodyfikować zawartość, odbywa się to przy pomocy narzędzia AJAX oraz pliku *resetcookies.php*. Podobnie jak w przypadku wyświetlania tabelki na stronie, jest tutaj deklarowane nowe żądanie *XMLHttpRequest*. Jednak warunkiem jest wybór opcji wyskakującego okna dialogowego. Kiedy użytkownik zatwierdzi przyciskiem „Ok”, wtedy zostaje otwarty plik *resetcookies.php* oraz następuje przekierowanie do strony głównej. Jeśli zostanie wybrana druga opcja, bądź okno zostanie zamknięte krzyżykiem, warunek nie jest spełniony i funkcja nie jest realizowana.

Wspomniany wcześniej plik *resetcookies.php* [Listing 42] wygląda następująco:

mysqli\_select\_db($polaczenie, "ajax\_demo");  
$sql = $polaczenie->query('SELECT \* FROM `cookies`');  
if ($sql){  
 $polaczenie->query("DELETE FROM `cookies` WHERE `cookies`.ID < 10");

$\_SESSION['car'] = NULL;  
 mysqli\_close($polaczenie);  
} else {  
 throw new Exception($polaczenie->error);  
}

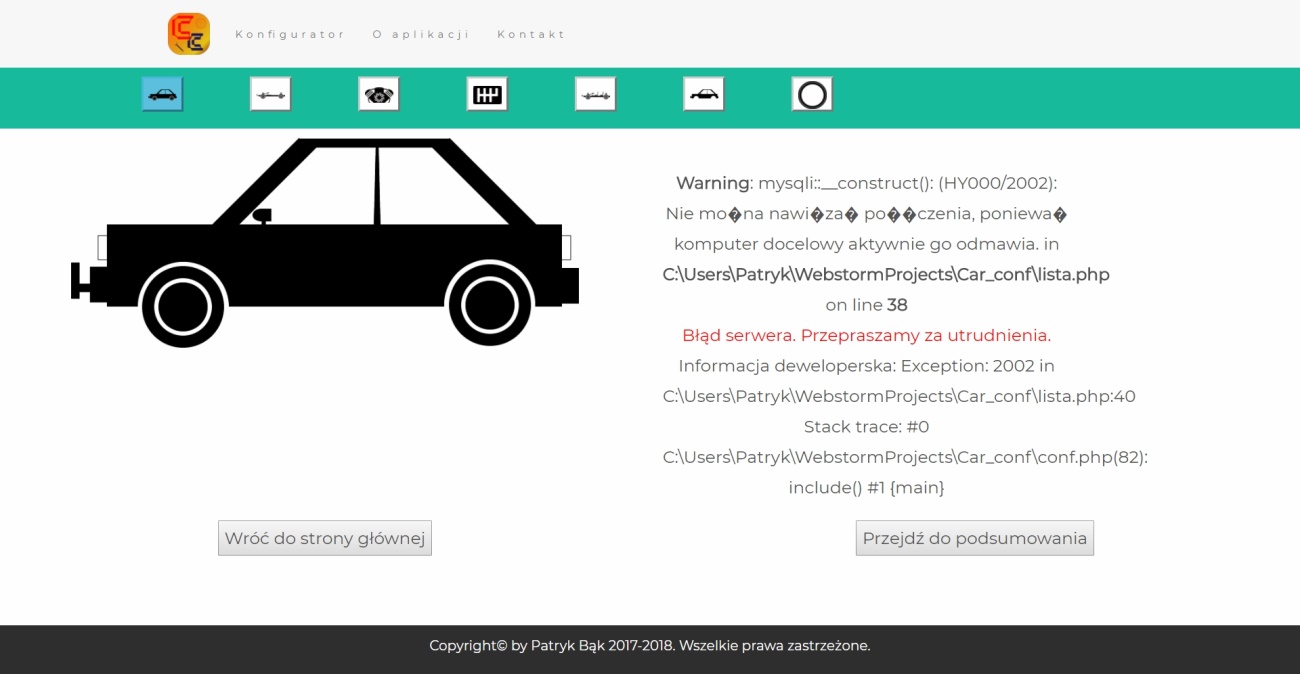
Listing 42: Plik *resetcookies.php* usuwający dane z tabelki „cookies” [źródło: opracowanie własne].

Po połączeniu się z bazą danych i deklaracji zmiennej *polaczenie* (tak samo jak w innych plikach PHP), zostaje zdefiniowana instrukcja warunkowa. Ponieważ plik jest używany tylko w przypadku realizacji funkcji *wroc*, nie ma tu warunku na wysyłanie zapytania do bazy. Zatem po odwołaniu się do tego pliku, usuwana jest zawsze cała zawartość tabelki „cookies”. Zapis oznacza skasowanie wszystkich wierszy o *id* mniejszym od 10. Tak naprawdę mogłaby to być dowolna liczba naturalna większa od 7, ponieważ w tabelce tej może znajdować się maksymalnie siedem rekordów. Zostaje tu również zmieniona wartość zmiennej sesyjnej *car*. Od tej pory zmienna ta jest pusta (NULL), co powoduje, że resetuje się także obrazek wyświetlany w pliku *sum.php*.

# TESTOWANIE

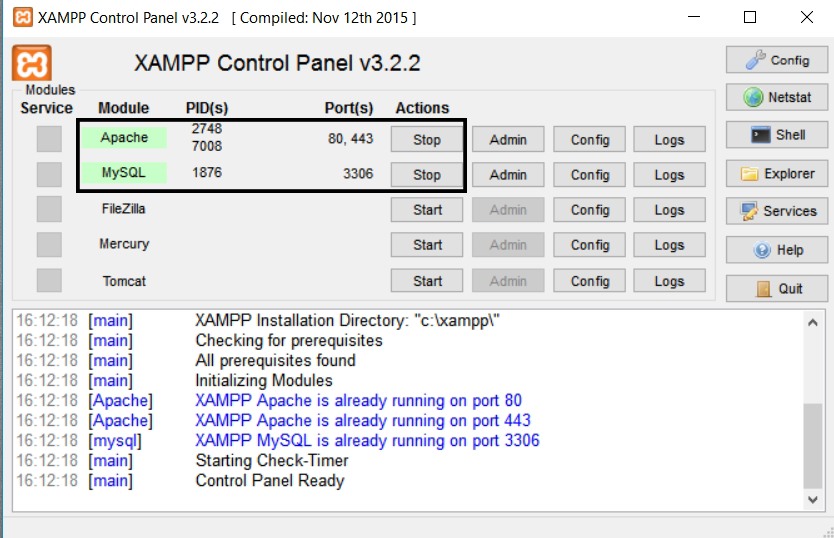
Do przeprowadzenia testów użyto pakietu XAMPP (interpreter PHP). Wykorzystano aplikację PhpStorm jako serwer lokalny, a przeglądarkę Google Chrome do wyświetlenia stron konfiguratora oraz testów pracy w różnych rozdzielczościach.

