

**Rozproszone systemy internetowe**

**PROJEKT 1 – SOAP WS**

**System rezerwacji i wynajmowania motocykli.**

Autorzy:

- Michał Wołosewicz
- Patryk Wójtowicz

Studia dzienne

Kierunek: Informatyka II stopień  
Semestr: I

Grupa zajęciowa: PS 4

Prowadzący ćwiczenie: dr hab. inż. prof. PB Jacek Grekow

Data wykonania oddania Projektu: 13.05.2024

# 1. Wprowadzenie

Celem projektu było stworzenie systemu pozwalającego na rezerwację i wynajem motocykli, wykorzystującego web serwis SOAP. Oprócz wyżej wymienionych funkcjonalności, system pozwala na zdalne zarządzanie bazą danych wypożyczalni, rezygnację z rezerwacji, filtrowanie motocykli na podstawie marki, wyświetlanie szczegółowych danych pojazdów oraz generowanie dokumentacji w formie PDF. Usługi wykorzystują middleware .NET (handlers) oraz protokół HTTPS.

## 2. Realizacja

### 2.1. Serwer

Do realizacji serwera wykorzystano platformę .NET, język programowania C# i silnik bazodanowy MSSQL. Kluczowym elementem serwera jest klasa MotorService, która implementuje interfejs usługi SOAP IMotorService (Rys. 1). Ta klasa zawiera metody umożliwiające zarządzanie motocyklami w bazie danych, takie jak dodawanie, usuwanie, aktualizowanie, rezerwowanie, anulowanie rezerwacji, wynajmowanie maszyn oraz generowanie dokumentów PDF. W projekcie zaimplementowano obsługę wyjątków po stronie serwera z wykorzystaniem Middleware (handlers) (Rys 2). Na rysunku 3 przedstawiono model DBMotor, który służy do reprezentacji danych. Wszystkie obiekty bazy danych posiadają pola:

- Id- Unikalny identyfikator,
- Brand- Marka motocykla,
- Name- Model motocykla,
- RequiredLicence- Rodzaj wymaganego prawa jazdy,
- Descriptions- Opis pojazdu,
- RentPrice- Cena wynajmu za dobę,
- RentTo- Data, do której motocykl jest wynajęty,
- Reservation- Flaga określająca, czy motocykl jest zarezerwowany.

```
Odwolanie 3
public interface IMotorService
{
    [OperationContract]
    1 odwołanie
    public Task Create(CreateMotor motor);

    [OperationContract]
    1 odwołanie
    public Task Remove(int id);

    [OperationContract]
    1 odwołanie
    public Task Update(UpdateMotor motor);

    [OperationContract]
    1 odwołanie
    public Task<DetailMotor> Detail(int id);

    [OperationContract]
    1 odwołanie
    public Task<List<DetailMotor>> GetAll();

    [OperationContract]
    1 odwołanie
    public Task<List<DetailMotor>> GetSelected(string brandString);

    [OperationContract]
    1 odwołanie
    public Task Reserve(int id);

    [OperationContract]
    1 odwołanie
    public Task CancelReserve(int id);

    [OperationContract]
    1 odwołanie
    public Task Rent(int id, int numberOfDays);

    [OperationContract]
    1 odwołanie
    public Task<byte[]?> GeneratePDF(int id);
}
```

Rysunek 1 Interfejs IMotorService.

```
Odwolanie 2
public class ExceptionMiddleware
{
    private readonly RequestDelegate _requestDelegate;

    Odwołanie 0
    public ExceptionMiddleware(RequestDelegate _requestDelegate) => this._requestDelegate = _requestDelegate;

    Odwołanie 0
    public async Task Invoke(HttpContext context)
    {
        try
        {
            await _requestDelegate(context);
        }
        catch (Exception ex)
        {
            await HandleExceptionAsync(context, ex);
        }
    }

    1 odwołanie
    private Task HandleExceptionAsync(HttpContext context, Exception ex)
    {
        context.Response.ContentType = "text/plain";
        HttpStatusCode statusCode = HttpStatusCode.InternalServerError;

        switch (ex)
        {
            case NotFoundException or MotorbikeReservedException or MotorbikeNotReservedException
            or MotorbikeCannotBeRentedException or ThisMotorbikeIsNotRentedException:
                statusCode = HttpStatusCode.BadRequest;
                break;
            case ConflictCreateInvoiceException:
                statusCode = HttpStatusCode.Conflict;
                break;
        }

        context.Response.StatusCode = (int)statusCode;
        return context.Response.WriteAsync(ex.Message);
    }
}

Odwolanie 0
public static class SoapCoreExceptionMiddleware
{
    1 odwołanie
    public static IApplicationBuilder UseSoapExceptionMiddleware(this IApplicationBuilder builder)
    => builder.UseMiddleware<ExceptionMiddleware>();
}
```

Rysunek 2 Obsługa Middleware. MotorbikeReservedException.cs

```

Odwolania: 17
public enum Licence
{
    B_A1, A2, A
}
Odwolania: 19
public class DBMotor : UpdateMotor
{
    Odwolania: 26
    public int Id { get; set; }
    Odwolania: 22
    public string Brand { get; set; }
    Odwolania: 21
    public string Name { get; set; }
    Odwolania: 20
    public Licence RequiredLicence { get; set; }
    Odwolania: 21
    public string Description { get; set; }
    Odwolania: 21
    public int RentPrice { get; set; }
    Odwolania: 24
    public DateTime? RentTo { get; set; } = null;
    Odwolania: 8
    public bool Reservation { get; set; } = false;
}

```

Rysunek 3 Model DBMotor.cs

## 2.1.1 Implementacja Metod

W tym rozdziale zostanie przedstawiona implementacja metod zawartych w klasie MotorService.

```

1 odwołanie
public async Task Create(CreateMotor motor)
{
    await _context.Motors.AddAsync(new DBMotor
    {
        Brand = motor.Brand,
        Name = motor.Name,
        Description = motor.Description,
        RequiredLicence = motor.RequiredLicence,
        RentPrice = motor.RentPrice
    });
    await _context.SaveChangesAsync();
}

