

Sprawozdanie z Laboratorium 4 – ARIUS

Piotr Gutowski 318472

1. Opis implementacji endpointów:

1. /teacher-list Pobranie listy nauczycieli:

Nie pobiera żadnych argumentów, zwraca listę nauczycieli w bazie, wraz z wymaganymi danymi: id_nauczyciela, imię, nazwisko, przedmioty

```
# 1. Lista nauczycieli
@app.route('/teacher-list', methods=['GET'])
def get_teacher_list():
    nauczyciele = Nauczyciel.query.all()
    response = [
        {
            'id_nauczyciela': nauczyciel.id_nauczyciela,
            'imie': nauczyciel.imie,
            'nazwisko': nauczyciel.nazwisko,
            'przedmioty': nauczyciel.prowadzone_przedmioty
        }
        for nauczyciel in nauczyciele
    ]
    return jsonify(response), 200
```

2. /teacher-details Szczegóły nauczyciela,

Przyjmuje id_nauczyciela i zwraca jego wszystkie dane (tak jak na zrzucie ekranu)

```
# 2. Szczegóły nauczyciela
@app.route('/teacher-details/<int:id_nauczyciela>', methods=['GET'])
def get_teacher_details(id_nauczyciela):
    nauczyciel = Nauczyciel.query.filter_by(id_nauczyciela=id_nauczyciela).first()
    if not nauczyciel:
        return jsonify({'error': 'Nie znaleziono nauczyciela'}), 404

    response = {
        'id_nauczyciela': nauczyciel.id_nauczyciela,
        'imie': nauczyciel.imie,
        'nazwisko': nauczyciel.nazwisko,
        'opis': nauczyciel.opis,
        'przedmioty': nauczyciel.prowadzone_przedmioty,
        'ocena': nauczyciel.ocena_nauczyciela,
        'numer_telefonu': nauczyciel.numer_telefonu,
        'stawka': nauczyciel.stawka,
        'waluta': nauczyciel.waluta,
        'email': nauczyciel.email
    }
    return jsonify(response), 200
```

3. /book-lesson Zarezerwowanie lekcji

Pobiera id_studenta, id_nauczyciela, oraz datę lekcji.

Sprawdza dostępność terminu i zwraca błąd jeśli termin jest niedostępny lub już zajęty

Zwraca informacje o powodzeniu zarezerwowania lekcji.

```
# 3. Zarezerwowanie lekcji
@app.route("/book-lesson", methods=["POST"])
def book_lesson():
    """Endpoint do rezerwowania lekcji.
    Sprawdza, czy wybrana data i godzina są dostępne."""
    data = request.get_json()

    # Pobieranie parametrów
    id_studenta = data.get("id_studenta")
    id_nauczyciela = data.get("id_nauczyciela")
    data_lekcji = data.get("data_lekcji")

    # Walidacja parametrów
    if not all([id_studenta, id_nauczyciela, data_lekcji]):
        return jsonify({"error": "Brak wymaganych parametrów"}), 400

    try:
        # Konwersja daty i godziny
        data_lekcji = datetime.strptime(data_lekcji, "%Y-%m-%d %H:%M")
    except ValueError:
        return jsonify({"error": "Nieprawidłowy format daty i godziny"}), 400

    # Sprawdzenie, czy termin jest już zajęty
    zajeta_lekcja = Lekcja.query.filter_by(
        id_nauczyciela=id_nauczyciela,
        data_lekcji=data_lekcji
    ).first()

    if zajeta_lekcja:
        return jsonify({"error": "Termin jest już zajęty"}), 409

    # Dodanie nowej lekcji
    nowa_lekcja = Lekcja(
        id_nauczyciela=id_nauczyciela,
        id_studenta=id_studenta,
        id_przedmiotu=1,
        data_lekcji=data_lekcji
    )
    db.session.add(nowa_lekcja)
    db.session.commit()

    return jsonify({"message": "Lekcja została zarezerwowana"}), 201
```

4. /add-teacher Dodanie nauczyciela

Pobiera jako dane wejściowe dane nowego nauczyciela, sprawdzana jest obecność wszystkich danych.

Podane id (kalendarza) przypisuje nauczyciela do wybranego grafiku.