Zanim aplikacja zostanie przetestowana, należy uruchomić panel kontrolny pakietu XAMPP, ponieważ wymaga ona serwera lokalnego oraz połączenia z bazą danych MySQL. W przeciwnym wypadku wyświetli się komunikat []:



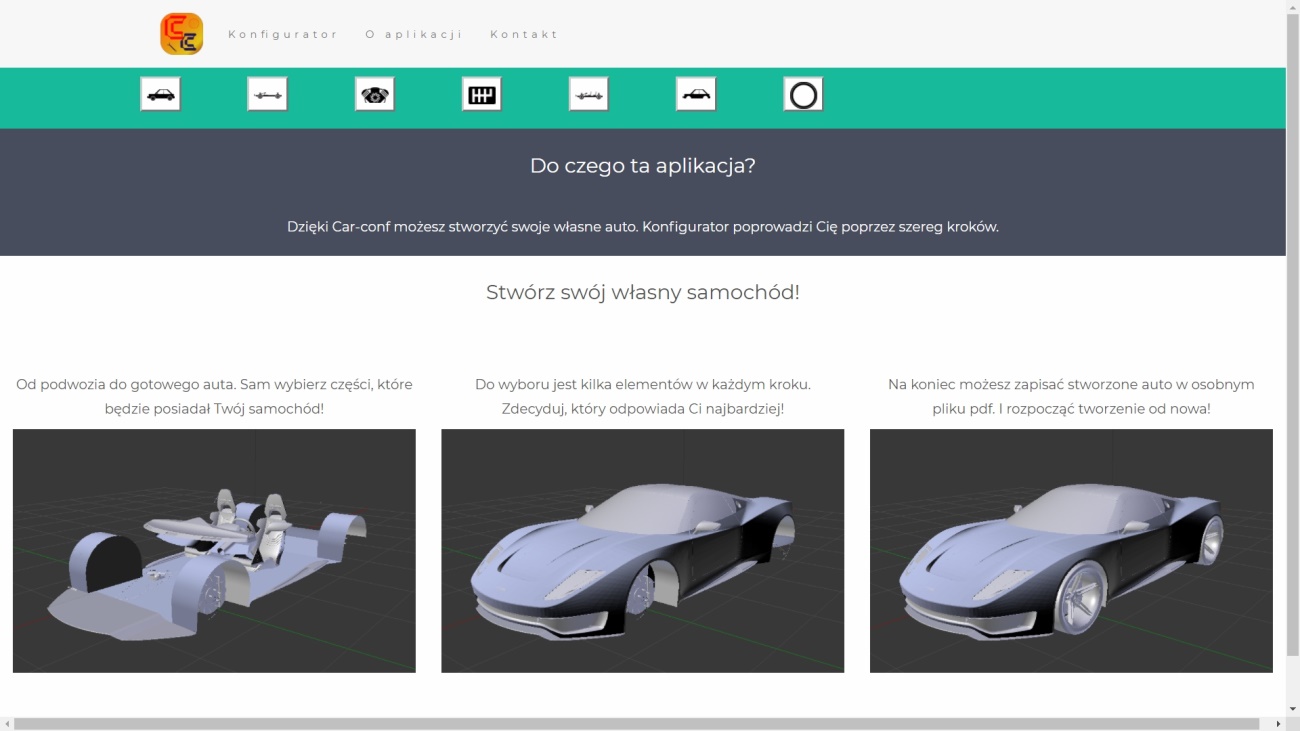
Rys. 4: Błąd spowodowany wyłączonym serwerem [źródło: opracowanie własne].

Po uruchomieniu panelu wystarczy włączyć serwer Apache oraz obsługę MySQL. Zaznaczono te opcje czarnym prostokątem na obrazku poniżej []:



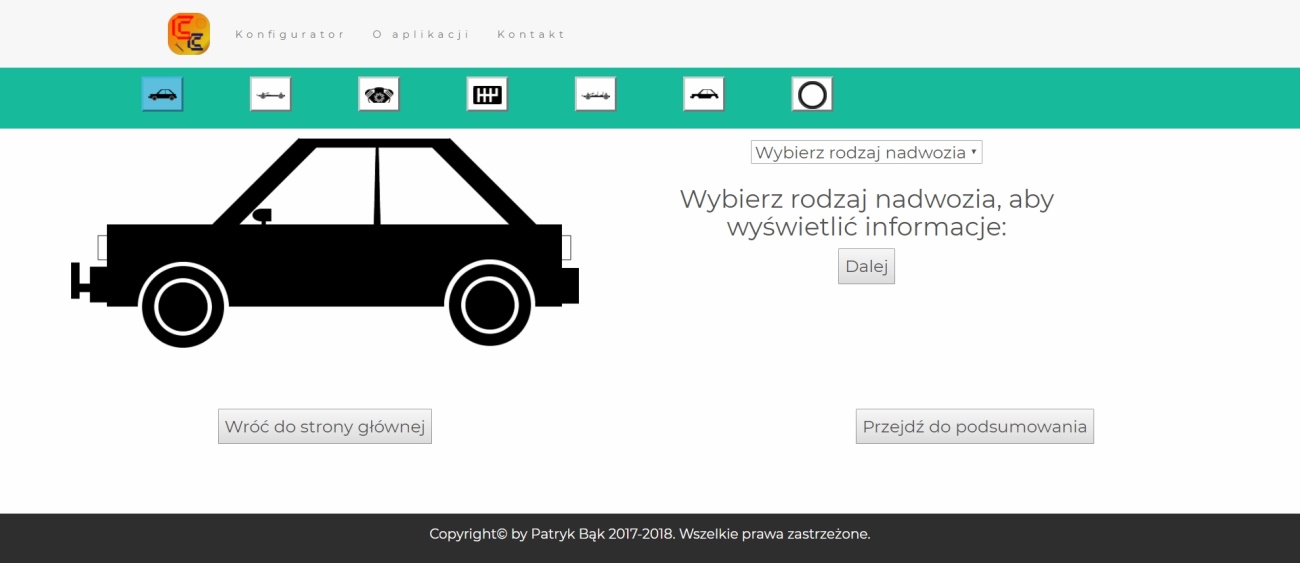
Rys. 5: Panel kontrolny pakietu XAMPP [źródło: opracowanie własne].

Następnie po uruchomieniu aplikacji PhpStorm, gdzie jest skonfigurowany serwer lokalny, należy otworzyć plik *index.html* zawierający stronę główną. Plik ten wyświetla się w domyślnej przeglądarce []:



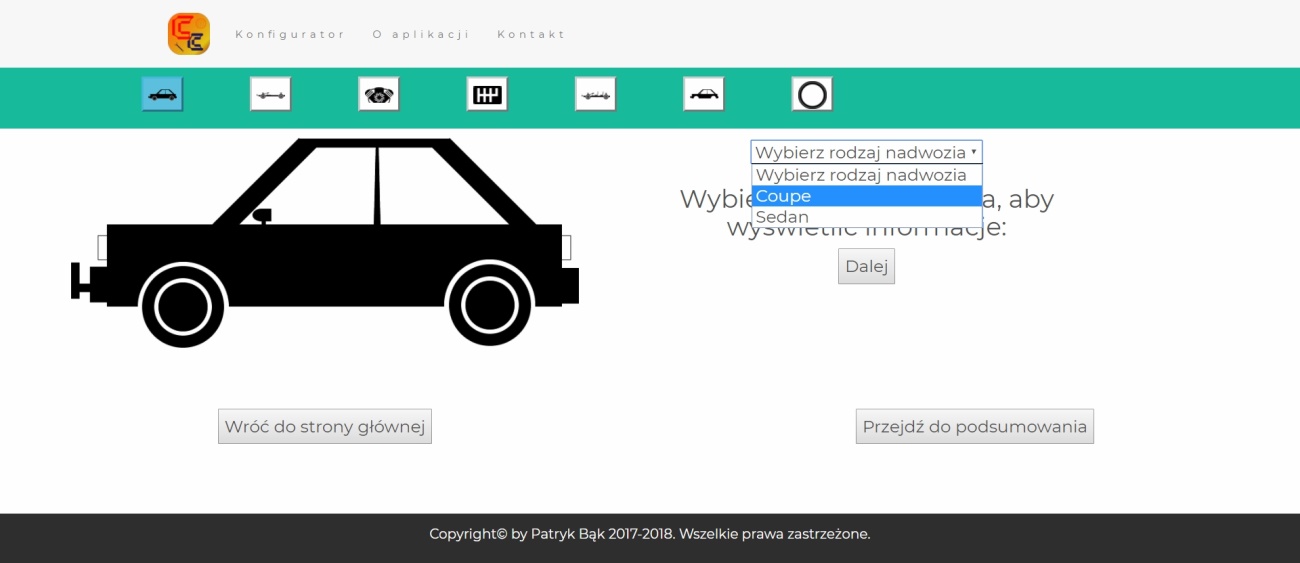
Rys. 6: Strona główna aplikacji Car-conf [źródło: opracowanie własne].

