

BAZY DANYCH

Wypożyczalnia Samochodów Sportowych

Etap 3 - Projekt, implementacja i testy aplikacji bazodanowej

Imię i Nazwisko:	Remigiusz Mielcarz, Mateusz Świątek
Nr indeksu:	252887, 252858

Termin zajęć: dzień tygodnia, godzina:	Piątek 15:15-16:55
Numer grupy ćwiczeniowej:	Y00-37m
Data wykonania ćwiczenia:	02.06.2022
Termin do oddania sprawozdania:	03.06.2022
Prowadzący kurs:	Dr inż. Paweł Głuchowski

Spis treści

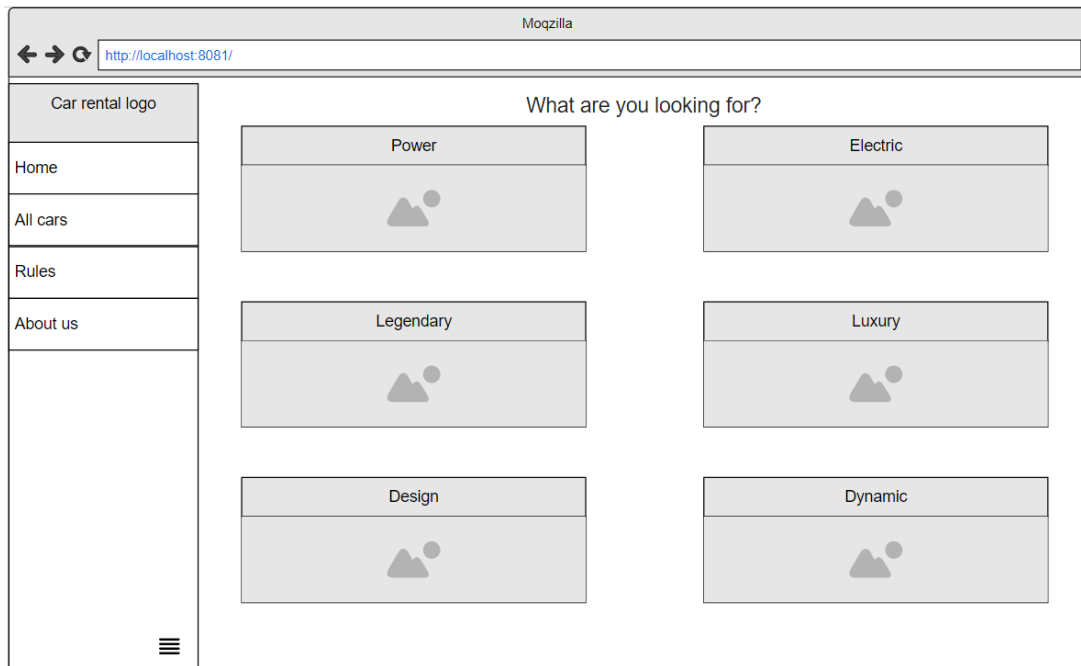
1	Makieta interfejsu graficznego	3
1.1	Strona główna	3
1.2	Przeglądanie samochodów w ofercie wypożyczalni	4
1.3	Zasady wypożyczalni	5
1.4	O wypożyczalni	6
1.5	Podstrona z informacjami o samochodzie	7
2	Wdrożenie aplikacji	8
2.1	Sprawdzenie statusu wypożyczenia	8
2.2	Wypisanie danych konkretnego auta	9
2.3	Sprawdzenie poprawności wpisywanych filtrów	10
2.4	Tworzenie odnośnika URL do bazy danych	11
2.5	Sortowanie bąbelkowe po mocy	11
2.6	Dane zwracane do klienta serwera	12
2.7	Zapytanie Query filtracja	13
2.8	Deklaracja Listy typu Item po czym będzie filtracja	14
2.9	Tabela SQL Item	15

3	Przetestowanie aplikacji bazodanowej	15
3.1	Wypożyczenie dostępnego samochodu	15
3.2	Próba wypożyczenia niedostępnego samochodu	17
3.3	Testy filtracji	18
3.4	Testy intuicyjności	20
4	Podsumowanie	20

1 Makieta interfejsu graficznego

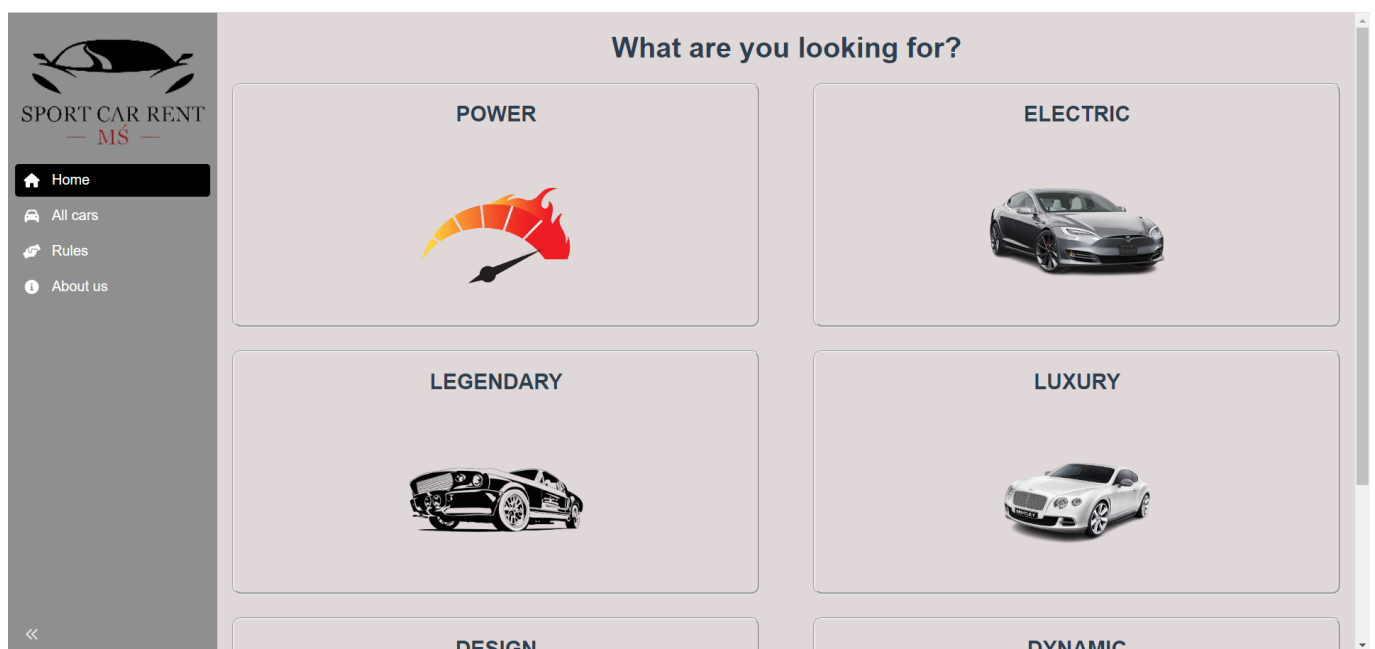
1.1 Strona główna

Stworzyliśmy makietę interfejsu Home Page, na której bazowaliśmy wygląd naszej strony głównej w aplikacji webowej.