```

Rysunek 4 Metoda Create()

**Metoda Create()** służy do dodawania nowego motocykla do bazy danych. Przyjmuje obiekt CreateMotor zawierający dane nowego motocykla. Po dodaniu motocykla do kontekstu bazy danych za pomocą metody AddAsync(), dane są zapisywane za pomocą metody SaveChangesAsync().

```

1 odwołanie
public async Task Remove(int id)
{
    var motor = await _context.Motors.FirstOrDefaultAsync(motor => motor.Id == id);
    if (motor is null) throw new NotFoundException(id);
    _context.Motors.Remove(motor);
    await _context.SaveChangesAsync();
}

```

Rysunek 5 Metoda Remove()

**Metoda Remove()** usuwa motocykl z bazy danych na podstawie jego identyfikatora. Sprawdza, czy motocykl istnieje w bazie danych, a następnie usuwa go za pomocą metody Remove() i zapisuje zmiany.

```

1 odwołanie
public async Task Update(UpdateMotor motor)
{
    var motor2 = await _context.Motors.FirstOrDefaultAsync(m => m.Id == motor.Id);
    if (motor2 is null) throw new NotFoundException(motor.Id);
    motor2.Brand = motor.Brand;
    motor2.Name = motor.Name;
    motor2.Description = motor.Description;
    motor2.RequiredLicence = motor.RequiredLicence;
    motor2.RentPrice = motor.RentPrice;
    await _context.SaveChangesAsync();
}

```

Rysunek 6 Metoda Update()

**Metoda Update()** aktualizuje dane motocykla w bazie danych na podstawie danych zawartych w obiekcie UpdateMotor. Sprawdza istnienie motocykla, a następnie aktualizuje jego dane i zapisuje zmiany.

```
1 odwołanie
public async Task<DetailMotor> Detail(int id)
{
    var motor = await _context.Motors.FirstOrDefaultAsync(motor => motor.Id == id);

    if (motor is null) throw new NotFoundException(id);

    return new DetailMotor
    {
        Id = motor.Id,
        Brand = motor.Brand,
        Name = motor.Name,
        RentPrice = motor.RentPrice,
        Description = motor.Description,
        RequiredLicence = motor.RequiredLicence.ToString(),
        RentTo = motor.RentTo.ToString() ?? "",
        Reservation = motor.Reservation ? "Reserved" : "Not reserved"
    };
}
```

Rysunek 7 Metoda Detail()

**Metoda Detail()** zwraca szczegółowe informacje o motocyklu na podstawie jego identyfikatora. Jeśli motocykl nie istnieje, rzucony jest wyjątek NotFoundException.

```
1 odwołanie
public async Task<List<DetailMotor>> GetAll()
{
    var motors = await _context.Motors.ToListAsync();
    var list = new List<DetailMotor>();

    foreach (var motor in motors)
    {
        list.Add(new DetailMotor
        {
            Id = motor.Id,
            Brand = motor.Brand,
            Name = motor.Name,
            RentPrice = motor.RentPrice,
            Description = motor.Description,
            RequiredLicence = motor.RequiredLicence.ToString(),
            RentTo = motor.RentTo.ToString() ?? "",
            Reservation = motor.Reservation ? "Reserved" : "Not reserved"
        });
    }

    return list;
}
```

Rysunek 8 Metoda GetAll()

**Metoda GetAll()** zwraca listę wszystkich motocykli z bazy danych w formie listy obiektów DetailMotor. Iteruje przez motocykle, tworząc obiekty DetailMotor dla każdego i dodając je do listy.

```
1 odwołanie
public async Task<List<DetailMotor>> GetSelected(string brandString)
{
    var motors = await _context.Motors.Where(motor => motor.Brand.Contains(brandString)).ToListAsync();
    var list = new List<DetailMotor>();

    foreach (var motor in motors)
    {
        list.Add(new DetailMotor
        {
            Id = motor.Id,
            Brand = motor.Brand,
            Name = motor.Name,
            RentPrice = motor.RentPrice,
            Description = motor.Description,
            RequiredLicence = motor.RequiredLicence.ToString(),
            RentTo = motor.RentTo.ToString() ?? "",
            Reservation = motor.Reservation ? "Reserved" : "Not reserved"
        });
    }

    return list;
}
```

Rysunek 9 Metoda GetSelected()

**Metoda GetSelected()** zwraca listę motocykli z bazy danych pasujących do podanej marki. Iteruje przez motocykle, sprawdzając dopasowanie marki i tworząc obiekty DetailMotor dla pasujących motocykli.

```

1 odwołanie
public async Task Reserve(int id)
{
    var motor = await _context.Motors.FirstOrDefaultAsync(motor => motor.Id == id);
    if (motor is null) throw new NotFoundException(id);
    if (motor.Reservation) throw new MotorbikeReservedException();
    motor.Reservation = true;
    await _context.SaveChangesAsync();
}

```

*Rysunek 10 Metoda Reserve()*

**Metoda Reserve()** rezerwuje motocykl na podstawie jego identyfikatora. Sprawdza, czy motocykl istnieje i czy nie jest już zarezerwowany. Jeśli tak, ustawia flagę rezerwacji na true.

```

1 odwołanie
public async Task CancelReserve(int id)
{
    var motor = await _context.Motors.FirstOrDefaultAsync(motor => motor.Id == id);
    if (motor is null) throw new NotFoundException(id);
    if (!motor.Reservation) throw new MotorbikeNotReservedException();
    motor.Reservation = false;
    await _context.SaveChangesAsync();
}