Teraz, gdy serwer już działa, można bez przeszkód uruchomić konfigurator. Klikając na napis „Konfigurator” strona przekieruje użytkownika do pliku *conf.php*, czyli pierwszej strony korzystającej z bazy danych []:



Rys. 7: Pierwsza strona konfiguratora [źródło: opracowanie własne].

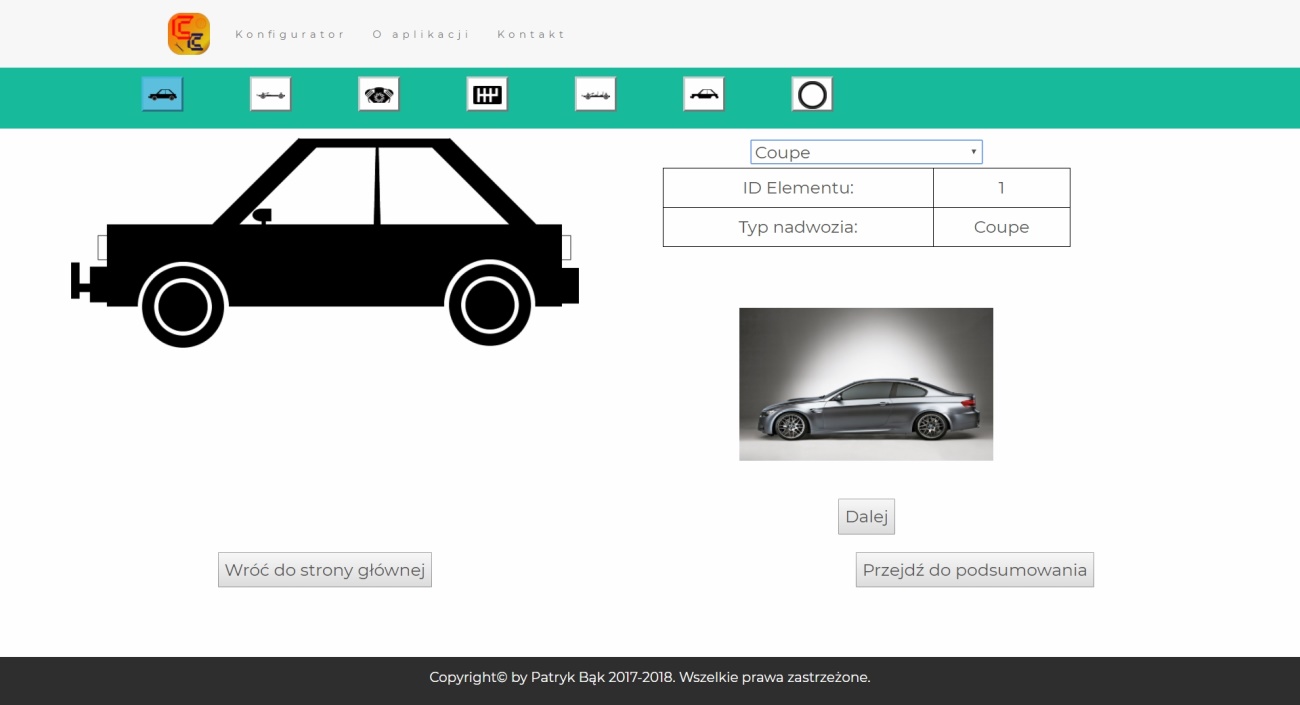
Widać tutaj podgląd po lewej stronie oraz listę wyboru po prawej []:



Rys. 8: Lista wyboru wyświetla wszystkie części z bazy danych z tabelki „Rodzaje\_nadwozia”

[źródło: opracowanie własne].

Pod listą znajduje się podpowiedź, która po wybraniu części zmienia się w tabelkę wyświetlającą dane o wybranym elemencie z bazy []:



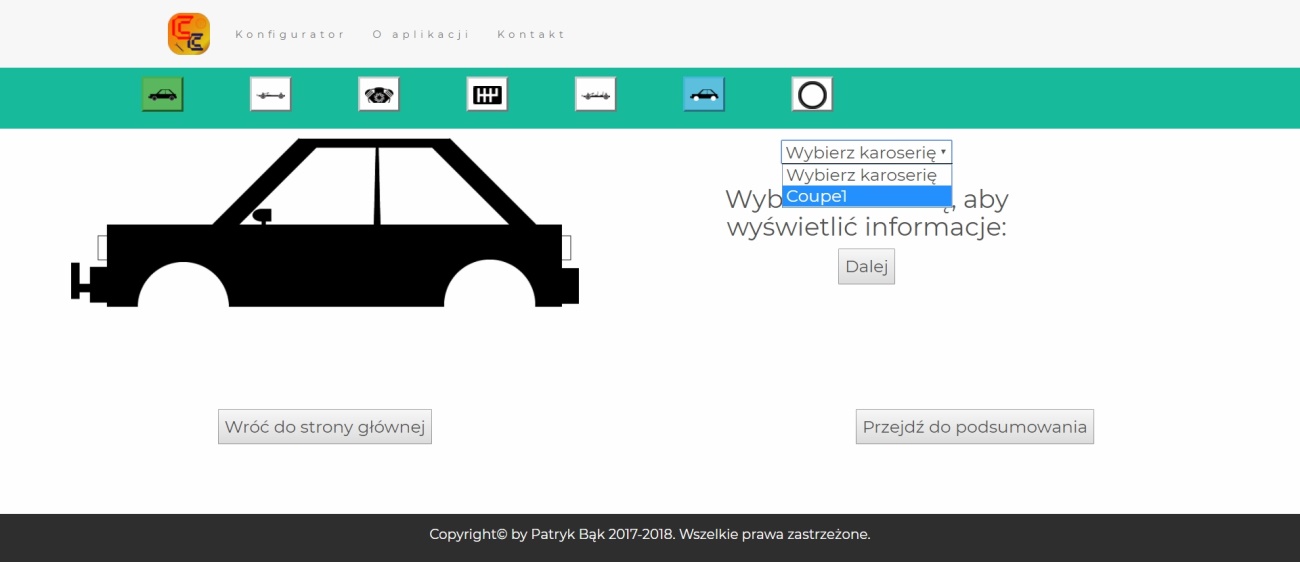
Rys. 9: Informacje o wybranej części wyświetlane są pod listą - jest tam również obrazek przedstawiający wygląd elementu [źródło: opracowanie własne].