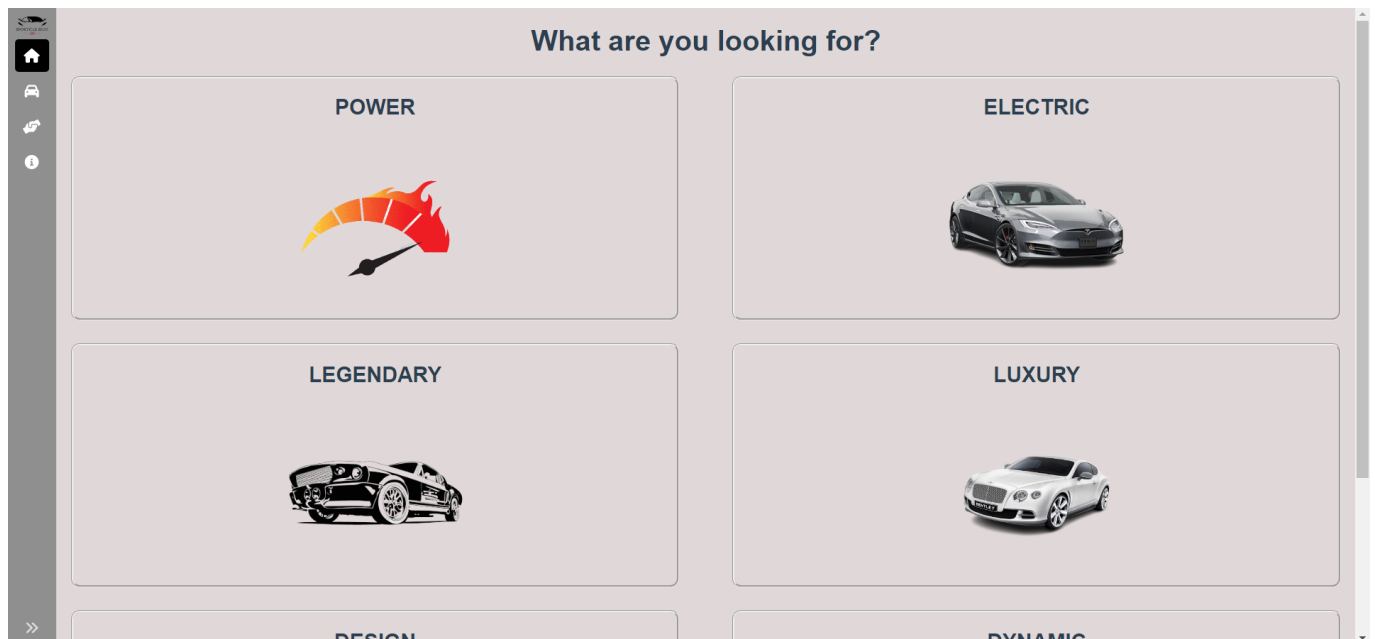


Rysunek 1: Widok strony głównej

Na poniższym zdjęciu widzimy wygląd strony głównej, która zostaje załadowana po uruchomieniu aplikacji.