```

*Rysunek 11 Metoda CancelReserve()*

**Metoda CancelReserve()** anuluje rezerwację motocykla na podstawie jego identyfikatora. Sprawdza, czy motocykl istnieje i czy jest zarezerwowany. Jeśli tak, ustawia flagę rezerwacji na false.

```

1 odwołanie
public async Task Rent(int id, int numberOfDays)
{
    var motor = await _context.Motors.FirstOrDefaultAsync(motor => motor.Id == id);
    if (motor is null) throw new NotFoundException(id);
    var rentDate = DateTime.Now.AddDays(numberOfDays);
    if (motor.RentTo != null && motor.RentTo > DateTime.Now)
        throw new MotorbikeCannotBeRentException(motor.RentTo ?? DateTime.Now);
    motor.RentTo = rentDate;
    motor.Reservation = false;
    await _context.SaveChangesAsync();
}

```

*Rysunek 12 Metoda Rent()*

**Metoda Rent()** wynajmuje motocykl na określoną liczbę dni na podstawie identyfikatora motocykla i liczby dni. Sprawdza, czy motocykl istnieje i czy jest dostępny do wynajęcia. Jeśli tak, ustawia datę zakończenia wynajmu dodatkowo zmienia flagę rezerwacji na false.

```

[return]
public async Task<byte[]> GeneratePDF(int id)
{
    var motor = await _context.Motors.FirstOrDefaultAsync(motor => motor.Id == id);
    if (motor is null) throw new NotFoundException(id);
    if (motor.RentTo is null || motor.RentTo < DateTime.Now) throw new ThisMotorBikeIsNotRentedException();
    string InvoiceName = $"Invoice {DateTime.Now.Year}[DateTime.Now.Month][DateTime.Now.Day]" +
        $"[{DateTime.Now.Hour}[DateTime.Now.Minute][DateTime.Now.Second].pdf";
    try
    {
        QuestPDF.Settings.License = LicenseType.Community;
        Document.Create(container =>
        {
            container.Page(page =>
            {
                page.Margin(50);
                page.Size(PageSizes.A4);
                page.BackgroundColor(Colors.White);
                page.DefaultTextStyle(x => x.FontSize(16));
                page.Header().AlignCenter().Text($"{motor.Brand}").Bold().FontSize(20).FontColor(Colors.Grey.Darken4);
                page.Content().Column(col =>
                {
                    col.Item().Table(table =>
                    {
                        table.ColumnsDefinition(columns =>
                        {
                            columns.ConstantColumn(30);
                            columns.RelativeColumn();
                            columns.RelativeColumn();
                        });
                        table.Header(header =>
                        {
                            header.Cell().Text("#");
                            header.Cell().Text("Motorbike");
                            header.Cell().AlignRight().Text("Rent Price");
                        });
                        table.Cell().Text(motor.Id.ToString());
                        table.Cell().Text($"{motor.Name}");
                        table.Cell().AlignRight().Text($"{motor.RentPrice} zł");
                    });
                    col.Item().Translate(4).LineHorizontal(2);
                    col.Item().Translate(20).Text($"{motor.Description}");
                });
            });
            GeneratePdf(InvoiceName);
        });
        return File.ReadAllBytes(InvoiceName);
    }
    catch
    {
        throw new CouldNotCreateInvoiceException(id);
    }
}

```

Rysunek 13 Metoda GeneratePDF()

**Metoda GeneratePDF()** generuje fakturę PDF dla wynajętego motocykla na podstawie jego identyfikatora. Tworzy dokument PDF za pomocą biblioteki QuestPDF, zawierający szczegóły motocykla oraz datę i godzinę generowania faktury.

## 2.2. Klient

Do realizacji klienta wykorzystano język programowania Python oraz bibliotekę Tkinter pozwalającą na tworzenie prostych, interaktywnych aplikacji okienkowych. Aplikacja powstała z myślą o pracownikach wypożyczalni i dostarcza im niezbędne narzędzia do komunikowania się z serwerem. Po uruchomieniu klienta aplikacja wykorzystuje certyfikat do połączenia się z serwerem (Rys. 14). Po udanej weryfikacji otwiera się główne okno (Rys. 15) z tabelą zawierającą wszystkie motocykle (wykorzystanie metody GetAll() Rys. 8) oraz przyciski i pola tekstowe pozwalające na korzystanie z usług dostarczanych przez serwer. W pierwszym wierszu znajdują się kontrolki do filtrowania po marce (wykorzystanie metody GetSelecte() Rys. 9) oraz przycisk do dodawania nowego motocykla (Wykorzystanie metody Create() Rys. 4). Na rysunku 16 przedstawiono widok tabeli po zastosowaniu filtrowania po marce „KTM”. Rysunek 17 przedstawia okno dodawania motocykla, jeśli operacja powiedzie się użytkownik najpierw otrzyma komunikat (Rys. 18), a następnie zostanie wywołana metoda odświeżająca widok tabeli motocykli (Rys. 19). Przycisk *Edytuj* otworzy okno edycji widoczne na rysunku 20, jeśli użytkownik wprowadzi zmiany i je zatwierdzi za pomocą metody Update() obiekt zostanie zaktualizowany w bazie danych. Rysunki 21 oraz 22 przedstawiają przykładowe komunikaty błędów obsługiwane przez middleware (Rys. 2). Przycisk *Wypożycz* otworzy okno, w którym użytkownik może podać na ile dni chce wypożyczyć motocykl (Rys. 23), po zatwierdzeniu przyciskiem *Wypożycz* zostanie wywołana metoda Rent() (Rys. 12). Przyciski *Rezerwuj* i *Anuluj Rezerwację* obsługują metody Reserve() i CancelReserve() (Rys. 10 i Rys. 11). Rysunek 24 przedstawia okno szczegółowych informacji o motocyklu gdzie oprócz podstawowych

danych z głównej tabeli zawarty jest opis danej maszyny. Przycisk *PDF* w głównej tabeli odpowiada za obsługę metody `GeneratePDF()`. Po jego naciśnięciu generowany jest plik pdf po stronie serwera, który następnie wysyłany jest w postaci binarny do klienta. Klient zapisuje plik pdf na maszynie użytkownika, po czym wyświetla komunikat (Rys. 25) i otwiera pobrany plik (Rys. 26).

```
session = requests.Session()
session.verify = "./cert.crt"
self.client = Client(wsd1="https://localhost:7107/Motor.wsd1", transport=Transport(session=session))
```

Rysunek 14 Weryfikacja klienta na podstawie certyfikatu.