Po kliknięciu przycisku „Dalej” pojawia się kolejna strona, na której z kolei należy wybrać podwozie samochodu []:



Rys. 10: Druga strona konfiguratora

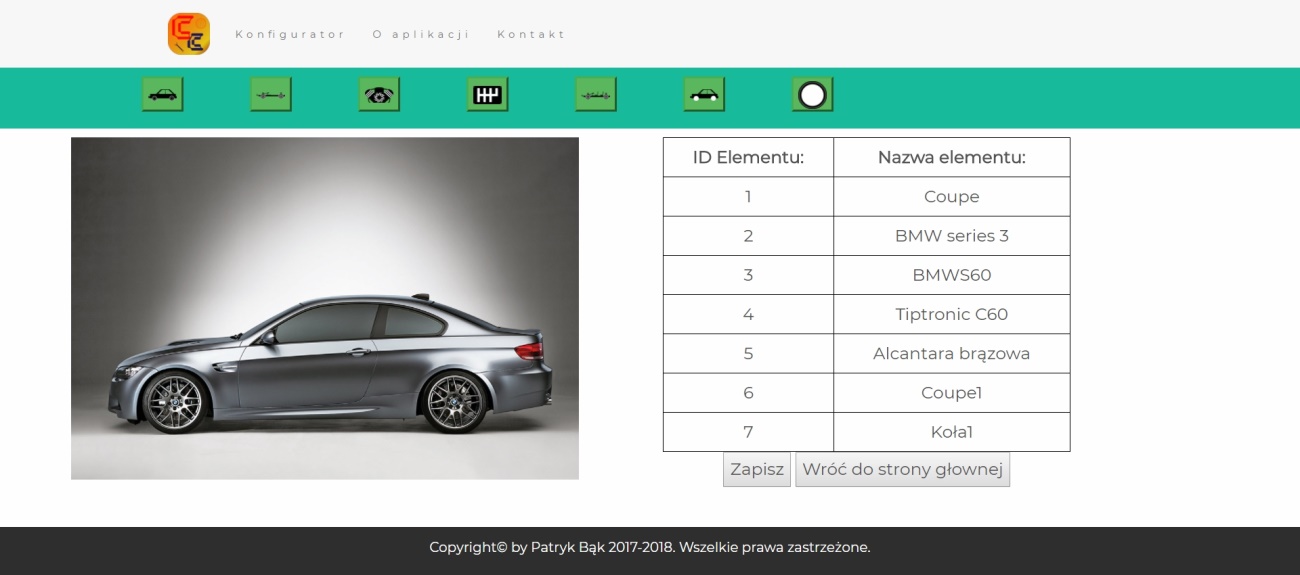
Ponieważ na każdej kolejnej stronie postępuje się podobnie przy wyborze części, ten fragment zostanie pominięty. Jedynie przy wyborze karoserii [] występuje pewne ograniczenie:



Rys. 11: Ograniczenie wyboru karoserii wiąże się z wyborem rodzaju nadwozia

[źródło: opracowanie własne].

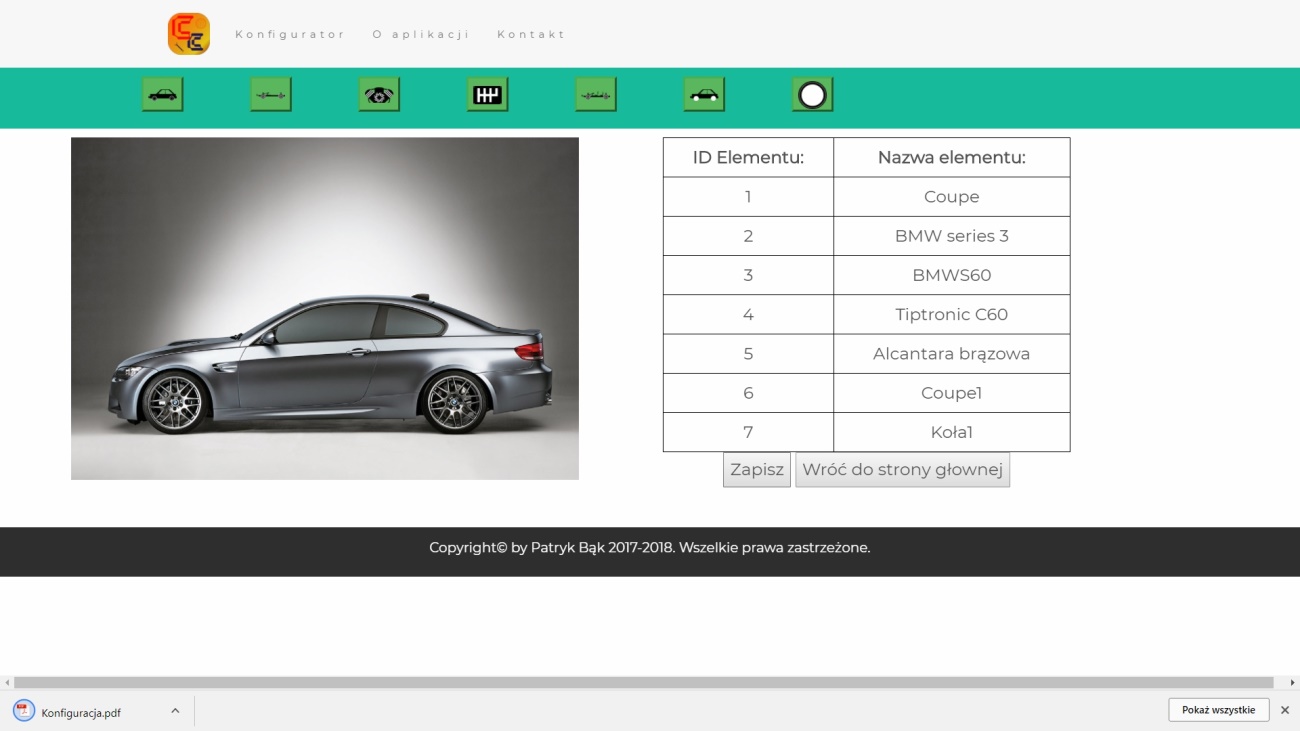
Po kliknięciu „Dalej” na siódmej stronie konfiguratora, aplikacja wyświetla podsumowanie []:



Rys. 12: Przykładowa konfiguracja samochodu pokazana na stronie podsumowania

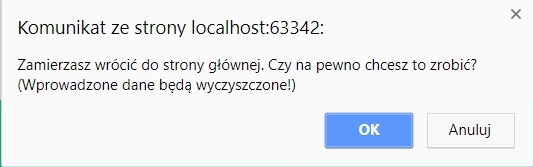
[źródło: opracowanie własne].

Można tutaj zapisać konfigurację bądź wrócić do strony głównej aplikacji. Po kliknięciu przycisku „Zapisz” zostanie utworzony plik w formacie .pdf z danymi []:



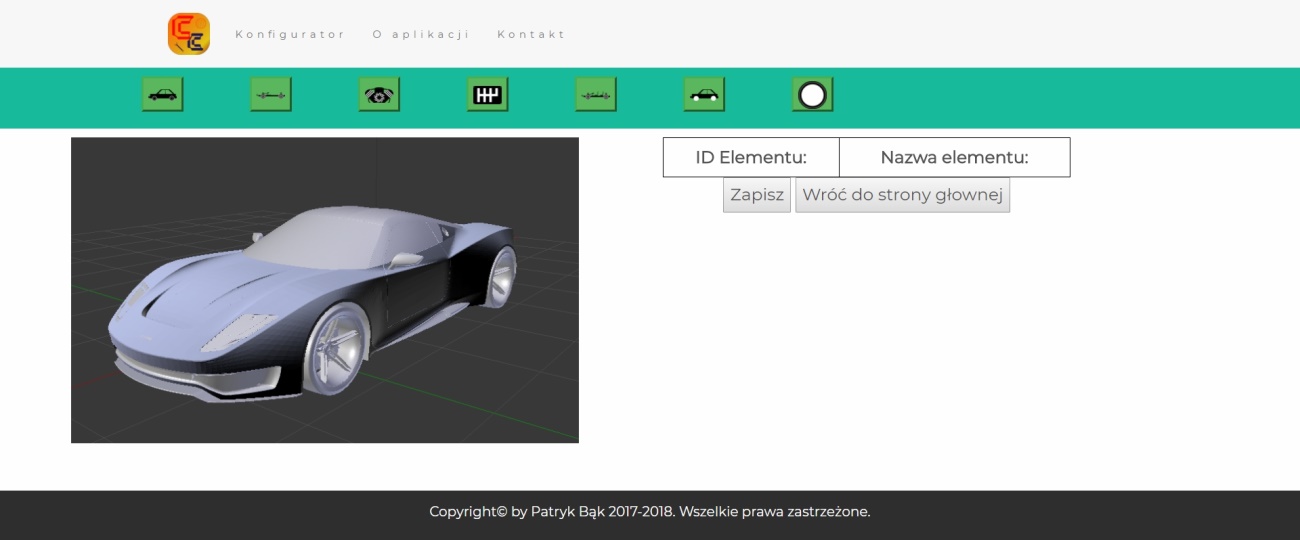
Rys. 13: Zapis do pliku .pdf [źródło: opracowanie własne].