Rysunek 2: Widok strony głównej z rozwiniętym paskiem bocznym

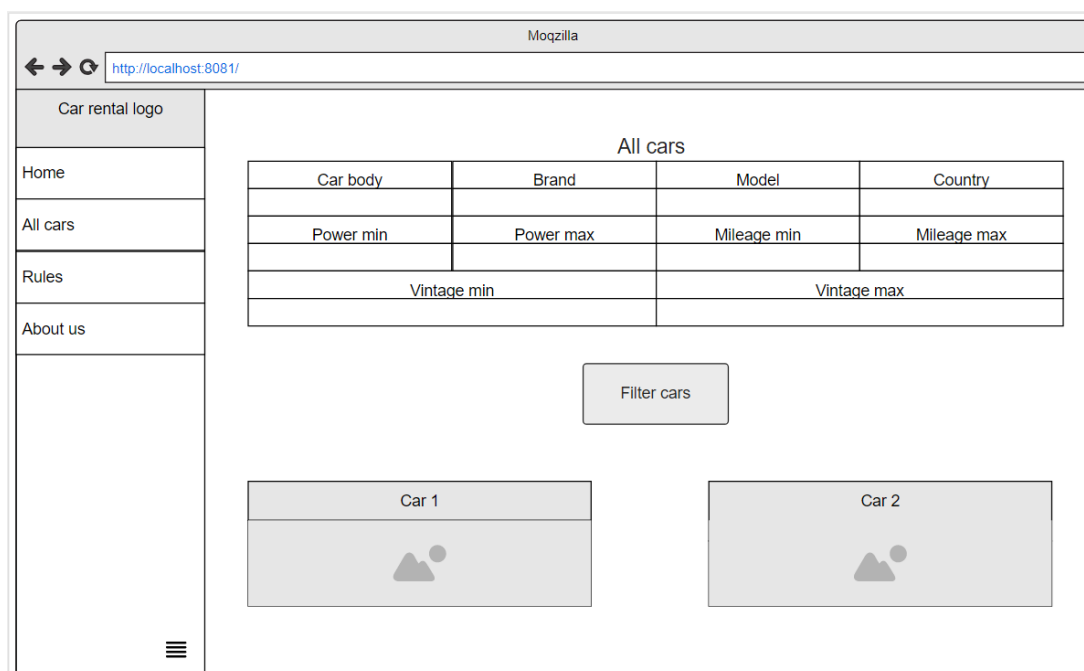


Rysunek 3: Widok strony głównej ze zwiniętym paskiem bocznym

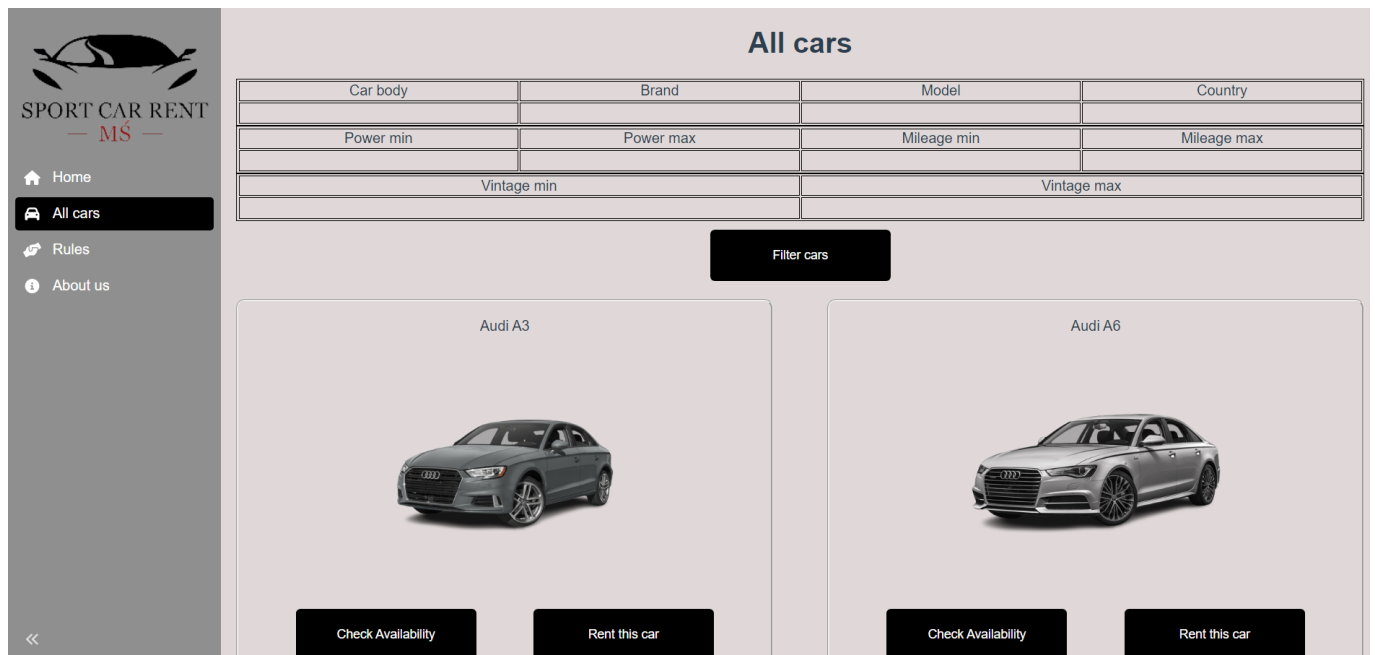
Powyższa strona składa się z rozwijanego paska bocznego z wyborem kart na stronie oraz z głównego widoku wyboru rodzaju samochodu, który chcemy wypożyczyć.

1.2 Przeglądanie samochodów w ofercie wypożyczalni

Kolejną przydatną funkcjonalnością aplikacji jest przeglądanie dostępnych samochodów. W tym celu należało stworzyć stronę „All cars”.



