P O L I T E C H N I K A R Z E S Z O W S K A
im. Ignacego Łukasiewicza
WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

Tomasz Kozioł, Wiktor Misiaszek

Wprowadzenie do programowania w języku python

„PORÓWNANIE CZASÓW WYSYŁANIA DANYCH, PRZY UŻYCIU RÓŻNYCH INTERFEJSÓW W PYTHONIE”

Rzeszów 2023

Spis treści

1. Opis projektu.

W projekcie wykorzystliśmy mikroserwis z informacjami o filmach, takich jak tytuł, opis, dane reżysera oraz gatunek. Mikroserwis ten działa w oparciu o pakiet XAMPP i pozwala na porównanie czasów wysyłania danych przy użyciu REST, SOAP oraz gRPC.

2. Opis użytych interfejsów.

- **REST** (Representational State Transfer) to architektura sieciowa, która pozwala na przesyłanie danych między różnymi systemami. Jest to proste i przyjazne dla użytkownika rozwiązanie, które opiera się na protokole HTTP i pozwala na przesyłanie danych w formacie JSON lub XML. REST jest lekkie, łatwe do implementacji i skalowania oraz pozwala na łatwą integrację z innymi systemami.

- **SOAP** (Simple Object Access Protocol) to protokół komunikacji sieciowej, który pozwala na przesyłanie danych między różnymi systemami. SOAP opiera się na protokole HTTP lub HTTPS i pozwala na przesyłanie danych w formacie XML. SOAP jest bardziej skomplikowany niż REST, ale daje więcej możliwości konfiguracji i zabezpieczeń, co czyni go bardziej odpowiednim dla rozbudowanych i złożonych systemów.

- **gRPC** to nowszy protokół komunikacji sieciowej stworzony prze Google, który pozwala na przesyłanie danych między różnymi systemami. gRPC opiera się na protokole HTTP/2 i pozwala na przesyłanie danych w formacie protobuf, co pozwala na szybszą komunikację niż SOAP i REST. gRPC jest skalowalne, ma lepsze zabezpieczenia, a także pozwala na tworzenie aplikacji z wykorzystaniem języków, takich jak C++, Java, Python, czy Go.

3. Użyte technologie.

W projekcie użyliśmy następujących technologii:

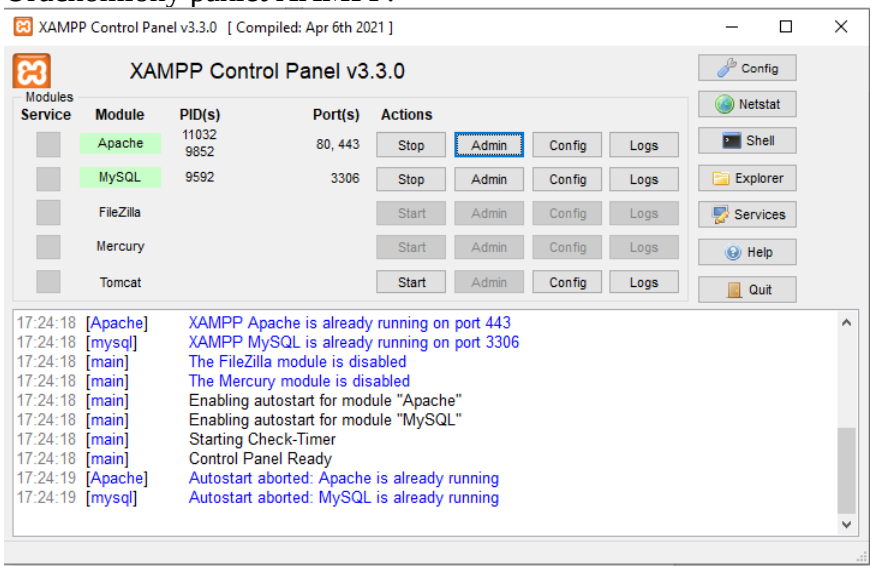
- XAMPP
- Flask
- REST
- SOAP
- gRPC

4. Instrukcje prowadzące do poprawnego działania programu.

- Pobieramy pythona
- Otwieramy cmd, następnie przechodzimy do katalogu naszego projektu i wpisujemy poniższą komendę, aby zainstalować wszystkie potrzebne paczki użyte w porgramie.
- `pip install -r requirements.txt`
- Pobieramy XAMPP, uruchamiamy na nim serwer Apache oraz bazę danych MySQL, po czym logujemy na panel administratora i dodajemy nową bazę danych o nazwie ‘python’ z metodą porównywania napisów ‘`utf8mb4_general_ci`’, aby wszystkie rekordy do bazy wczytały się bez problemu.
- Uruchamiamy plik ‘create_db_only.py’, który dodaje rekordy do naszej bazy danych.
- Uruchamiamy plik ‘server.py’.
- Uruchamiamy plik ‘client.py’.
- Wyniki są zawarte w konsoli oraz na wykresie.