W drugim przypadku strona wyświetli komunikat, czy użytkownik na pewno chce to zrobić oraz informuje go, że wszystkie te dane zostaną usunięte []:



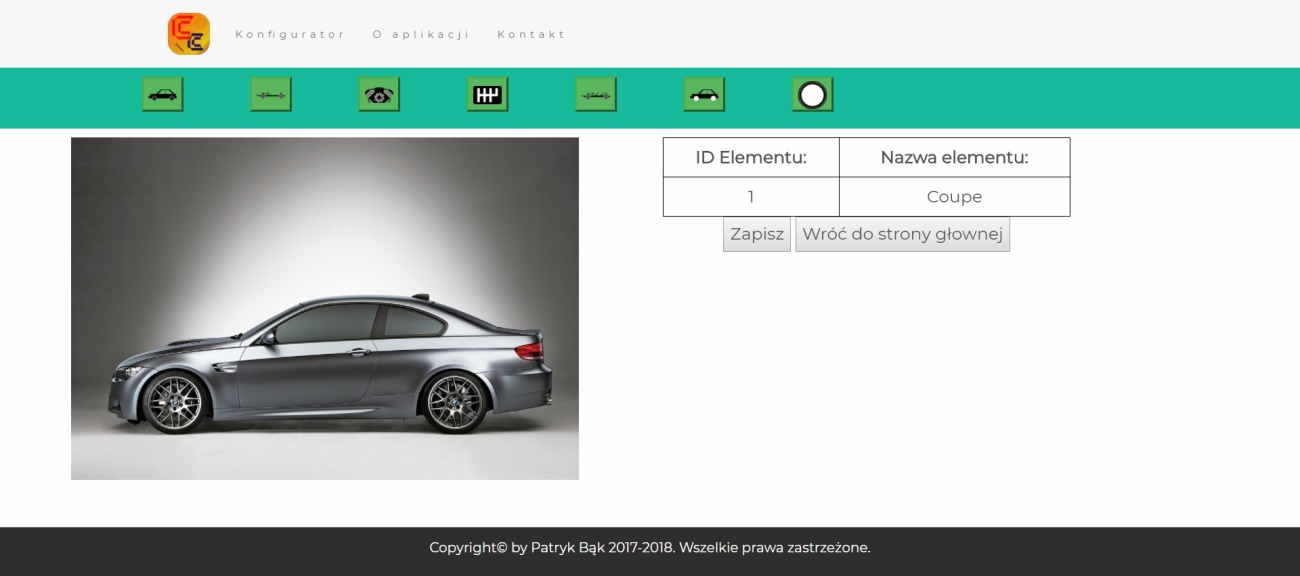
Rys. 14: Komunikat informujący użytkownika o usunięciu danych w przypadku powrotu do strony głównej [źródło: opracowanie własne].

Po kliknięciu „OK” strona usuwa dane oraz przenosi użytkownika do strony głównej. Dzięki przyciskowi „Przejdź do podsumowania”, po ponownym wejściu na stronę konfiguratora można się przekonać, że informacje rzeczywiście zostały skasowane z bazy danych []:



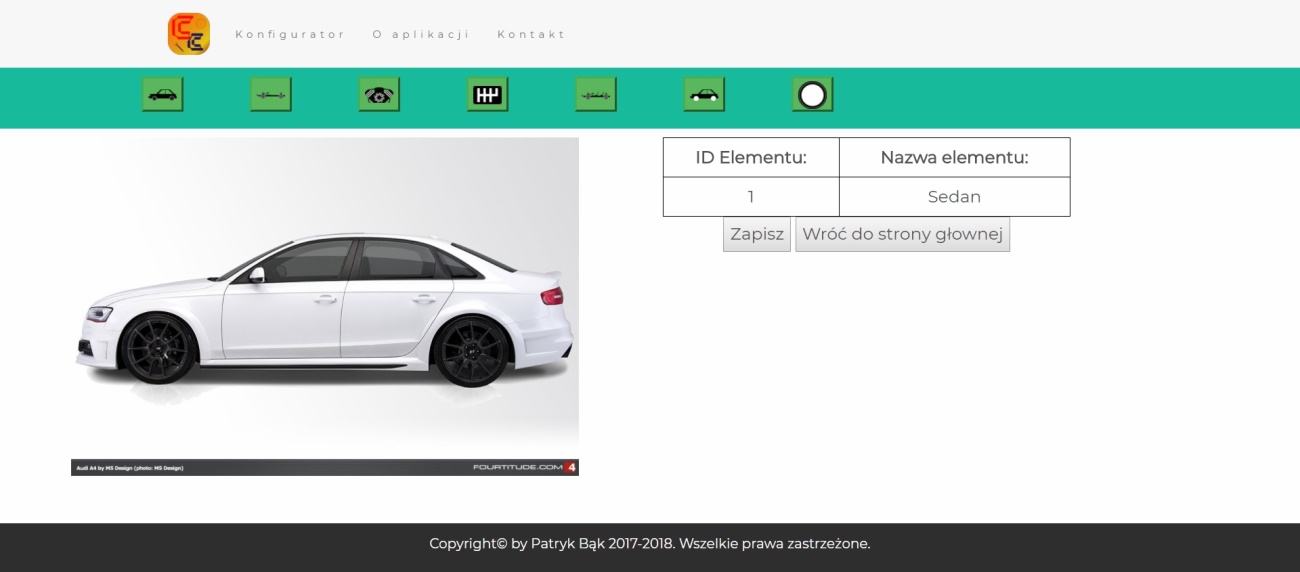
Rys. 15: Po ponownym wejściu do konfiguratora oraz kliknięciu przycisku „Przejdź do podsumowania”, strona wyświetli pustą tabelkę [źródło: opracowanie własne].

Kiedy zostanie wybrana przynajmniej jedna część (na przykład rodzaj nadwozia) i użytkownik przejdzie do podsumowania, wtedy zostanie ona wyświetlona w tabelce na stronie []:



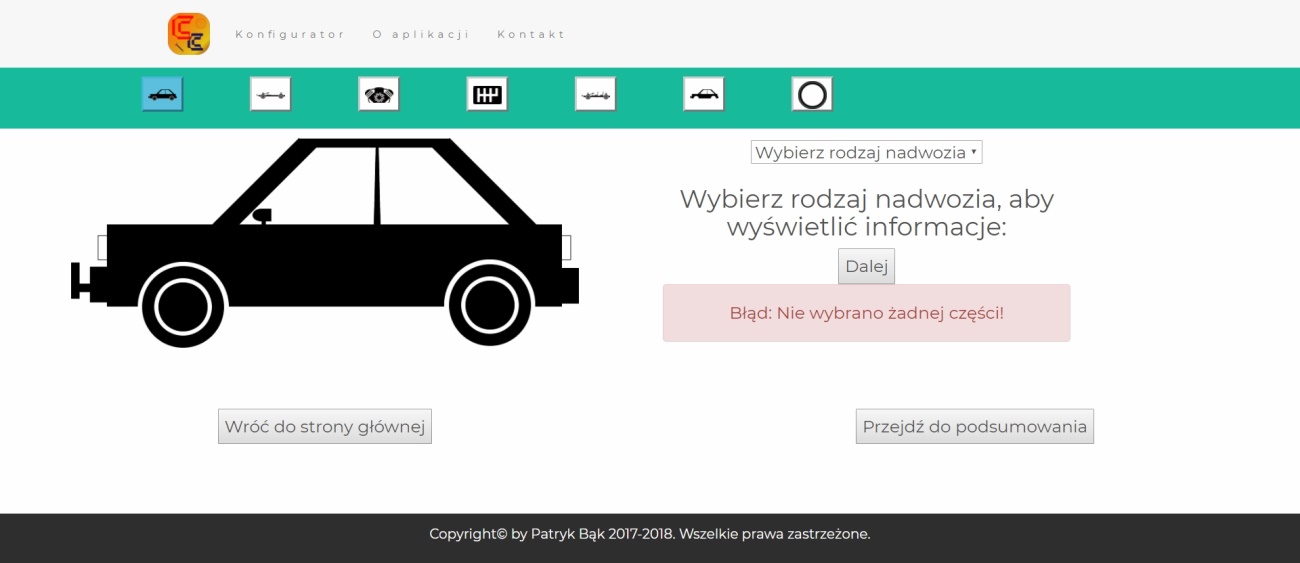
Rys. 16: Część wybrana przez użytkownika jest zapisana w bazie danych i wyświetlona na stronie [źródło: opracowanie własne].