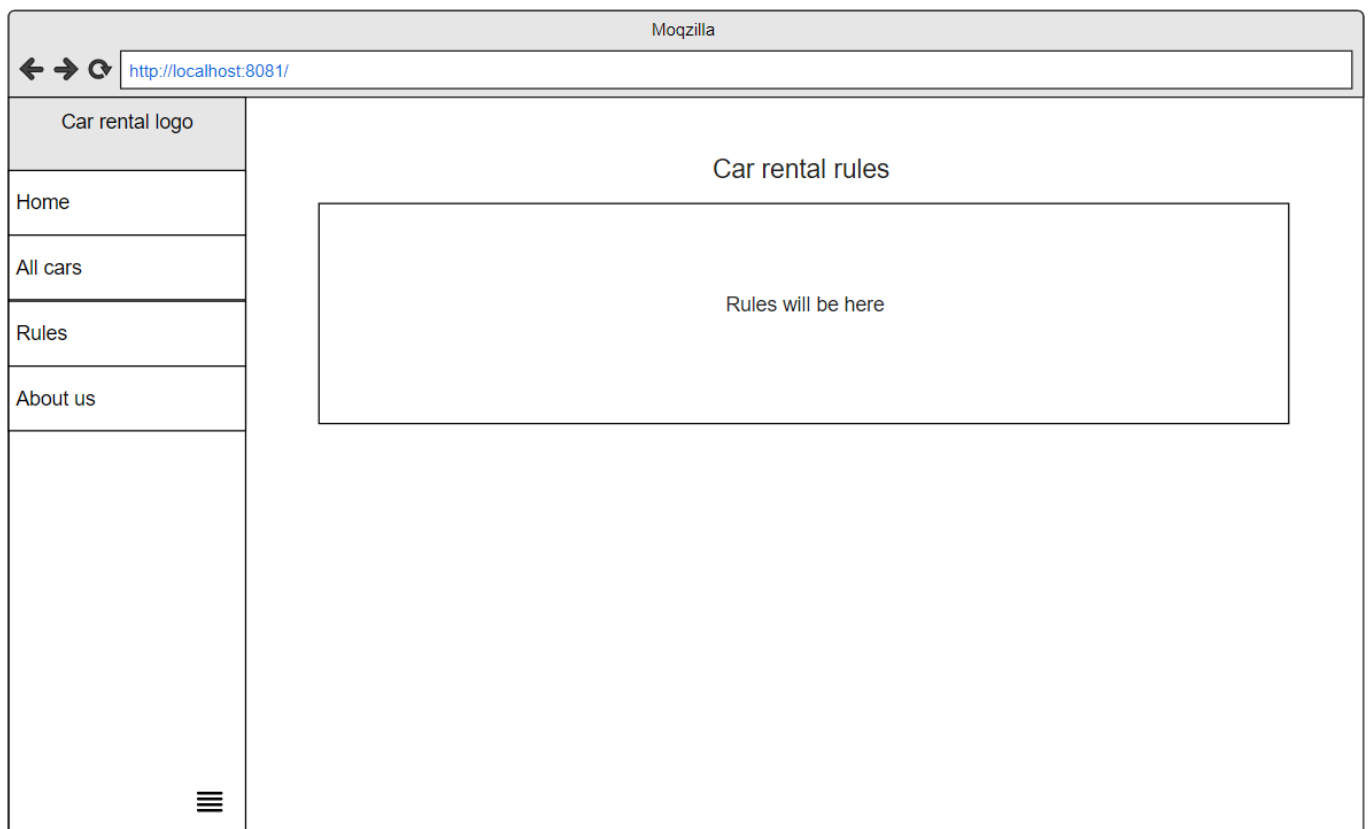
Rysunek 4: Widok strony All cars



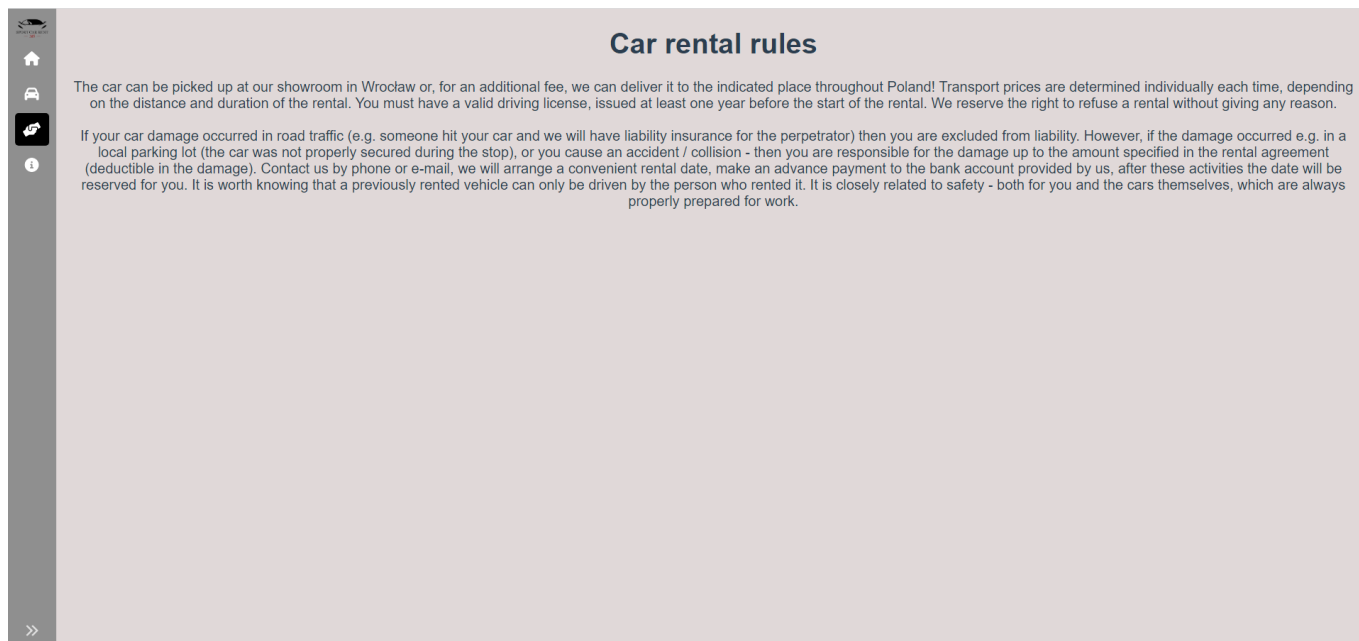
Rysunek 5: Widok strony All cars w aplikacji

1.3 Zasady wypożyczalni

Wypożyczalnia samochodów powinna posiadać także stronę z zasadami oraz informacjami o działaniu firmy.



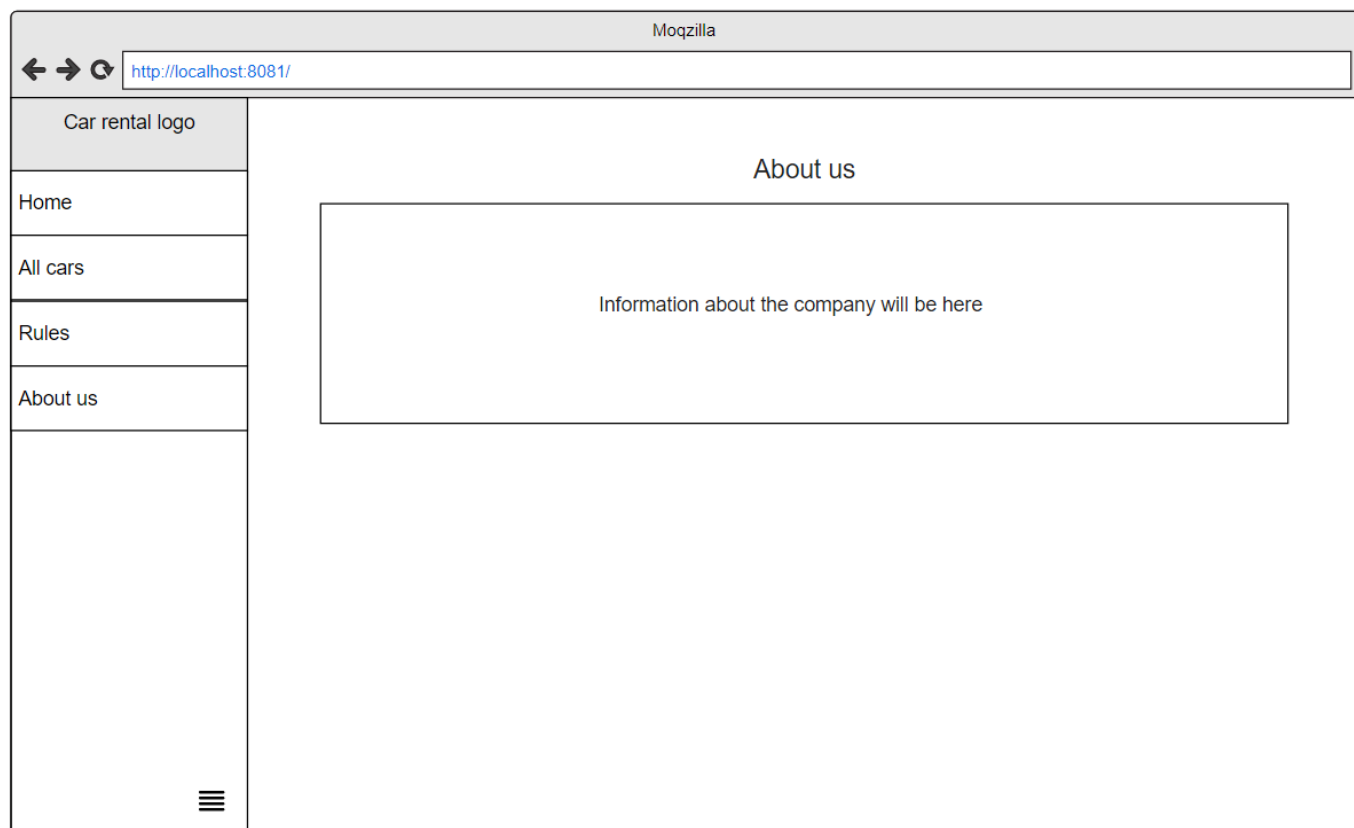
Rysunek 6: Widok strony Car rental rules