5. Baza danych

Uruchomiony pakiet XAMPP.



Połączenie z bazą danych.

```python
class Database:
    def __init__(self, user='root', password='', host='localhost', database='python'):
        try:
            #Połączenie z bazą danych MySQL z XAMPP
            self.cnx = mysql.connector.connect(
                user=user,
                password=password,
                host=host,
                database=database
            )
            #Utwórz kursor - operacje na bazie danych
            self.cursor = self.cnx.cursor()
            print('Utworzono połączenie z bazą danych')
        except:
            print('Nie udało się nawiązać połączenia z bazą danych')
```

Utworzenie nowych tabel.

```python
def create_tables(self):
    try:
        #Utworz tabele
        self.cursor.execute("CREATE TABLE movies (id INT AUTO_INCREMENT PRIMARY KEY, title VARCHAR(255), description VARCHAR(255))")
        self.cursor.execute("CREATE TABLE directors (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), surname VARCHAR(255))")
        self.cursor.execute("CREATE TABLE genres (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255))")

        #Utworz relacje
        self.cursor.execute("ALTER TABLE movies ADD director_id INT, ADD FOREIGN KEY (director_id) REFERENCES directors(id)")
        self.cursor.execute("ALTER TABLE movies ADD genre_id INT, ADD FOREIGN KEY (genre_id) REFERENCES genres(id)")

        #Zapisz w bazie
        self.cnx.commit()
        print('Utworzono tabelę')
    except:
        print('Nie można było utworzyć tabeli')
```

Dodanie filmu do naszej bazy.

```python
def add_movie(self, title, description, directorName, directorSurname, genre):
    #Sprawdzanie czy tytuł już istnieje
    self.cursor.execute("SELECT * FROM movies WHERE title = %s", (title,))
    if self.cursor.fetchone():
        print("Film o takim tytule już istnieje.")
        return

    #Sprawdzanie czy reżyser już istnieje
    self.cursor.execute("SELECT id FROM directors WHERE name = %s AND surname = %s", (directorName,directorSurname))
    director_id = self.cursor.fetchone()

    #Przygotowanie zmiennej
    try:
        director_id = director_id[0]
    except:
        pass

    if not director_id:
        #Dodawanie reżysera do tabeli
        self.cursor.execute("INSERT INTO directors (name, surname) VALUES (%s, %s)", (directorName,directorSurname))
        director_id = self.cursor.lastrowid
        print('Dodanie nowego reżysera do tabeli')
    else:
        print('Skorzystanie z istniejącego reżysera w tabeli')

    #Sprawdzanie czy gatunek już istnieje
    self.cursor.execute("SELECT id FROM genres WHERE name = %s", (genre,))
    genre_id = self.cursor.fetchone()

    #Przygotowanie zmiennej
    try:
        genre_id = genre_id[0]
    except:
        pass

    if not genre_id:
        #Dodawanie gatunku do tabeli
        self.cursor.execute("INSERT INTO genres (name) VALUES (%s)", (genre,))
        genre_id = self.cursor.lastrowid
        print('Dodanie nowego gatunku do tabeli')
    else:
        print('Skorzystanie z istniejącego gatunku w tabeli')

    #Dodawanie filmu do tabeli z odpowiednim reżyserem i gatunkiem
    self.cursor.execute("INSERT INTO movies (title, description, director_id, genre_id) VALUES (%s, %s, %s, %s)", (title, description, director_id, genre_id))
    self.cnx.commit()
    print('Film dodany z powodzeniem')
```