Motorcycle Rental App													
Dodaj motocykl		Filtruj po marce:		Filtruj									
ID	Marka	Nazwa	Wymagane prawo jazdy	Wynajęty do	Rezerwacja	Cena za dobę	Opcje						
1	Honda	PCX 125	B_A1	14.05.2024 18:58:58	Not reserved	131 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
2	KTM	ADVENTURE 390 SW	A2		Not reserved	190 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
3	KTM	Duke 390	A2		Reserved	170 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
4	KTM	790 ADVENTURE	A		Reserved	240 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
5	KTM	1290 SUPER ADVENTURE R	A	13.05.2024 19:14:47	Reserved	290 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
6	Honda	CB 650 R	A		Not reserved	190 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
7	Honda	X-ADV	A2		Not reserved	190 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
8	Honda	XL750 Transalp	A		Not reserved	210 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
9	Honda	CRF1100L Africa Twin	A		Not reserved	280 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
10	Yamaha	NMAX 125	B_A1		Not reserved	130 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
11	Yamaha	TENIERE 700	A		Not reserved	210 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
12	Yamaha	R3	A2		Not reserved	170 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
13	Yamaha	XSR125 LEGACY	B_A1		Not reserved	140 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
14	BMW	G 310 GS	A2		Not reserved	160 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
15	BMW	F 900 GS	A2		Not reserved	230 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF

Rysunek 15 Główne okno aplikacji klienta.

Motorcycle Rental App													
Dodaj motocykl		Filtruj po marce: KTM		Filtruj									
ID	Marka	Nazwa	Wymagane prawo jazdy	Wynajęty do	Rezerwacja	Cena za dobę	Opcje						
2	KTM	ADVENTURE 390 SW	A2		Not reserved	190 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
3	KTM	Duke 390	A2		Reserved	170 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
4	KTM	790 ADVENTURE	A		Reserved	240 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF
5	KTM	1290 SUPER ADVENTURE R	A	13.05.2024 19:14:47	Reserved	290 zł	Usuń	Edytuj	Szczegóły	Rezerwuj	Anuluj rezerwację	Wypożycz	PDF

Rysunek 16 Główne okno po zastosowaniu filtrowania na podstawie marki "KTM".

Rysunek 17 Okienko Dodaj motocykl.

Rysunek 18 Potwierdzenie udania się operacji.

ID	Marka	Nazwa	Wymagane prawo jazdy	Wynajęty do	Rezerwacja	Cena za dobę	Opcje
2	KTM	ADVENTURE 390 SW	A2		Not reserved	190 zł	Usuń Edytuj Szczegóły Rezerwuj Anuluj rezerwację Wypożycz PDF
3	KTM	Duke 390	A2		Reserved	170 zł	Usuń Edytuj Szczegóły Rezerwuj Anuluj rezerwację Wypożycz PDF
4	KTM	790 ADVENTURE	A		Reserved	240 zł	Usuń Edytuj Szczegóły Rezerwuj Anuluj rezerwację Wypożycz PDF
5	KTM	1290 SUPER ADVENTURE R	A	13.05.2024 19:14:47	Reserved	290 zł	Usuń Edytuj Szczegóły Rezerwuj Anuluj rezerwację Wypożycz PDF
1018	KTM	TEST	A		Not reserved	130 zł	Usuń Edytuj Szczegóły Rezerwuj Anuluj rezerwację Wypożycz PDF

Rysunek 19 Odświeżona tabela z motocyklami z filtrem "KTM"

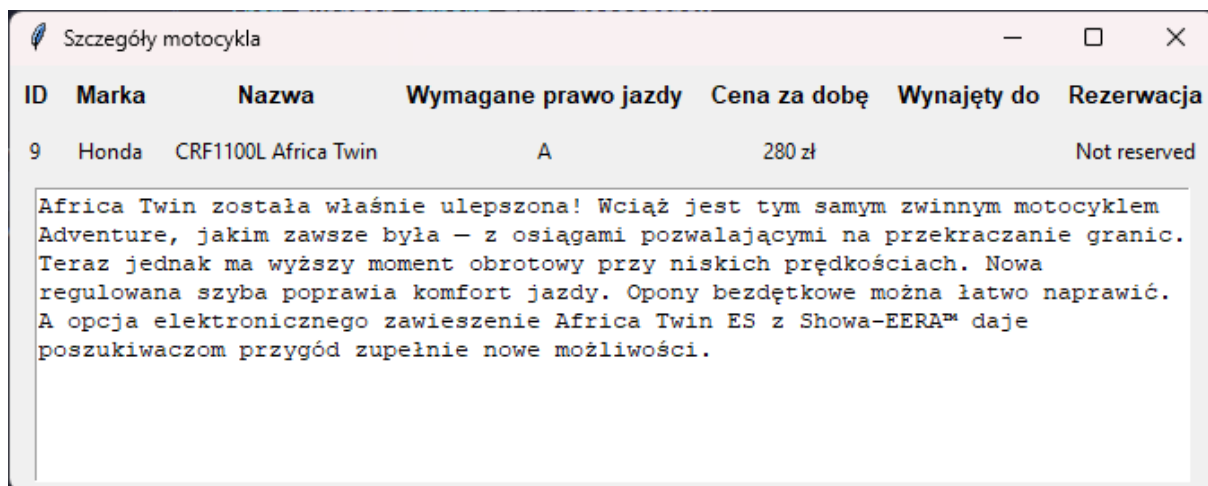
Rysunek 20 Okno edycji motocykla

Rysunek 21 Komunikat o błędzie generowania PDF.