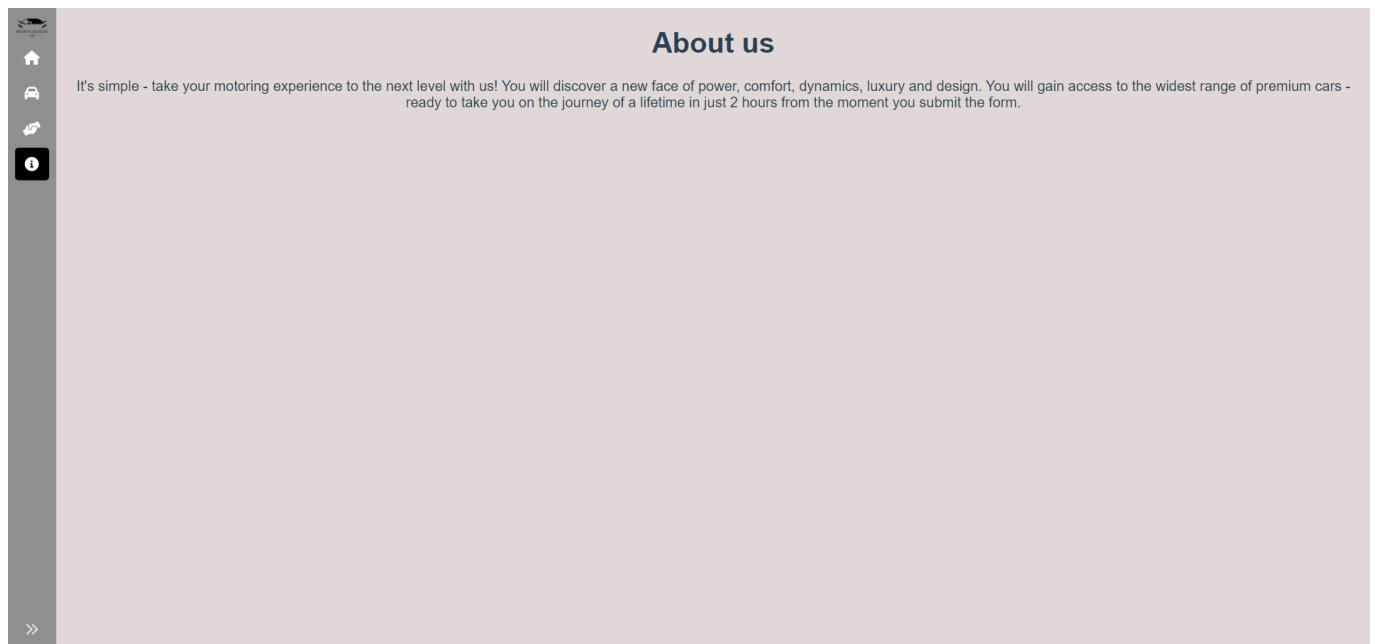
Rysunek 7: Widok strony Car rental rules w aplikacji

1.4 O wypożyczalni

Analogicznie dodaliśmy także stronę z informacjami o wypożyczalni.