W przypadku, gdy użytkownik zmieni zdanie i wybierze inną część z tej samej strony, widoczne pole zostaje aktualizowane na takie, które jest właściwe dla wybranego elementu []:



Rys. 17: Program zaktualizował pole tabelki w bazie danych - teraz wyświetlana jest ostatnio wybrana część [źródło: opracowanie własne].

Jeśli użytkownik nie wybierze żadnej części i przypadkowo kliknie przycisk „Dalej”, zostaje wyświetlony komunikat o błędzie []:



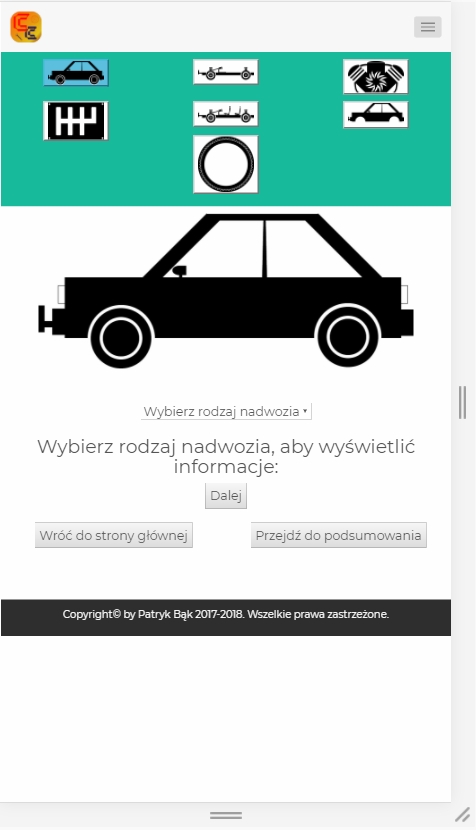
Rys. 18: Komunikat o błędzie związanym z brakiem wybranej części [źródło: opracowanie własne].

Aplikacja również wymaga wybrania najpierw rodzaju nadwozia. W przeciwnym wypadku na każdej kolejnej stronie pojawi się komunikat []:



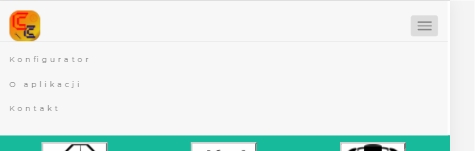
Rys. 19: Błąd związany z brakiem wyboru rodzaju nadwozia [źródło: opracowanie własne].

Warto zauważyć, że zastosowanie frameworka Bootstrap pozwala na elastyczne rozmieszczenie elementów na stronie, dzięki czemu możliwe jest również uruchomienie aplikacji na mniejszych ekranach. Przykładowo, standardowa rozdzielczość ekranu w dzisiejszych smartfonach to 720x1280 pikseli []:



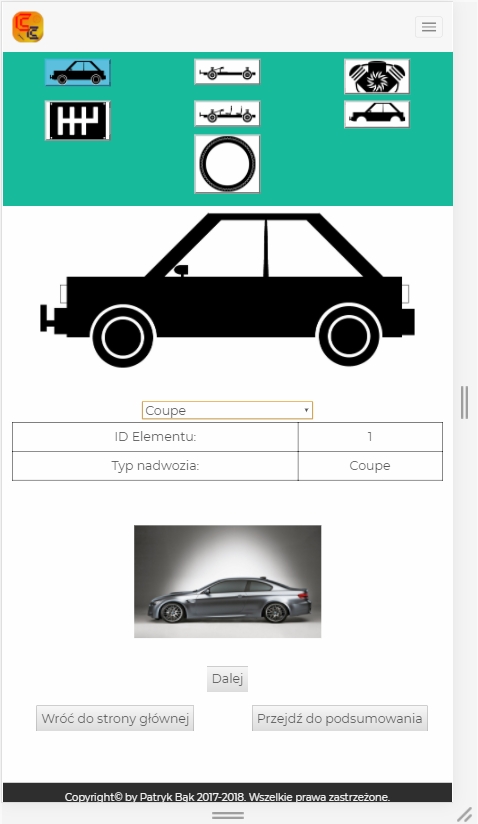
Rys. 20: Układ elementów na stronie w rozdzielczości 720x1280 [źródło: opracowanie własne].

Gdy szerokość ekranu jest mniejsza niż wysokość, pasek nawigacji zwija się do ikony, po kliknięciu której zostaje rozwinięte menu []:

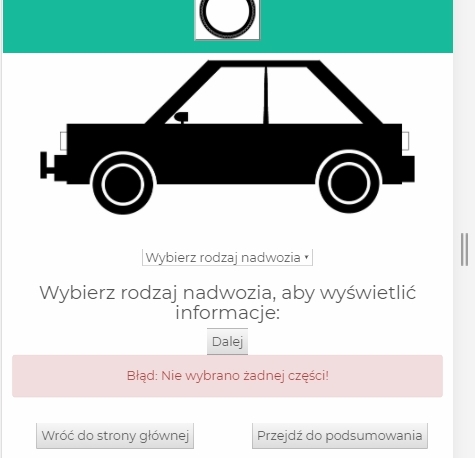


Rys. 21: Rozwinięty pasek menu [źródło: opracowanie własne].

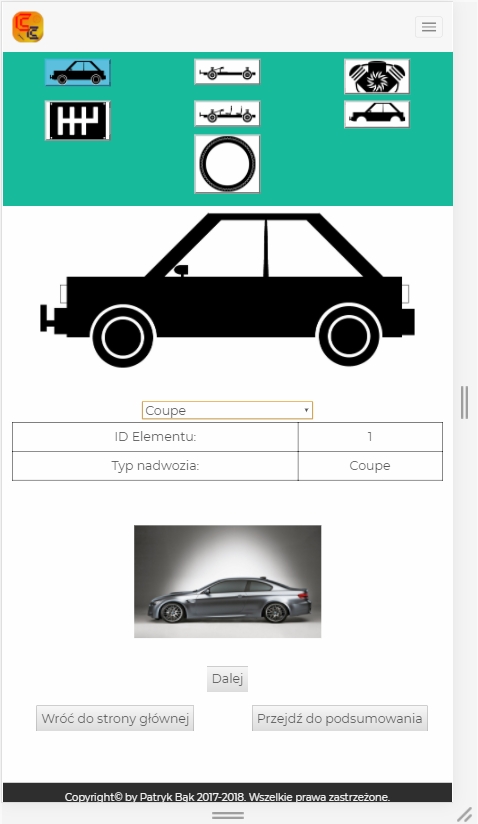
Układ elementów wydaje się przyjazny i czytelny dla użytkownika [, , ,]:



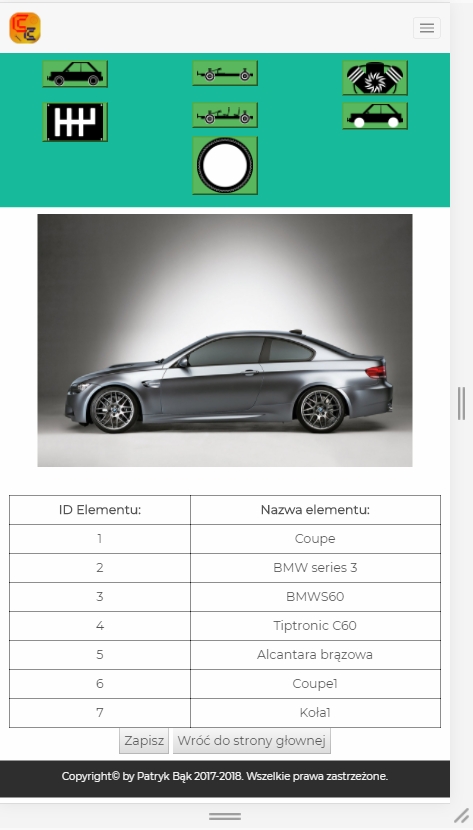
Rys. 22: Pierwsza strona konfiguratora w rozdzielczości 720x1280 [źródło: opracowanie własne].



Rys. 23: Komunikat o błędzie związanym z brakiem wybranej części [źródło: opracowanie własne].



Rys. 24: Układ elementów po wybraniu części z listy [źródło: opracowanie własne].



Rys. 25: Strona podsumowania w rozdzielczości 720x1280 [źródło: opracowanie własne].

# WNIOSKI

Program jest prostym narzędziem do tworzenia własnego samochodu, posiada niewiele opcji wyboru, może być jednak rozwinięty o nowe pozycje. Działanie programu nie budzi zastrzeżeń, choć czasami wydaje się, że mógłby być szybszy.

Aplikacja działa sprawnie, jest odporna na błędy użytkownika i informuje go o wszystkich niepożądanych skutkach jego działań. Kiedy serwer jest włączony, połączenie z bazą danych jest zawsze możliwe. Wszystkie wykonywane czynności są realizowane prawidłowo i bez zastrzeżeń.

Zaproponowane rozwiązanie problemu opiera się na podstawowej znajomości użytkownika w zakresie budowy samochodu i nie wymaga od niego żadnych umiejętności informatycznych (poza umiejętnością obsługi komputera i przeglądarki internetowej). Strony konfiguratora są proste, mają logiczny układ elementów i niewiele funkcji. To wszystko sprawia, że jest to program doskonały dla osób, które posiadają niewielką wiedzę w zakresie wymienionych wyżej zagadnień, ale są zainteresowane tematem motoryzacji i chcą stworzyć własny samochód - czy to w celach rozrywkowych, czy edukacyjnych.

Bardziej atrakcyjnym wizualnie przykładem takiej aplikacji jest wymieniony w rozdziale program *Car visualizer*. Używa on bibliotek WebGL oraz *three.js* do wyświetlania grafiki trójwymiarowej w czasie rzeczywistym. Wymaga on jednak znacznych zasobów sprzętu, na którym jest uruchomiony, przede wszystkim procesora i karty graficznej. Wynika stąd, że w każdym przypadku trzeba znaleźć odpowiedni kompromis, aby dotrzeć do konkretnej grupy użytkowników.

# BIBLIOGRAFIA

x

|  |  |
| --- | --- |
| [1] | https://pl.wikipedia.org/wiki/Mercedes-Benz |
| [2] | https://www.mercedes-benz.pl/content/poland/mpc/mpc\_poland\_website/pl/home\_mpc/passengercars/home/new\_cars/model\_overview\_configurator.html |
| [3] | https://pl.wikipedia.org/wiki/JSON |
| [4] | https://pl.wikipedia.org/wiki/AJAX |
| [5] | http://konfigurator.seat-auto.pl/seat-cc/desktop-desktop-015\_DEFAULT-pl-normal.view?msk=1 |
| [6] | http://carvisualizer.plus360degrees.com/threejs/ |
| [7] | https://threejs.org/ |
| [8] | https://pl.wikipedia.org/wiki/ASP.NET |
| [9] | https://pl.wikipedia.org/wiki/XAMPP |
| [10] | https://pl.wikipedia.org/wiki/Microsoft\_SQL\_Server |
| [11] | https://getbootstrap.com/ |
| [12] | https://www.apachefriends.org/pl/index.html |
| [13] | https://www.jetbrains.com/ |
| [14] | http://www.adobe.com/pl/products/illustrator.html?promoid=PGRQQLFS&mv=other |
| [15] | https://www.w3schools.com/ |
| [16] | https://www.blender.org/ |
| [17] | http://www.adobe.com/pl/products/photoshop.html?promoid=PC1PQQ5T&mv=other |
| [18] | https://www.google.pl/imghp?hl=pl |
| [19] | https://www.stimulsoft.com/en |
| [20] | https://www.google.pl/chrome/browser/desktop/index.html |
| [21] | https://pl.wikipedia.org/wiki/Samoch%C3%B3d |
| [22] | https://pl.wikipedia.org/wiki/HTML |
| [23] | https://pl.wikipedia.org/wiki/Kaskadowe\_arkusze\_styl%C3%B3w |
| [24] | https://pl.wikipedia.org/wiki/Bootstrap\_(framework) |
| [25] | https://pl.wikipedia.org/wiki/MySQL |
| [26] | https://pl.wikipedia.org/wiki/PhpMyAdmin |
| [27] | https://pl.wikipedia.org/wiki/PHP |
| [28] | https://pl.wikipedia.org/wiki/Sesja\_(informatyka) |
| [29] | https://pl.wikipedia.org/wiki/JavaScript |
| [30] | https://www.w3schools.com/js/js\_ajax\_http.asp |
| [33] | Czesław Blok, Wiesław Jeżewski, *Ilustrowany słownik samochodowy 6-języczny*. Wydawnictwa Komunikacji i Łączności, Warszawa, 2016. |
| [34] | Bogdan Brinzarea-Iamandi, Cristian Darie, and Audra Hendrix, *AJAX i PHP. Tworzenie interaktywnych aplikacji internetowych*, wydanie drugie, Helion, Gliwice: 2011. |
| [35] | Laura Lemay, Rafe Colburn, Jennifer Kyrnin, *HTML,CSS i JavaScript dla każdego*, wydanie siódme, Helion, Gliwice, 2016. |
| [36] | Marcin Lis, *PHP i MySQL. Dla każdego*, wydanie trzecie, Helion, Gliwice, 2017. |
| [37] | Marcin Lis, *PHP7. Praktyczny kurs*, Helion, Gliwice, 2017. |
| [38] | Robin Nixon, *PHP, MySQL i JavaScript. Wprowadzenie*, wydanie czwarte, Helion, Gliwice, 2015. |
| [39] | Joachim Potrykus, Krystyna Wilczewska, Eugeniusz Wnuczak, *Poradnik techniki samochodowej*, Rea, Warszawa, 2010. |
| [40] | Jake Spurlock, *Bootstrap*. O'Reilly Media, California, 2013. |
| [41] | Tessloff Verlag, *Co i jak? Samochody*, ATLAS Polska, Wrocław, 2003, str. 4-5. |
| [42] | Tessloff Verlag, *Co i jak? Samochody*, ATLAS Polska, Wrocław, 2003, str. 8. |
| [43] | Tessloff Verlag, *Co i jak? Samochody*, ATLAS Polska,Wrocław, 2003, str. 10. |
| [44] | Praca zbiorowa, *Napędy hybrydowe, ogniwa paliwowe i paliwa alternatywne*. Warszawa: Wydawnictwa Komunikacji i Łączności WKŁ, 2010. |

x

# SPIS LISTINGÓW

[Listing 1: Wykorzystanie technologii AJAX. 32](#_Toc506389960)