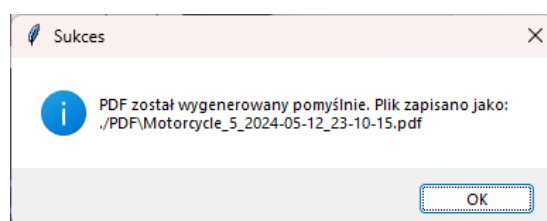
Rysunek 22 Komunikat błędu rezerwacji motocykla.

Rysunek 23 Okno wypożyczenia motocykla.

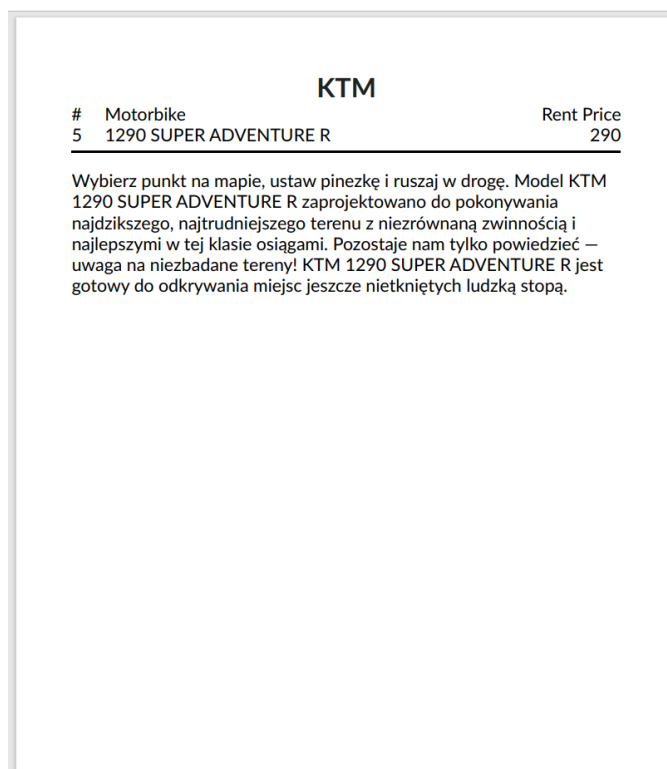




Rysunek 24 Okno szczegółowych informacji o motocyklu.

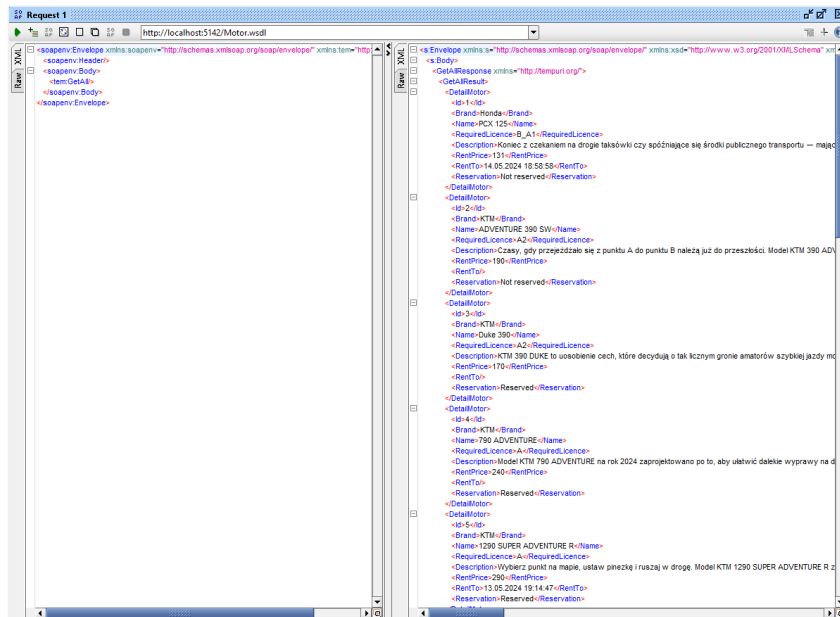


Rysunek 25 Komunikat o pomyślnym wygenerowaniu i zapisaniu PDF.

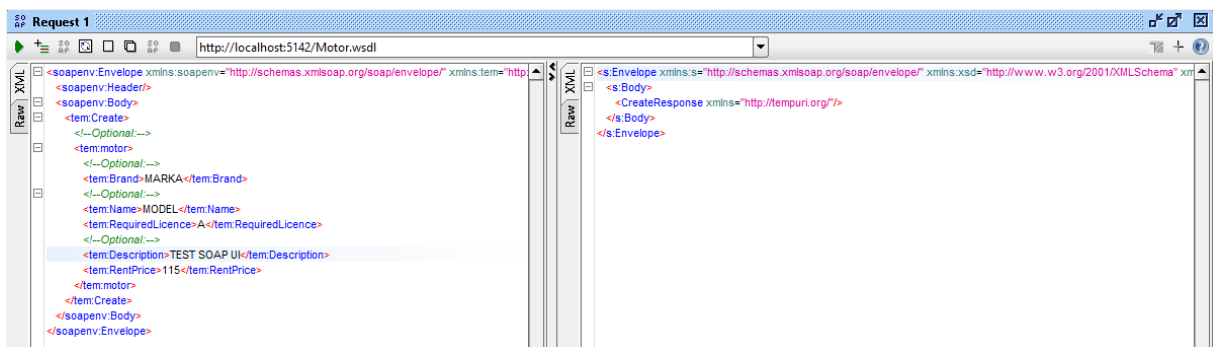


Rysunek 26 Wygenerowany PDF.