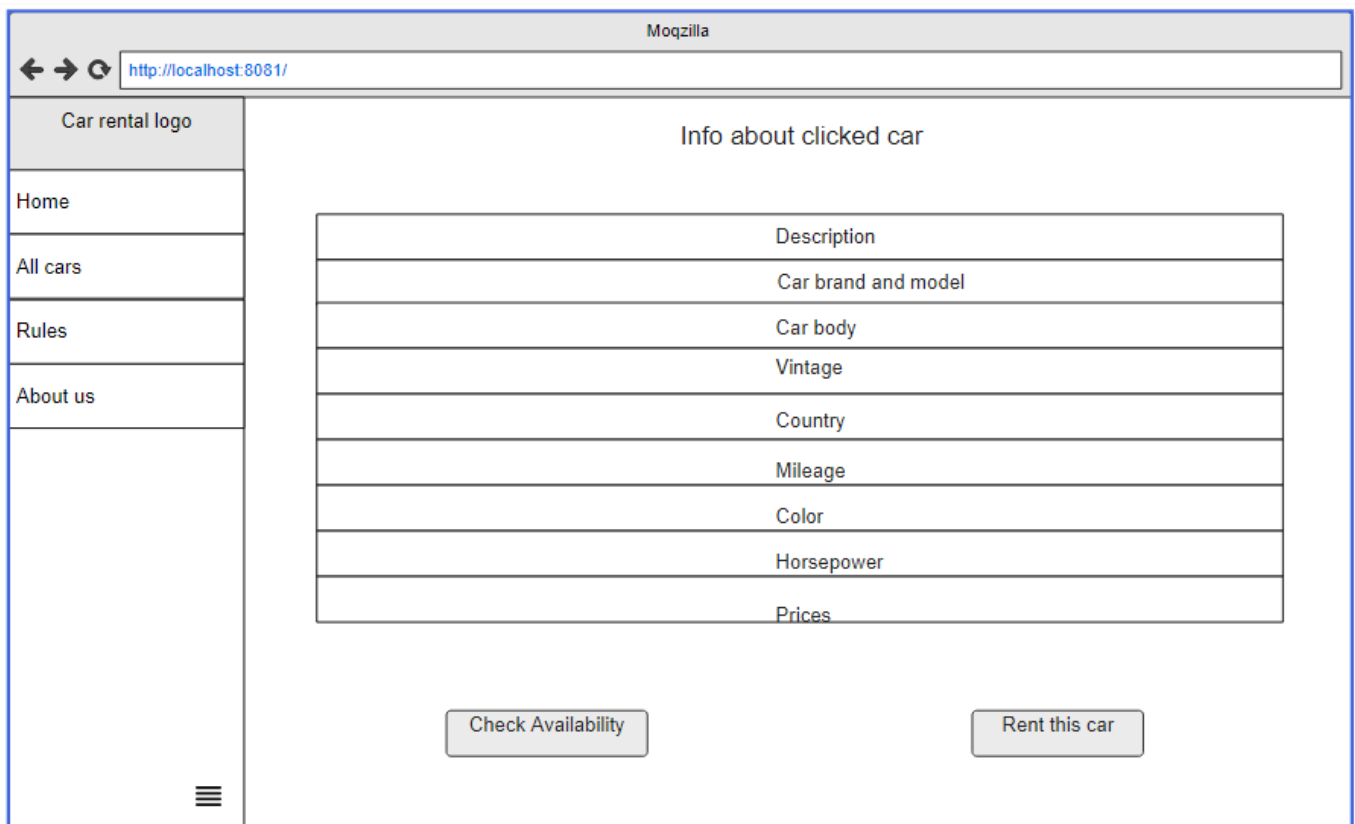


Rysunek 8: Widok strony About us

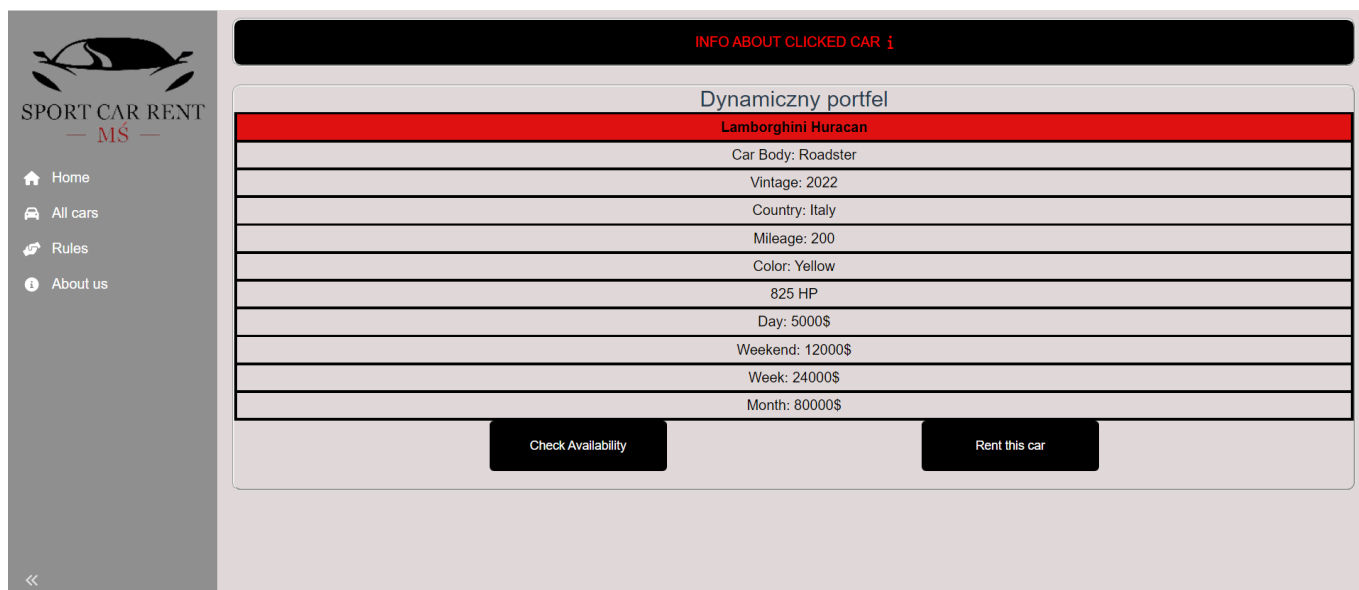


Rysunek 9: Widok strony About us w aplikacji

1.5 Podstrona z informacjami o samochodzie



Rysunek 10: Widok podstrony



Rysunek 11: Widok podstrony w aplikacji

2 Wdrożenie aplikacji

Baza danych została stworzona przy pomocy zapytań SQL w Postgres. Backend aplikacji stworzono w Java Spring Boot. Frontend wykonano przy pomocy Vue.js. Sprawdzanie działania odnośników do bazy danych realizowano przy pomocy Postman.

2.1 Sprawdzenie statusu wypożyczenia

```
methods: {
  checkAvailable(selectedCar) {
    if(selectedCar.status === false) {
      console.log(selectedCar)
      this.selectedCar = selectedCar;
      return this.message = 'Unavailable';
    }
    else {
      document.getElementById( elementId: "firstbtn").style.display = "none";
      document.getElementById( elementId: "secondbtn").style.display = "block";
      this.selectedCar = selectedCar;
      return this.message = 'Available';
    }
  },
  rentCar(selectedCar) {
    if(selectedCar.status === true) {
      axios.patch( url: "http://localhost:8080/api/items/rent/"+selectedCar.modelName, data: {});
      this.selectedCar=selectedCar
      this.$router.push('/')
    }
  }
}
```


2.2 Wypisanie danych konkretnego auta

```
<table class="customTable">
  <thead>
    <tr>
      <th>{{ selectedCar.brandName + " " + selectedCar.modelName }}</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>{{ "Car Body: " + selectedCar.carBody }}</td>
    </tr>
    <tr>
      <td>{{ "Vintage: " + selectedCar.vintage }}</td>
    </tr>
    <tr>
      <td>{{ "Country: " + selectedCar.country }}</td>
    </tr>
    <tr>
      <td>{{ "Mileage: " + selectedCar.mileage + " miles" }}</td>
    </tr>
    <tr>
      <td>{{ "Color: " + selectedCar.color }}</td>
    </tr>
    <tr>
      <td>{{ selectedCar.power + " HP" }}</td>
    </tr>
    <tr>
```

2.3 Sprawdzenie poprawności wpisywanych filtrów

```
data: function () {  
  const schema = yup.object().shape({  
    carBody: yup  
      .string(),  
    brandName: yup  
      .string(),  
    model: yup  
      .string(),  
    country: yup  
      .string(),  
    vintage_min: yup  
      .number()  
      .moreThan(1900)  
      .lessThan(2023),  
    vintage_max: yup  
      .number()  
      .moreThan(1900)  
      .lessThan(2023),  
    mileage_min: yup  
      .number()  
      .positive(),  
    mileage_max: yup  
      .number()  
      .positive(),  
    power_min: yup  
      .number()  
      .positive(),  
    power_max: yup  
      .number()  
      .positive(),  
    color: yup  
      .string(),  
  })  
}
```