Wczytanie rekordów do bazy danych.

```python
from database.database import Database

if __name__ == "__main__":
    dataBase = Database()
    dataBase.create_tables()
    dataBase.add_movie('The Shawshank Redemption',"Historia dwóch więźniów, Andy'ego i Reda, którzy przebywają w więzieniu Shawshank. Andy próbuje pomóc swoim towarzyszom więzienia, podcz
    dataBase.add_movie('The Godfather',"Historia rodziny Corleone, mafijnej rodziny z Nowego Jorku, która próbuje utrzymać swój wpływ i pozycję",'Francis','Ford Coppola','Gangsterski')
    dataBase.add_movie('The Dark Knight',"Batman staje do walki z przestępczym genialnym psychopatą Jokerem, który chce zniszczyć Gotham City",'Christopher','Nolan','Kryminał')
    dataBase.add_movie('Pulp Fiction',"Historie kilku osób, w tym pary mafiosów, pary gangusterów, kilku bohaterów przypadkowych i trójki hakerów, których losy splatają się w Los Angeles"
    dataBase.add_movie("Schindler's List","Historia biznesmena Oskara Schindlera, który próbuje ratować żydowskich pracowników podczas Holocaustu",'Steven','Spielberg','Wojenny')
    dataBase.add_movie("The Matrix","Film science-fiction opowiadający o rzeczywistości, w której ludzkość jest uwięziona w wirtualnej rzeczywistości przez maszyny, które rządzą światem",
    dataBase.add_movie("The Lord of the Rings: The Fellowship of the Ring","Film fantasy opowiadający o hobbie imieniem Frodo, który wyrusza w podróż, by zniszczyć pierścień, który daje j
    dataBase.add_movie("Forrest Gump","Historia życia Forresta, prostego człowieka z niedosłuchem, który przeżywa wiele ważnych wydarzeń w historii USA w XX wieku","Robert","Zemeckis","Ko
    dataBase.add_movie("Incepcja","Film opowiada o zespołach specjalistów, którzy wykorzystują tajemniczą technologię do wchodzenia do snów ludzi, celem manipulowania ich marzeniami i ich
    dataBase.add_movie("The Silence of the Lambs","Film sensacyjny opowiadający o śledztwie FBI w sprawie seryjnego mordercy, który zabija i zjada swoje ofiary, a także o agentce Clarice
    dataBase.add_movie("The Green Mile", "Film opowiadający o strażniku więziennym, który zaczyna wierzyć w nadprzyrodzoną moc skazańca oskarżonego o zabójstwo, który przebywa na karze śm
    dataBase.add_movie("The Prestige", "Film science fiction, opowiada o rywalizacji dwóch iluzjonistów, którzy próbują pokonać siebie nawzajem i zdobyć tajemnicę największego sztuczki",
    dataBase.add_movie("The Departed", "Film kryminalny opowiadający o agentach pod przykrywką, którzy próbują przeniknąć do irlandzkiej mafii w Bostonie", "Martin", "Scorsese", "Kryminal
    dataBase.add_movie("The Lion King", "Animowany film opowiadający o młodym lwie, Simbie, który musi odzyskać swoje miejsce na czele stada po śmierci ojca", "Roger", "Allers", "Animacja
    dataBase.add_movie("The Social Network", "Film opowiadający o powstaniu serwisu społecznościowego Facebook i jego twórcy Marku Zuckerbergu", "David", "Fincher", "Dramat")
    dataBase.add_movie("The Great Gatsby", "Film opowiadający o bogatym obywatelu Jay Gatsbym, który próbuje odzyskać swoją byłą miłość, Daisy Buchanan", "Baz", "Luhrmann", "Dramat")
    dataBase.add_movie("The Pianist", "Film opowiadający o życiu polskiego pianisty Władysława Szpilmana podczas II wojny światowej", "Roman", "Polanski", "Dramat")
    dataBase.add_movie("The Incredibles", "Animowany film opowiadający o rodzinie superbohaterów, która musi powrócić do walki z przestępcami po latach ukrywania się", "Brad", "Bird", "An
    dataBase.add_movie("The Big Lebowski", "Komediowy film noir opowiadający o nieporadnym, leniwym bowlerze Jeffie Leboswskim, który jest pomylony z bogatym Jeffem Leboswskim", "Joel", "
    dataBase.add_movie("The Terminator", "Film science fiction opowiadający o cyborgu, który przybywa z przyszłości, by zabić matkę przyszłego lidera rebelii przeciwko maszynom", "James",