[Listing 2: Start sesji i zmienna *s* 40](#_Toc506389961)

[Listing 3: Otwarcie linku w nowej karcie 40](#_Toc506389962)

[Listing 4: Podział na kolumny według siatki Bootstrapa 41](#_Toc506389963)

[Listing 5: Margines jako odrębna kolumna 41](#_Toc506389964)

[Listing 6: Deklaracja wykorzystania pliku z listą 41](#_Toc506389965)

[Listing 7: Instrukcja *switch* obsługująca kroki konfiguratora w pliku *lista.php* 42](#_Toc506389966)

[Listing 8: Warunek określający istnienie zmiennej *n* 43](#_Toc506389967)

[Listing 9: Instrukcja uniemożliwiająca wybranie części bez określenia nadwozia 43](#_Toc506389968)

[Listing 10: Formularz z listą wyboru części 43](#_Toc506389969)

[Listing 11: Deklaracja zmiennej przechowującej nazwy karoserii 44](#_Toc506389970)

[Listing 12: Pętla *for* odpowiadająca za tworzenie opcji wyboru. 45](#_Toc506389971)

[Listing 13: Podpowiedź wyświetlana pod listą 46](#_Toc506389972)

[Listing 14: Przycisk „Dalej” 46](#_Toc506389973)

[Listing 15: Funkcja *showPart* korzystająca z technologii AJAX 47](#_Toc506389974)

[Listing 16: Deklaracja zmiennych używanych do połączenia z serwerem 48](#_Toc506389975)

[Listing 17: Deklaracja wykorzystania pliku *connect.php* 48](#_Toc506389976)

[Listing 18: Instrukcja *try...catch* użyta do łapania wyjątków 49](#_Toc506389977)

[Listing 19: Deklaracja zmiennej *q* 49](#_Toc506389978)

[Listing 20: Połączenie z serwerem przy użyciu technologii AJAX 49](#_Toc506389979)

[Listing 21: Instrukcja warunkowa *switch* w pliku *sync.php* 50](#_Toc506389980)

[Listing 22: Deklaracja zmiennej *nazwatab* 50](#_Toc506389981)

[Listing 23: Warunek określający ograniczenie dla wyboru karoserii 50](#_Toc506389982)

[Listing 24: Zmienna przechowująca rezultat zapytania wysłanego do bazy 51](#_Toc506389983)

[Listing 25: Instrukcja warunkowa dotycząca wyświetlania podpowiedzi pod listą 51](#_Toc506389984)

[Listing 26: Zapis tabelki, która wyświetla się po wybraniu opcji z listy 52](#_Toc506389985)

[Listing 27: Zamknięcie połączenia z bazą danych 53](#_Toc506389986)

[Listing 28: Instrukcja warunkowa *switch* w pliku *dataupdate.php* 54](#_Toc506389987)

[Listing 29: Instrukcja warunkowa kierująca zapytania do bazy danych 54](#_Toc506389988)

[Listing 30: Deklaracja zmiennej przechowującej wybrany rodzaj nadwozia 55](#_Toc506389989)

[Listing 31: Instrukcja *switch* odpowiadająca za przekierowanie użytkownika 55](#_Toc506389990)

[Listing 32: Instrukcja określająca wyświetlanie właściwego obrazka samochodu 56](#_Toc506389991)

[Listing 33: Zmienna przechowująca zawartość tabelki „cookies” 56](#_Toc506389992)

[Listing 34: Tabelka podsumowująca konfigurację samochodu 57](#_Toc506389993)

[Listing 35: Pętla *while* dodająca każdą wybraną część do tabelki 57](#_Toc506389994)

[Listing 36: Przycisk „Zapisz” 57](#_Toc506389995)

[Listing 37: Przycisk „Wróć do strony głównej” 58](#_Toc506389996)

[Listing 38: Funkcja sprawdzająca, czy wybrana została część z listy. 58](#_Toc506389997)

[Listing 39: Funkcja przekierowująca do strony z podsumowaniem 59](#_Toc506389998)

[Listing 40: Plik zapisujący konfigurację 59](#_Toc506389999)

[Listing 41: Funkcja używająca pliku *resetcookies.php*. 60](#_Toc506390000)

[Listing 42: Plik *resetcookies.php* usuwający dane z tabelki „cookies” 60](#_Toc506390001)

# SPIS RYSUNKÓW

[Rys. 1: Konfigurator firmy Mercedes-Benz 10](#_Toc506390402)

[Rys. 2: Konfigurator Seata 12](#_Toc506390403)

[Rys. 3: Konfigurator Car visualizer 13](#_Toc506390404)

[Rys. 4: Błąd spowodowany wyłączonym serwerem. 61](#_Toc506390405)

[Rys. 5: Panel kontrolny pakietu XAMPP 62](#_Toc506390406)

[Rys. 6: Strona główna aplikacji Car-conf 62](#_Toc506390407)

[Rys. 7: Pierwsza strona konfiguratora. 63](#_Toc506390408)

[Rys. 8: Lista wyboru wyświetla wszystkie części z tabelki „Rodzaje\_nadwozia” 63](#_Toc506390409)

[Rys. 9: Wyświetlanie informacji o wybranej części 64](#_Toc506390410)

[Rys. 10: Druga strona konfiguratora 64](#_Toc506390411)

[Rys. 11: Ograniczenie wyboru karoserii wiąże się z wyborem rodzaju nadwozia 65](#_Toc506390412)

[Rys. 12: Przykładowa konfiguracja samochodu pokazana na stronie podsumowania 65](#_Toc506390413)

[Rys. 13: Zapis do pliku .pdf. 66](#_Toc506390414)

[Rys. 14: Komunikat informujący użytkownika o usunięciu danych. 66](#_Toc506390415)

[Rys. 15: Pusta tabelka 67](#_Toc506390416)

[Rys. 16: Wyświetlanie na stronie części wybranej przez użytkownika 67](#_Toc506390417)

[Rys. 17: Aktualizowaacja danych 68](#_Toc506390418)

[Rys. 18: Komunikat o błędzie związanym z brakiem wybranej części 68](#_Toc506390419)

[Rys. 19: Błąd związany z brakiem wyboru rodzaju nadwozia 69](#_Toc506390420)

[Rys. 20: Układ elementów na stronie w rozdzielczości 720x1280 69](#_Toc506390421)

[Rys. 21: Rozwinięty pasek menu 70](#_Toc506390422)

[Rys. 22: Pierwsza strona konfiguratora w rozdzielczości 720x1280 70](#_Toc506390423)

[Rys. 23: Komunikat o błędzie związanym z brakiem wybranej części 71](#_Toc506390424)

[Rys. 24: Układ elementów po wybraniu części z listy 71](#_Toc506390425)

[Rys. 25: Strona podsumowania w rozdzielczości 720x1280. 72](#_Toc506390426)

# SPIS TABEL

[Tabela 1: Pierwsza tabela z bazy danych zawierająca rodzaje nadwozia 34](#_Toc506390453)

[Tabela 2: Druga tabela zawierająca podwozia 35](#_Toc506390454)

[Tabela 3: Trzecia tabela zawierająca silniki 35](#_Toc506390455)

[Tabela 4: Czwarta tabela zawierająca skrzynie biegów. 35](#_Toc506390456)

[Tabela 5: Piąta tabela zawierająca wnętrza 35](#_Toc506390457)

[Tabela 6: Szósta tabela zawierająca karoserie 35](#_Toc506390458)

[Tabela 7: Siódma tabela zawierająca koła 36](#_Toc506390459)

[Tabela 8: Tabela „cookies” 36](#_Toc506390460)

[Tabela 9: Zaznaczenie wartości używanych przez pętlę *for* na przykładzie tabeli 45](#_Toc506390461)