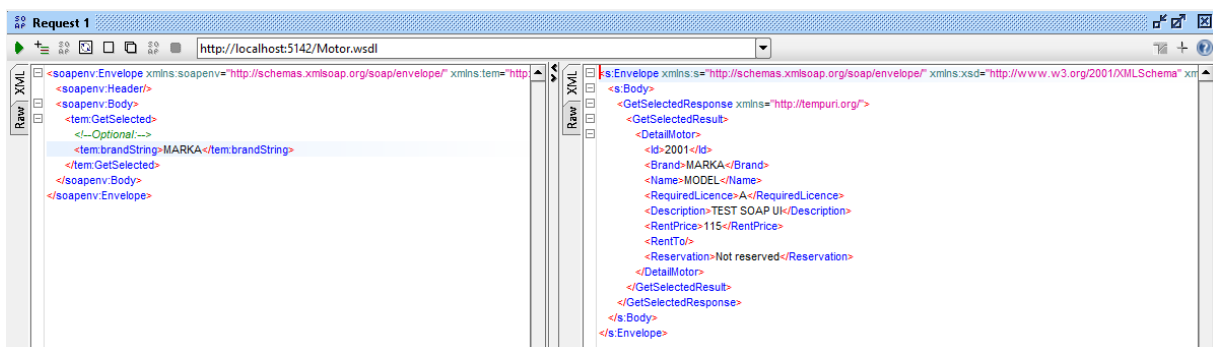
## 2.2 Testowanie WS w aplikacji SoapUI



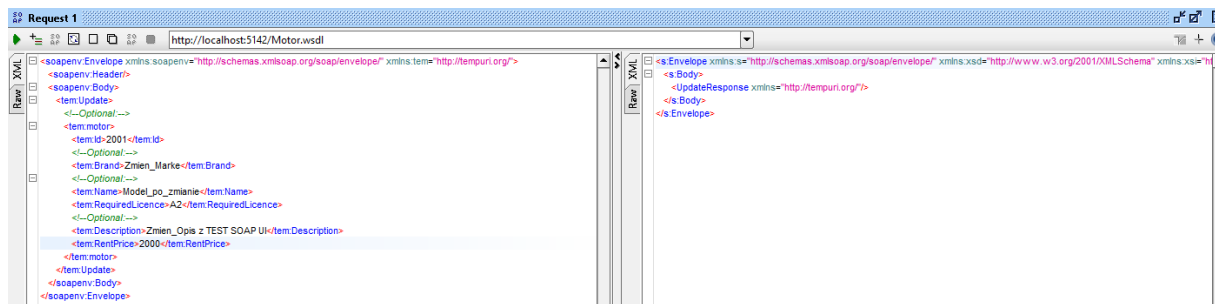
Rysunek 27 Usługa GetAll().



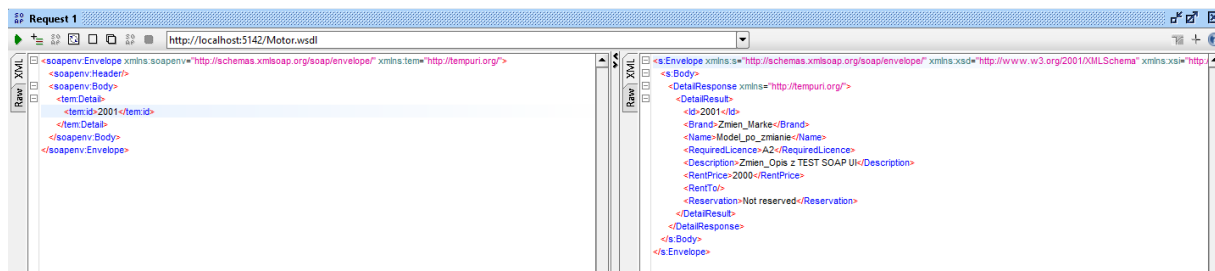
Rysunek 28 Stworzenie obiektu za pomocą Create().



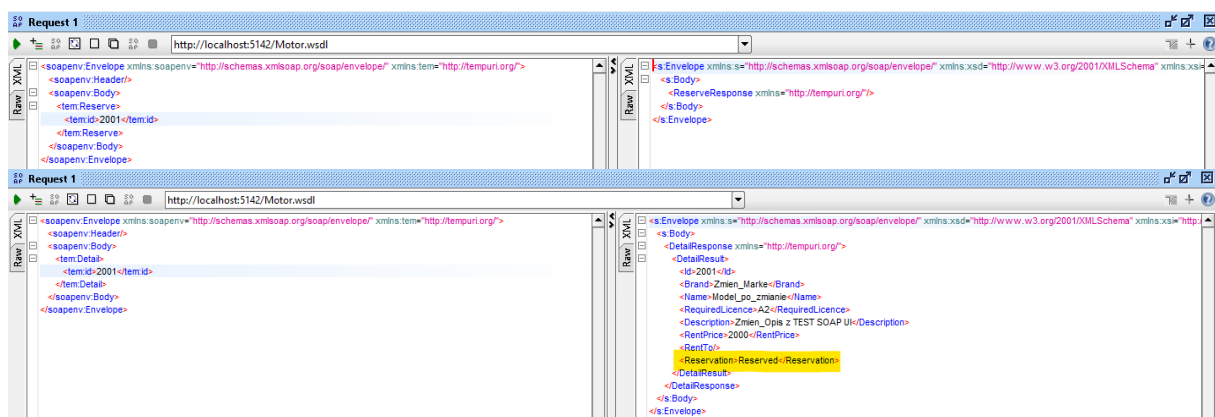
Rysunek 29 Wyszukanie stworzonego obiektu za pomocą GetSelected().