    dataBase.add_movie("The Elephant Man", "Film opowiadający o życiu Josepha Merricka, człowieka z niezwykłymi wadami genetycznymi, który zostaje wykorzystany jako atrakcja w cyrku", "Da
    dataBase.add_movie("The Grand Budapest Hotel", "Film komediowy opowiadający o przygodach dyrektora hotelu i jego pomocnika w Europie międzywojennej", "Wes", "Anderson", "Komedia")
    dataBase.add_movie("The Thing", "Film science fiction opowiadający o załodze stacji badawczej na Antarktydzie, która musi walczyć z obcym, który przybywa z kosmosu", "John", "Carpente
    dataBase.add_movie("The Truman Show", "Film science fiction opowiadający o człowieku, którego całe życie jest transmitowane na żywo jako reality show, nie wiedząc o tym", "Peter", "We
    dataBase.add_movie("The Exorcist", "Film horror opowiadający o dziewczynce, która zostaje opętana przez demona i lekarze i księża próbują ją uwolnić", "William", "Friedkin", "Horror")
    dataBase.add_movie("The Godfather: Part II", "Kontynuacja historii rodziny Corleone, która koncentruje się zarówno na wydarzeniach z pierwszej części, jak i na młodości Don Corleone",
    dataBase.add_movie("The Shining", "Film horror opowiadający o rodzinie, która zostaje zamknięta w zimowym hotelu, gdzie zaczynają dziać się dziwne rzeczy", "Stanley", "Kubrick", "Horr
    dataBase.add_movie("La La Land", "Film musicalowy opowiadający o miłości między aktorem a jazzową pianistką w Los Angeles", "Damien", "Chazelle", "Musical")
    dataBase.add_movie("The Revenant", "Film przygodowy opowiadający o trapperze, który przemierza dziewiczą prerię, by się zemścić na swoim byłym przyjacielu", "Alejandro", "González Iñá
    dataBase.add_movie("Moonlight", "Film opowiadający o życiu młodego człowieka z Miami, który próbuje zrozumieć swoją tożsamość seksualną i relacje z rodziną", "Barry", "Jenkins", "Dram
    dataBase.add_movie("The Shape of Water", "Film fantasy opowiadający o przygodzie kobiety, która zakochuje się w potworze przetrzymywanym w laboratorium rządowym", "Guillermo del", "To
    dataBase.add_movie("Parasite", "Film kryminalny opowiadający o rodzinie biednych, która próbuje wkraść się do życia bogatej rodziny poprzez podstęp", "Bong Joon", "Ho", "Kryminalny")
    dataBase.add_movie("Nomadland", "Film opowiadający o kobiecie, która po utracie swojego domu i pracy, decyduje się na życie w vanie i podróżowanie po kraju jako nomadka", "Chloe", "Zh
    dataBase.add_movie("Promising Young Woman", "Film thriller opowiadający o kobiecie, która po traumie z przeszłości, przeprowadza swoiste dochodzenie w sprawie gwałtu", "Emerald", "Fen
    dataBase.add_movie("The Father", "Film dramatyczny opowiadający o relacji ojca z córką, gdy ten zaczyna chorować na demencję", "Florian", "Zeller", "Dramat")
    dataBase.add_movie("Sound of Metal", "Film opowiadający o perkusiście, który traci słuch i musi przestawić się na nowe życie", "Darius", "Marder", "Dramat")
    dataBase.add_movie("Minari", "Film opowiadający o rodzinie koreańskiej, która przeprowadza się na farmę w Arkansasie, by spełnić swoje marzenia", "Lee Isaac", "Chung", "Dramat")
    dataBase.add_movie("The Trial of the Chicago 7","Film opowiadający o procesie siedmiu aktywistów, oskarżonych o zamieszki podczas Demokratycznego Kongresu Narodowego w 1968 roku", "Aa
    dataBase.add_movie("Tenet","Film science fiction opowiadający o agenta specjalnego, który jest zmuszony do udziału w operacji, która ma zapobiec III wojnie światowej przy użyciu inwer
    dataBase.add_movie("The Prom", "Film musicalowy opowiadający o grupie aktorów, którzy przyjeżdżają do małego miasteczka, by pomóc parze homoseksualnej zakochanych, którzy nie mogą pój
    dataBase.add_movie("The French Dispatch", "Film komediowy opowiadający o grupie dziennikarzy, którzy próbują ujawnić prawdę o różnych wydarzeniach na świecie", "Wes", "Anderson", "Kom
    dataBase.close_connection()
```