2.4 Tworzenie odnośnika URL do bazy danych

```
let URL = 'http://localhost:8080/api/items' +
  '?brand_name=${advert.brandName}&model=${advert.model}&car_body=${advert.carBody}' +
  '&color=${advert.color}&vintage_min=${advert.vintage_min}' +
  '&vintage_max=${advert.vintage_max}&mileage_min=${advert.mileage_min}&mileage_max=${advert.mileage_max}' +
  '&power_min=${advert.power_min}&power_max=${advert.power_max}&country=${advert.country}';
this.$emit('carURL', URL)

console.log(URL)
axios.get(URL).then(function (response) {
  this.sortedCars = response.data
}.bind(this))
},
```

2.5 Sortowanie bąbelkowe po mocy

```
methods: {
  fetchItems() {
    axios.get({ url: "http://localhost:8080/api/items" }).then(function (response) {
      this.allCars = response.data;
      let temp = 0;

      for (let i = 0; i < this.allCars.length; i++) {
        for (let j = 0; j < this.allCars.length; j++) {
          if(this.allCars[j].power < this.allCars[i].power)
          {
            temp = this.allCars[i];
            this.allCars[i] = this.allCars[j];
            this.allCars[j] = temp;
          }
        }
      }
    }.bind(this))
  },
}
```

2.6 Dane zwracane do klienta serwera

```
return new ItemResponse(  
    advert.getTitle(),  
    advert.getPicture(),  
    advert.getType(),  
    carModel.getDescription().getSpecification(),  
    carBrand.getBrandName(),  
    carModel.getModelName(),  
    carModel.getCarBody(),  
    carBrand.getCountry(),  
    parameters.getVintage(),  
    parameters.getPower(),  
    parameters.getMileage(),  
    parameters.getColor(),  
    priceList.getDay(),  
    priceList.getWeekend(),  
    priceList.getWeek(),  
    priceList.getMonth(),  
    status.getIsAvailable()  
);
```

2.7 Zapytanie Query filtracja

```
@Query("SELECT a " +  
    "FROM Item a " +  
    "WHERE a.car.parameters.power BETWEEN ?1 AND ?2 " +  
    "AND a.car.parameters.vintage BETWEEN ?3 AND ?4 " +  
    "AND a.car.parameters.mileage BETWEEN ?5 AND ?6 " +  
    "AND a.car.carModel.priceList.day BETWEEN ?7 AND ?8 " +  
    "AND a.car.carModel.priceList.weekend BETWEEN ?9 AND ?10 " +  
    "AND a.car.carModel.priceList.week BETWEEN ?11 AND ?12 " +  
    "AND a.car.carModel.priceList.month BETWEEN ?13 AND ?14 " +  
    "AND a.car.carModel.modelName LIKE ?15 " +  
    "AND a.car.carModel.carBrand.country LIKE ?16 " +  
    "AND a.car.carModel.carBody LIKE ?17 " +  
    "AND a.car.carModel.carBrand.brandName LIKE ?18 ")
```

2.8 Deklaracja Listy typu Item po czym będzie filtracja

```
List<Item> findByFilters(  
    Integer powerMin,  
    Integer powerMax,  
    Integer vintageMin,  
    Integer vintageMax,  
    Integer mileageMin,  
    Integer mileageMax,  
    Integer dayMin,  
    Integer dayMax,  
    Integer weekendMin,  
    Integer weekendMax,  
    Integer weekMin,  
    Integer weekMax,  
    Integer monthMin,  
    Integer monthMax,  
    String modelName,  
    String country,  
    String carBody,  
    String brandName  
);
```

2.9 Tabela SQL Item

	id	title	picture	type	car_id	status_id
1	1	Luksus i wygoda	a3.png	luxury	1	2
2	2	Szybka limuzyna	a6.png	luxury	2	2
3	4	Dynamiczny portfel	huracan.png	dynamic	4	1
4	6	Mistrz Nurburgring	m5cs.png	dynamic	6	1
5	8	Masywny supersamochód	charger.png	legendary	8	1
6	9	Biała perła	r35.png	dynamic	9	2
7	10	Tokyo Style	z.png	dynamic	10	2
8	12	Silny staruszek	mustang.png	legendary	12	2
9	3	Czerwony diabeł	aventador.png	dynamic	3	1
10	7	0d 0 do 100kmh w 3.4s	challenger.png	legendary	7	1
11	5	Piękny bulgot V8	e92m3.png	design	5	1
12	11	Szybki elektryk	tesla.png	electric	11	1


3 Przetestowanie aplikacji bazodanowej

3.1 Wypożyczenie dostępnego samochodu

W celu przetestowania aplikacji należy wypożyczyć dostępny samochód. Spróbujemy wypożyczyć Tesłę:

	id	title	picture	type	car_id	status_id
1	1	Luksus i wygoda	a3.png	luxury	1	2
2	4	Dynamiczny portfel	huracan.png	dynamic	4	1
3	6	Mistrz Nurburgring	m5cs.png	dynamic	6	1
4	7	0d 0 do 100kmh w 3.4s	challenger.png	legendary	7	2
5	8	Masywny supersamochód	charger.png	legendary	8	1
6	11	Szybki elektryk	tesla.png	electric	11	2

Tesla Model S



350 HORSEPOWER!


Check Availability

Rent this car

Available

Po wciśnięciu 'Rent this car' zostajemy przeniesieni do strony głównej. Po ponownym wejściu w listę samochodów widzimy, że Tesla jest już niedostępna.