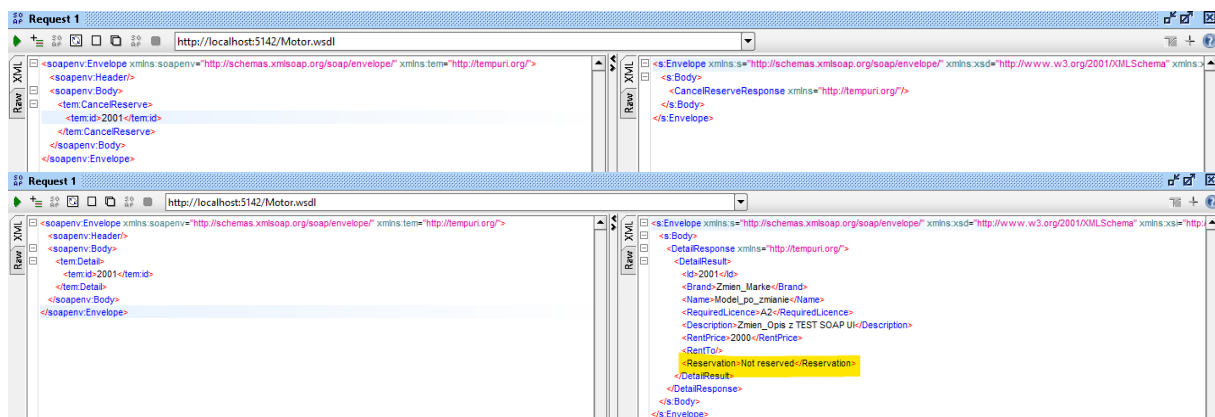
Rysunek 30 Zmiana wartości atrybutów za pomocą Update() wcześniej utworzonego motocykla.



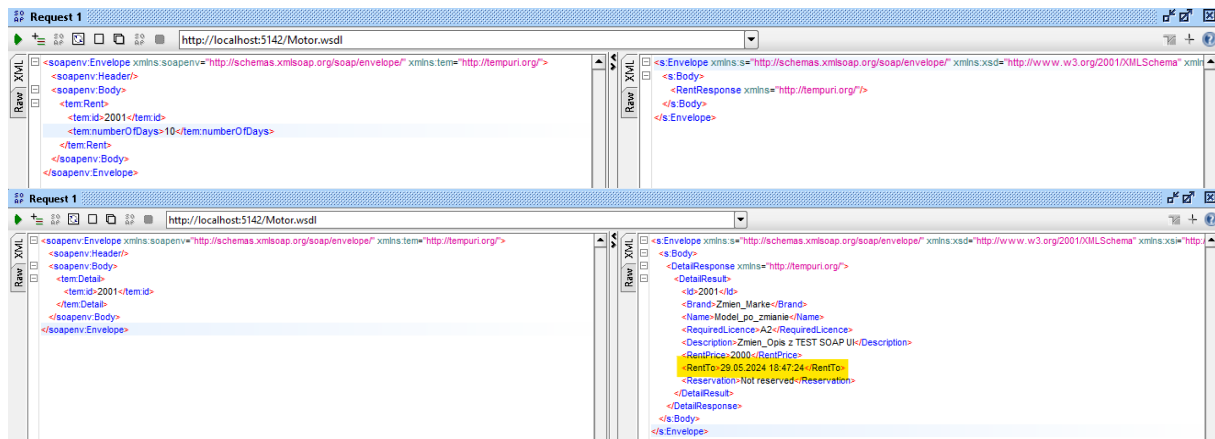
Rysunek 31 Użycie Detail() w celu wypisania szczegółowych informacji o naszym motocyklu.



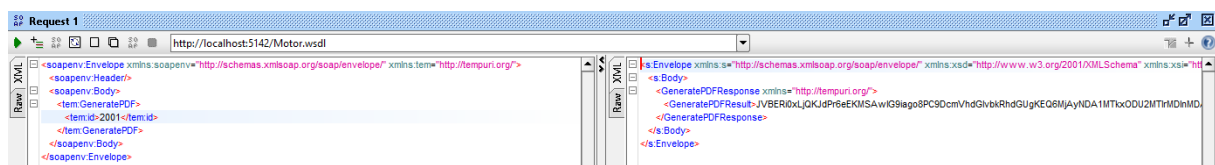
Rysunek 32 Zarezerwowanie motocykla przy pomocy Reserve() oraz sprawdzenie zmian przy pomocy Detail().



Rysunek 33 Rezygnacja z rezerwacji motocykla przy pomocy CancelReserve() oraz sprawdzenie zmian przy pomocy Detail().



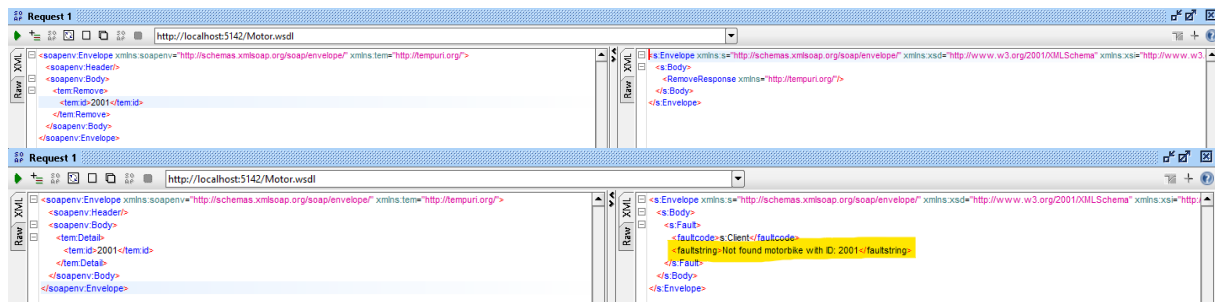
Rysunek 34 Wyporzczenie motocykla za pomcą Rent() na 10 dni oraz sprawdzenie zmian przy pomocy Detail().



Rysunek 35 Wygenerowanie PDF dla wynajętego motocykla GeneratePDF().



Rysunek 36 Przetestowanie dzialania middleware .NET (handlers).



Rysunek 37 Usunięcie motocykla Remove() i sprawdzenie zmian za pomocą Detail().