Jak widać dane zostały wczytane prawidłowo.

	Tabela	Działanie	Rekordy	Typ	Metoda porównywania napisów	Rozmiar	Nadmiar
☐	directors	Przeglądaj Struktura Szukaj Wstaw Opróżnij Usuń	36	InnoDB	utf8mb4_general_ci	16.0 KB	-
☐	genres	Przeglądaj Struktura Szukaj Wstaw Opróżnij Usuń	13	InnoDB	utf8mb4_general_ci	16.0 KB	-
☐	movies	Przeglądaj Struktura Szukaj Wstaw Opróżnij Usuń	41	InnoDB	utf8mb4_general_ci	48.0 KB	-
	3 tabel	**Suma**	**90**	**InnoDB**	**utf8mb4_general_ci**	**80.0 KB**	**0 B**

6. Utworzenie serwera Flask oraz gRPC

Dla REST'a i SOAP korzystamy z Flaska.

```
#Flask
app = Flask(__name__)
```

„Podpinamy" REST'a z filmami w bazie danych.

```
@app.route('/rest/movies', methods=['GET'])
def get_movies():
    #Pobierz informacje z bazy danych
    database = Database()
    movies = database.get_table('movies')
    database.close_connection()
    movies_list = []

    #Sformatuj dane
    for movie in movies:
        key = ["id", "title", "description", "director_id", "genre_id"]
        values = movie
        movie_dictionary = dict(zip(key, values))
        movies_list.append(movie_dictionary)

    #Zwróć odpowiedź w postaci JSON
    return jsonify({'movies': movies_list})

@app.route('/rest/movie/<int:id>', methods=['GET'])
def get_movie(id):
    database = Database()
    movie = database.get_table('movies',str(id))
    database.close_connection()

    #Sformatuj dane
    key = ["id", "title", "description", "director_id", "genre_id"]
    try:
        values = movie[0]
    except:
        #Gdy nie ma w bazie danych zwróć błąd w postaci JSON
        error = jsonify({"error" : 404, "description" : "Movie not found"})
        return make_response(error, 404)

    movie_dictionary = dict(zip(key, values))

    #Zwróć odpowiedź w postaci JSON
    return jsonify({'movie': movie_dictionary})
```

„Podpinamy” REST’a z reżyserami w bazie danych.

```python
#REŻYSERZY
@app.route('/rest/directors', methods=['GET'])
def get_directors():
    #Pobierz informacje z bazy danych
    database = Database()
    directors = database.get_table('directors')
    database.close_connection()
    directors_list = []

    #Sformatuj dane
    for director in directors:
        key = ["id", "name", "surname"]
        values = director
        director_dictionary = dict(zip(key, values))
        directors_list.append(director_dictionary)

    #Zwróć odpowiedź w postaci JSON
    return jsonify({'directors': directors_list})

@app.route('/rest/director/<int:id>', methods=['GET'])
def get_director(id):
    database = Database()
    director = database.get_table('directors',str(id))
    database.close_connection()

    #Sformatuj dane
    key = ["id", "name", "surname"]
    try:
        values = director[0]
    except:
        #Gdy nie ma w bazie danych zwróć błąd w postaci JSON
        error = jsonify({"error" : 404, "description" : "Director not found"})
        return make_response(error, 404)

    director_dictionary = dict(zip(key, values))

    #Zwróć odpowiedź w postaci JSON
    return jsonify({'director': director_dictionary})
```

„Podpinamy” REST’a z gatunkami filmówymi w bazie danych.

```python
#GATUNKI
@app.route('/rest/genres', methods=['GET'])
def get_genres():
        #Pobierz informacje z bazy danych
        database = Database()
        genres = database.get_table('genres')
        database.close_connection()
        genres_list = []

        #Sformatuj dane
        for genre in genres:
            key = ["id", "name"]
            values = genre
            genre_dictionary = dict(zip(key, values))
            genres_list.append(genre_dictionary)

        #Zwróć odpowiedź w postaci JSON
        return jsonify({'genres': genres_list})

@app.route('/rest/genre/<int:id>', methods=['GET'])
def get_genre(id):
    database = Database()
    genre = database.get_table('genres',str(id))
    database.close_connection()

    #Sformatuj dane
    key = ["id", "name"]
    try:
        values = genre[0]
    except:
        #Gdy nie ma w bazie danych zwróć błąd w postaci JSON
        error = jsonify({"error" : 404, "description" : "Genre not found"})
        return make_response(error, 404)

    genre_dictionary = dict(zip(key, values))

    #Zwróć odpowiedź w postaci JSON
    return jsonify({'genre': genre_dictionary})
```