Tesla Model S



350 HORSEPOWER!

Check Availability

Rent this car

Unavailable

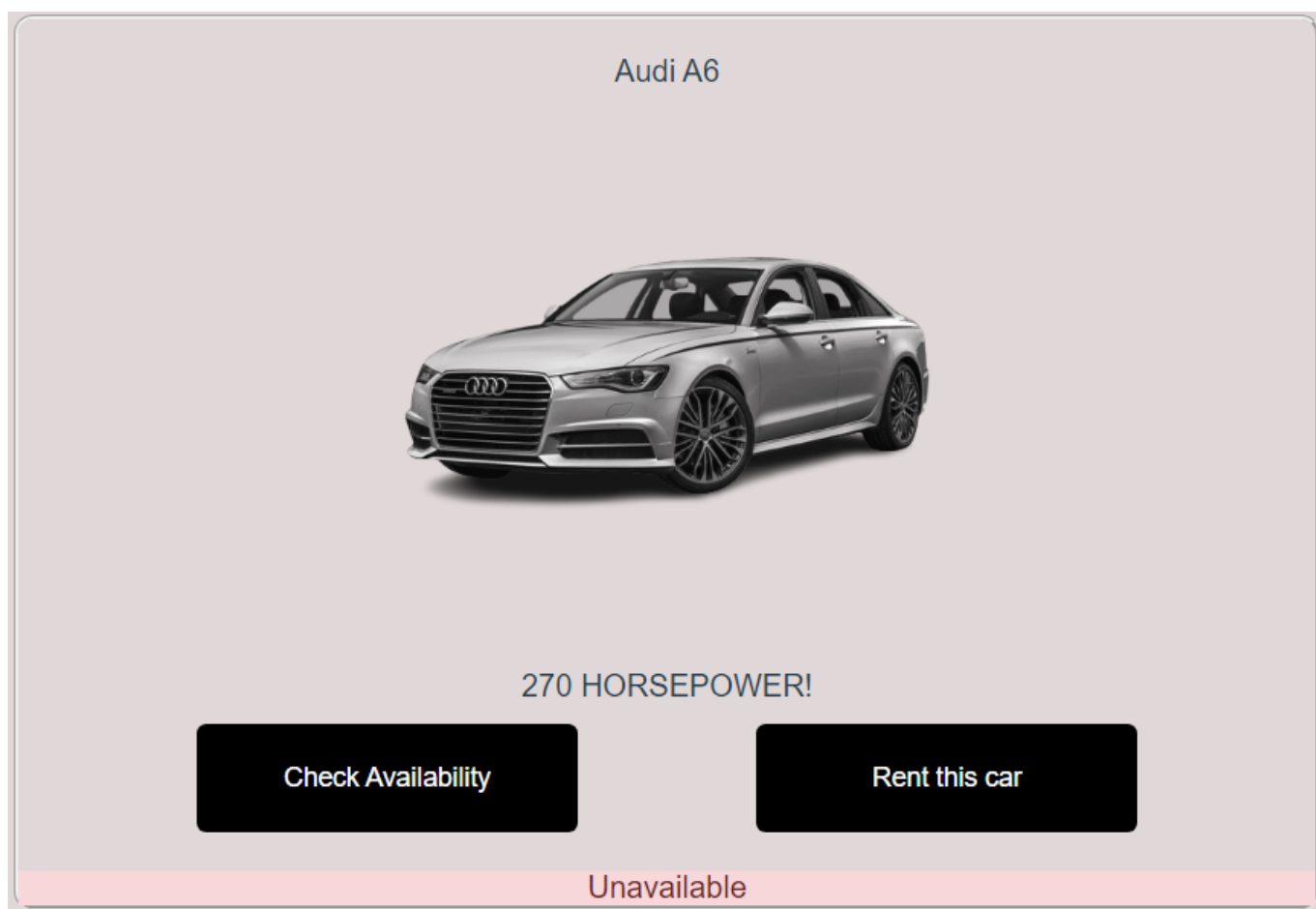
	id	title	picture	type	car_id	status_id
1	12	Silny staruszek	mustang.png	legendary	12	1
2	11	Szybki elektryk	tesla.png	electric	11	1

W bazie danych został zmieniony jej status na niedostępny, a zatem połączenia aplikacji webowej z bazą danych działa poprawnie.

3.2 Próba wypożyczenia niedostępnego samochodu

Należy sprawdzić także, co dzieje się w przypadku, gdy nasz użytkownik spróbuje wynająć niedostępny samochód. W tym celu wynajmijmy niedostępne Audi A6 :

	id	title	picture	type	car_id	status_id
1	1	Luksus i wygoda	a3.png	luxury	1	2
2	2	Szybka limuzyna	a6.png	luxury	2	1



Po wciśnięciu przycisku 'Rent this car' aplikacja nie pozwoliła nam na wypożyczenie pojazdu.

	id	title	picture	type	car_id	status_id
1	1	Luksus i wygoda	a3.png	luxury	1	2
2	2	Szybka limuzyna	a6.png	luxury	2	1

W bazie danych status samochodu nie uległ zmianie. Co za tym idzie, aplikacja webowa działa poprawnie.

3.3 Testy filtracji

Aby przetestować działanie filtracji sprawdziliśmy działanie dla poniższych scenariuszy :

Podanie nieprawidłowej wartości do filtru

All cars

Car body		Brand		Model		Country	
Power min		Power max		Mileage min		Mileage max	
		<input type="text" value=""/> <small>power_max must be a 'number' type, but the final value was: 'NaN' (cast from the value "").</small>					
Vintage min				Vintage max			

Filter cars

Do okna Power max nie da się wpisać liter, jedynie cyfry. Z tego powodu po wciśnięciu 'Filter cars' dostajemy ostrzeżenie, że nasze pole z mocą jest puste.

Podanie nielogicznej wartości do filtru

All cars

Car body		Brand		Model		Country	
Power min		Power max		Mileage min		Mileage max	
		1				1	
Vintage min				Vintage max			

Filter cars


Po podaniu zbyt małej wartości w 'Mileage max' nie został wyświetlony żaden samochód, ponieważ pojazdy z bazy danych nie spełniają kryterium wyszukiwania.

Przetestowanie działania jednego filtra

Car body	Brand	Model	Country
			Italy
Power min	Power max	Mileage min	Mileage max
Vintage min		Vintage max	

Filter cars


Lamborghini Aventador



Check Availability

Rent this car

Lamborghini Huracan



Check Availability

Rent this car

Po podaniu kraju zostały wyświetlone jedynie włoskie samochody.


Przetestowanie działania kilku filtrów

All cars

Car body	Brand	Model	Country
		Huracan	Italy
Power min	Power max	Mileage min	Mileage max
100	900		1000
Vintage min		Vintage max	

Filter cars

Lamborghini Huracan



Check Availability

Rent this car

Po podaniu kilku wartości do filtracji został wyświetlony interesujący nas samochód.

3.4 Testy intuicyjności

W celu przetestowania intuicyjności, oddaliśmy naszą aplikację do testów jednemu z kolegów. Otrzymaliśmy następujący feedback :

- W sekcji filtracji po podaniu nieprawidłowej wartości do filtru użytkownik otrzymuje informację o tym jaka jest prawidłowa wartość. Jest to bardzo pomocne.
- Do obsługi aplikacji potrzebna jest znajomość języka angielskiego na poziomie minimum A1/A2
- UI jest przejrzyste, łatwe przemieszczanie się pomiędzy kartami.
- Wypożyczenie samochodu oraz sprawdzenie jego statusu można wykonać na tyle łatwo, że większość osób obsługujących komputer powinna sobie z tym poradzić.

4 Podsumowanie

Zaimplementowaliśmy większość funkcjonalności, które zostały podane przez nas w pierwszym etapie projektu.