## 2.4 WSDL

```
<wSDL:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://tempuri.org"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.microsoft.com/ws/06/2004/policy/http"
xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://tempuri.org"
name="IMotorService">
  <wSDL:types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <!-- Element Create, zawiera obiekt motor typu CreateMotor -->
      <xsd:element name="Create">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element minOccurs="0" maxOccurs="1" name="motor" type="tns:CreateMotor"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="CreateResponse">
        <xsd:complexType/>
      </xsd:element>

      <!-- Element Remove, zawiera id typu int -->
      <xsd:element name="Remove">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element minOccurs="1" maxOccurs="1" name="id" type="xsd:int"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="RemoveResponse">
        <xsd:complexType/>
      </xsd:element>

      <!-- Element Update, zawiera obiekt motor typu UpdateMotor -->
      <xsd:element name="Update">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element minOccurs="0" maxOccurs="1" name="motor" type="tns:UpdateMotor"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="UpdateResponse">
        <xsd:complexType/>
      </xsd:element>

      <!-- Element Detail, zawiera id typu int -->
      <xsd:element name="Detail">
        <xsd:complexType>
```

```

    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="id" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="DetailResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" name="DetailResult" type="tns:DetailMotor"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- Element GetAll, bez parametrów wejściowych -->
<xsd:element name="GetAll">
  <xsd:complexType/>
</xsd:element>

<xsd:element name="GetAllResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" name="GetAllResult" nillable="true"
type="tns:ArrayOfDetailMotor"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- Element GetSelected, zawiera brandString typu string -->
<xsd:element name="GetSelected">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" name="brandString" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="GetSelectedResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" name="GetSelectedResult" nillable="true"
type="tns:ArrayOfDetailMotor"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- Element Reserve, zawiera id typu int -->
<xsd:element name="Reserve">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="id" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ReserveResponse">
  <xsd:complexType/>
</xsd:element>

<!-- Element CancelReserve, zawiera id typu int -->
<xsd:element name="CancelReserve">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="id" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

```

<xsd:element name="CancelReserveResponse">
  <xsd:complexType/>
</xsd:element>

<!-- Element Rent, zawiera id i numberOfDays typu int -->
<xsd:element name="Rent">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="id" type="xsd:int"/>
      <xsd:element minOccurs="1" maxOccurs="1" name="numberOfDays" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RentResponse">
  <xsd:complexType/>
</xsd:element>

<!-- Element GeneratePDF, zawiera id typu int -->
<xsd:element name="GeneratePDF">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="id" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="GeneratePDFResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" name="GeneratePDFResult"
type="xsd:base64Binary"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- Definicje typów kompleksowych -->
<xsd:complexType name="CreateMotor">
  <xsd:sequence>
    <xsd:element minOccurs="0" maxOccurs="1" name="Brand" type="xsd:string"/>
    <xsd:element minOccurs="0" maxOccurs="1" name="Name" type="xsd:string"/>
    <xsd:element minOccurs="1" maxOccurs="1" name="RequiredLicence" type="tns:Licence"/>
    <xsd:element minOccurs="0" maxOccurs="1" name="Description" type="xsd:string"/>
    <xsd:element minOccurs="1" maxOccurs="1" name="RentPrice" type="xsd:int"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="UpdateMotor">
  <xsd:sequence>
    <xsd:element minOccurs="1" maxOccurs="1" name="Id" type="xsd:int"/>
    <xsd:element minOccurs="0" maxOccurs="1" name="Brand" type="xsd:string"/>
    <xsd:element minOccurs="0" maxOccurs="1" name="Name" type="xsd:string"/>
    <xsd:element minOccurs="1" maxOccurs="1" name="RequiredLicence" type="tns:Licence"/>
    <xsd:element minOccurs="0" maxOccurs="1" name="Description" type="xsd:string"/>
    <xsd:element minOccurs="1" maxOccurs="1" name="RentPrice" type="xsd:int"/>
  </xsd:sequence>
</xsd:complexType>

<!-- Definicje bindowania operacji -->
<wsdl:binding name="BasicHttpBinding_IMotorService_soap" type="tns:IMotorService">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>

  <wsdl:operation name="Create">
    <soap:operation soapAction="http://tempuri.org/IMotorService/Create" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>

```

```

</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Remove">
<soap:operation soapAction="http://tempuri.org/IMotorService/Remove" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Update">
<soap:operation soapAction="http://tempuri.org/IMotorService/Update" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Detail">
<soap:operation soapAction="http://tempuri.org/IMotorService/Detail" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetAll">
<soap:operation soapAction="http://tempuri.org/IMotorService/GetAll" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetSelected">
<soap:operation soapAction="http://tempuri.org/IMotorService/GetSelected" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Reserve">
<soap:operation soapAction="http://tempuri.org/IMotorService/Reserve" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="CancelReserve">
<soap:operation soapAction="http://tempuri.org/IMotorService/CancelReserve" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>

```



```
</wsdl:operation>
<wsdl:operation name="Rent">
<soap:operation soapAction="http://tempuri.org/IMotorService/Rent" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GeneratePDF">
<soap:operation soapAction="http://tempuri.org/IMotorService/GeneratePDF" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<!-- Definicje portu -->
<wsdl:service name="IMotorService">
<wsdl:port name="BasicHttpBinding_IMotorService_soap"
binding="tns:BasicHttpBinding_IMotorService_soap">
<soap:address location="https://localhost:7107/Motor.wsdl"/>
```

### 3. Wnioski

W ramach projektu wykonano system oparty na web serwisach SOAP, który nie tylko spełnia założenia co do funkcjonalności rezerwacji i wynajmu motocykli, ale również dostarcza solidne fundamenty pod kontem skalowalności, bezpieczeństwa oraz wydajności co czyni go wartościowym narzędziem dla wypożyczalni motocykli.

#### Co zrobiono w ramach projektu:

- Stworzono web serwis
- Stworzono klienta w formie aplikacji okienkowej
- Stworzono dokumentację projektu
- Wykorzystano wysyłanie plików binarnych (PDF)
- Użyto Handlers
- **Dodatkowy punkt** - użycie szyfrowania SSL/TLS (konsumowanie WS przez https)
- **Dodatkowy punkt** – napisanie klienta w innym języku niż java (Chodzi o aby WS i klient byli w różnych językach)