„Podpinamy” SOAP’a z bazą danych.

```python
#Podpięcie ścieżek
@app.route("/soap/movies", methods=['POST'])
def movies():
    return wsgi_application_movies

@app.route("/soap/directors", methods=['POST'])
def directors():
    return wsgi_application_directors

@app.route("/soap/genres", methods=['POST'])
def genres():
    return wsgi_application_genres
```

Dla gRPC tworzymy własny serwer.

```python
#Utwórz nowy obiekt serwera
server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))

#Implementacja metod
add_MoviesServiceServicer_to_server(Movies(), server)
add_DirectorsServiceServicer_to_server(Directors(), server)
add_GenresServiceServicer_to_server(Genres(), server)
```

Asychroniczne uruchomienie serwerów.

```python
#Asynchroniczne uruchomienie
async def server_flask():
    #Uruchomienie serwera
    print('Uruchamianie serwera Flask')
    app.run(debug=True)

async def server_grpc():
    #Uruchomienie serwera
    print('Uruchamianie serwera gRPC')
    server.add_insecure_port('[::]:50051')
    server.start()

async def main():
    await asyncio.gather(server_grpc(),server_flask())

if __name__ == "__main__":
    #Główna część programu
    asyncio.run(main())
```

7. Klient.

Inicjalizacja klienta.

```python
class Client:

    #Konfiguracja - Inicjalizacja
    def __init__(self, print_response_body=False, print_response_time=True, element_response="movies"):
        self.print_response_body=print_response_body
        self.print_response_time=print_response_time
        self.element_response=element_response
```

Wysłanie zapytania na serwer za pomocą interfejsu REST.

```python
def rest(self):
    print('---REST---')

    #Zapytanie do REST API
    start_time = time.time()
    response = requests.get('http://127.0.0.1:5000/rest/{}'.format(self.element_response))
    end_time = time.time()

    #Zapisz zmienną w obiekcie
    self.rest_time = end_time - start_time

    #Pokazanie otrzymanych danych
    if self.print_response_time == True:
        print('Response time: {}'.format(self.rest_time))

    if self.print_response_body == True:
        print(json.dumps(response.json(), indent=4))
```

Wysłanie zapytania na serwer za pomocą interfejsu SOAP.

```python
def soap(self):
    print('---SOAP---')

    #Przygotowanie XML zmiennej
    data = '''
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:{}="{}">
        <soapenv:Header/>
            <soapenv:Body>
                <{}:get_{}/>
            </soapenv:Body>
    </soapenv:Envelope>
    '''.format(self.element_response,self.element_response,self.element_response,self.element_response)

    #Zapytanie do SOAP API
    start_time = time.time()
    response = requests.post('http://127.0.0.1:5000/soap/{}'.format(self.element_response), headers={'content-type': 'application/soap+xml'}, data=data)
    end_time = time.time()

    #Zapisz zmienną w obiekcie
    self.soap_time = end_time - start_time

    #Pokazanie otrzymanych danych
    if self.print_response_time == True:
        print('Response time: {}'.format(self.soap_time))

    if self.print_response_body == True:
        xmldoc = minidom.parseString(response.content)
        print(xmldoc.toprettyxml(indent="    ", newl='\n', encoding='UTF-8').decode())
```

Wysłanie zapytania na serwer za pomocą interfejsu gRPC.

```
def grpc(self):
    print('---gRPC---')

    #Przygotowanie kanału
    channel = grpc.insecure_channel('localhost:50051')

    #Przygotowanie map
    class_map = {
        'movies': MoviesServiceStub,
        'directors': DirectorsServiceStub,
        'genres': GenresServiceStub
    }

    #Przygotowanie zmiennych
    stub = class_map[self.element_response](channel)
    method_name = 'get_{}'.format(self.element_response)

    #Zapytanie do gRPC API
    start_time = time.time()
    response = getattr(stub, method_name)(Empty())
    end_time = time.time()

    #Zapisz zmienną w obiekcie
    self.grpc_time = end_time - start_time

    #Pokazanie otrzymanych danych
    if self.print_response_time == True:
        print('Response time: {}'.format(self.grpc_time))

    if self.print_response_body == True:
        if self.element_response == 'movies':
            for movie in response:
                print(movie.id, movie.title, movie.description, movie.director_id, movie.genre_id)
        elif self.element_response == 'directors':
            for director in response:
                print(director.id, director.name, director.surname)
        elif self.element_response == 'genres':
            for genre in response:
                print(genre.id, genre.name)
```

8. Uruchomienie programu oraz przedstawienie wyników.

Funckja main programu:

```
if __name__ == "__main__":
    client = Client()
    client.rest()
    client.soap()
    client.grpc()
    client.matplotlib()
```

Kod przedstawiający funkcję wyświetlającą wykres z porównanymi czasami.

```
def matplotlib(self):
    try:
        #Przygotuj dane
        height = [self.rest_time, self.soap_time, self.grpc_time]
        bars = ('REST', 'SOAP', 'gRPC')
        y_pos = np.arange(len(height))

        #Ustaw rozmiar okna
        plt.figure(figsize=(10,5))

        #Ustaw bars
        plt.bar(y_pos, height, color = 'blue')

        #Ustaw nazwy dla interfe
        plt.xticks(y_pos, bars)

        #Ustaw label'e
        plt.xlabel('Interfejs', fontsize=12, color='#323232')
        plt.ylabel('Czas odpowiedzi', fontsize=12, color='#323232')
        plt.title('Interfejsy komunikacyjne REST SOAP gRPC', fontsize=16, color='#323232')

        #Pokaż wykres
        plt.show()
    except:
        print('No data')
```

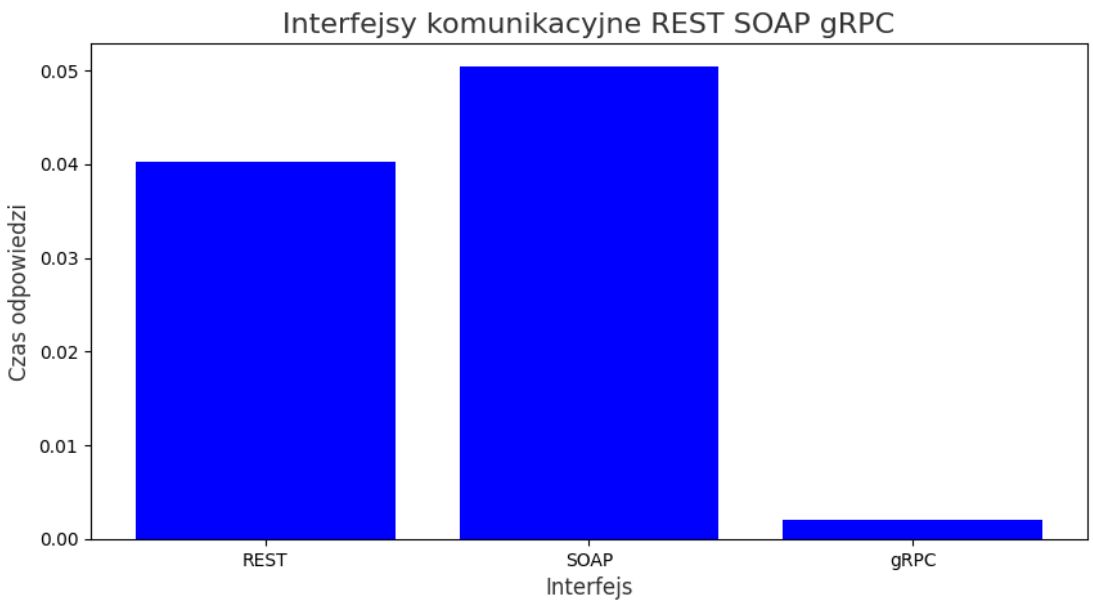
Uruchomienie servera.

```
C:\Windows\System32\cmd.exe - python  server.py
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Tomek\Desktop\Projekt_python\Interfejsy>python server.py
Uruchamianie serwera gRPC
Uruchamianie serwera Flask
 * Serving Flask app 'server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
Uruchamianie serwera gRPC
E0114 22:44:57.725000000  7832 src/core/ext/transport/chttp2/server/chttp2_server.cc:1057] UNKNOWN:No address added out
of total 1 resolved for '[::]:50051' {created_time:"2023-01-14T21:44:57.7252693+00:00", children:[UNKNOWN:Failed to add
port to server {created_time:"2023-01-14T21:44:57.7252441+00:00", children:[UNAVAILABLE:WSA Error {syscall:"bind", os_er
ror:"Unable to retrieve error string", grpc_status:14, wsa_error:10048, created_time:"2023-01-14T21:44:57.7250228+00:00"
}]}]}
Uruchamianie serwera Flask
 * Debugger is active!
 * Debugger PIN: 506-289-505
```

Test numer 1

```
C:\Windows\System32\cmd.exe - python  client.py
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Tomek\Desktop\Projekt_python\Interfejsy>python client.py
---REST---
Response time: 0.04787087440490723
---SOAP---
Response time: 0.05385589599609375
---gRPC---
Response time: 0.0019922256469726562
```

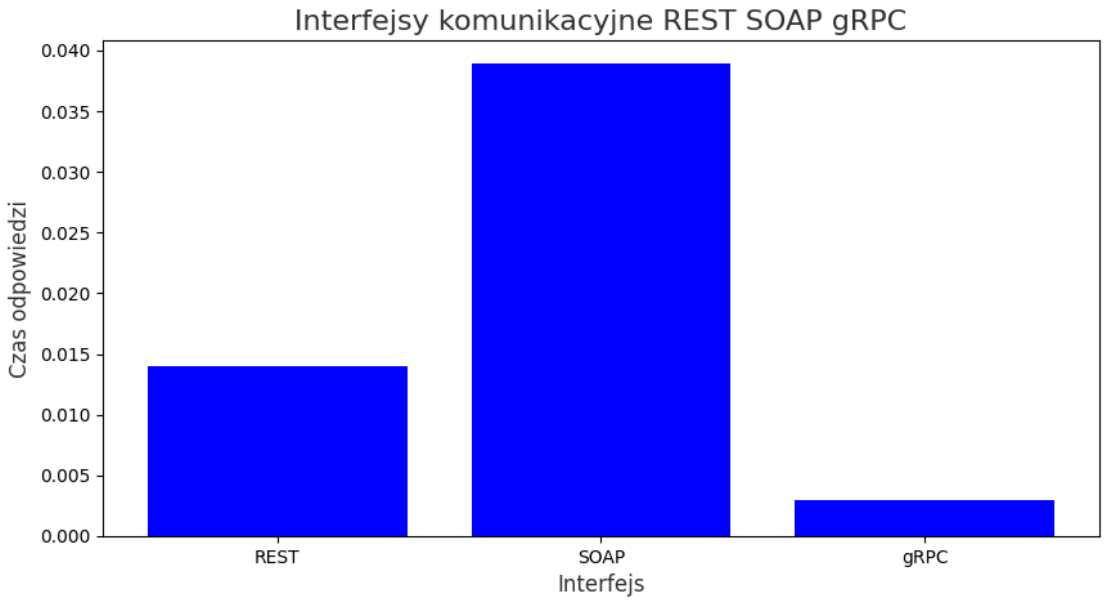
Wykres numer 1



Test numer 2

```
C:\Windows\System32\cmd.exe - python  client.py
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Tomek\Desktop\Projekt_python\Interfejsy>python client.py
---REST---
Response time: 0.013960599899291992
---SOAP---
Response time: 0.03889608383178711
---gRPC---
Response time: 0.002990245819091797
```

Wykres numer 2



9. Wnioski

Wniosek, który można wyciągnąć po porównaniu interfejsów REST, SOAP i gRPC, jest taki, że gRPC jest znacznie szybszy niż pozostałe dwa interfejsy. W pierwszym teście, gRPC jest o 95.61% szybszy od REST i o 96.25% szybszy od SOAP. W drugim teście, gRPC jest o 97.85% szybszy od REST i o 92.67% szybszy od SOAP. Z tego powodu, jeśli szybkość przesyłania danych jest krytyczna dla działania aplikacji, gRPC jest najlepszym wyborem. Należy jednak pamiętać, że każdy interfejs ma swoje własne zalety i wady, więc należy dokładnie przeanalizować potrzeby aplikacji przed wyborem odpowiedniego rozwiązania.