Zwracana jest informacja o dodaniu nauczyciela wraz z jego id_nauczyciela lub błąd jeśli operacja nie powiodła się (np. Z powodu braku danych).

```
# 4. Dodawanie nauczyciela
@app.route('/add-teacher', methods=['POST'])
def add_teacher():
    data = request.get_json()

    required_fields = ['imie', 'nazwisko', 'prowadzone_przedmioty', 'opis', 'ocena_nauczyciela', 'numer_telefonu', 'stawka', 'waluta', 'email', 'id']
    for field in required_fields:
        if field not in data:
            return jsonify({'error': f"Brak wymaganego pola: {field}"}), 400

    # Pobranie istniejącego kalendarza
    kalendarz = KalendarzNauczyciela.query.get(data['id'])
    if not kalendarz:
        return jsonify({'error': 'Nie znaleziono kalendarza o podanym id'}), 404

    # Tworzenie nowego nauczyciela
    nowy_nauczyciel = Nauczyciel(
        imie=data['imie'],
        nazwisko=data['nazwisko'],
        prowadzone_przedmioty=data['prowadzone_przedmioty'],
        opis=data['opis'],
        ocena_nauczyciela=data['ocena_nauczyciela'],
        numer_telefonu=data['numer_telefonu'],
        stawka=data['stawka'],
        waluta=data['waluta'],
        email=data['email']
    )

    # Powiązanie nauczyciela z kalendarzem
    kalendarz.nauczyciel = nowy_nauczyciel

    # Zapis do bazy danych
    db.session.add(nowy_nauczyciel)
    db.session.commit()

    return jsonify({
        'message': 'Nauczyciel został dodany',
        'id_nauczyciela': nowy_nauczyciel.id_nauczyciela
    }), 201
```

5. /get-lessons Pobranie informacji o lekcjach danego studenta w danym przedziale czasowym

Pobiera id_studenta oraz ramy przedziału czasowego.

Zwraca wszystkie lekcje zarezerwowane dla danego studenta w danym okresie i podaje ich dane (w tym id, imię i nazwisko prowadzącego, nazwę przedmiotu oraz termin)

W przypadku podania nieprawidłowego id studenta, zwraca odpowiedni błąd.

```

#5. Pobranie informacji o lekcjach studenta w danym przedziale
@app.route("/get-lessons", methods=["GET"])
def get_lessons():
    # Pobranie parametrów z zapytania
    id_studenta = request.args.get("id_studenta", type=int)
    data_początkowa = request.args.get("data_początkowa")
    data_koncowa = request.args.get("data_koncowa")

    # Walidacja wymaganych parametrów
    if not id_studenta or not data_początkowa or not data_koncowa:
        return "", 400 # Zwraca pustą odpowiedź z kodem 400 (Bad Request)

    try:
        # Konwersja dat na obiekt datetime
        data_początkowa = datetime.strptime(data_początkowa, "%Y-%m-%d %H:%M")
        data_koncowa = datetime.strptime(data_koncowa, "%Y-%m-%d %H:%M")
    except ValueError:
        return "", 400 # Błąd parsowania daty, zwraca pustą odpowiedź

    # Sprawdzenie, czy student istnieje
    student = Student.query.get(id_studenta)
    if not student:
        return "", 404 # Brak studenta, zwraca pustą odpowiedź z kodem 404 (Not Found)

    # Pobieranie lekcji dla studenta w danym przedziale czasowym
    lekcje = (
        Lekcja.query
        .filter(Lekcja.id_studenta == id_studenta)
        .filter(Lekcja.data_lekcji >= data_początkowa)
        .filter(Lekcja.data_lekcji <= data_koncowa)
        .all()
    )

    # Jeśli brak wyników, zwracamy pustą odpowiedź z kodem 200
    if not lekcje:
        return "", 200

    # Konwersja wyników na JSON
    lekcje_json = [
        {
            "id_lekcji": lekcja.id_lekcji,
            "id_nauczyciela": lekcja.id_nauczyciela,
            "imie_nauczyciela": lekcja.nauczyciel.imie,
            "nazwisko_nauczyciela": lekcja.nauczyciel.nazwisko,
            "id_studenta": lekcja.id_studenta,
            "data_lekcji": lekcja.data_lekcji.strftime("%Y-%m-%d %H:%M"),
            "id_przedmiotu": lekcja.id_przedmiotu,
            "przedmiot": lekcja.przedmiot.nazwa_przedmiotu
        }
        for lekcja in lekcje
    ]

    return jsonify(lekcje_json), 200

```

2. Skrypty implementujące poszczególne żądania

W pierwszym przypadku nie ma możliwości przeprowadzić testu negatywnego, ponieważ funkcja nie przyjmuje danych wejściowych.

```
BASE_URL = "http://127.0.0.1:5000" # Adres lokalnego serwera Flask
HEADERS = {"Authorization": "Piotr Gutowski"}

def test_teacher_list():
    print("### TEST /teacher-list ###")
    # Funkcja nie przyjmuje danych wejściowych więc nie rozpatruję przypadku negatywnego
    response = requests.get(f"{BASE_URL}/teacher-list", headers=HEADERS)
    print("Wynik:", response.status_code, response.json())

def test_teacher_details():
    print("\n### TEST /teacher-details ###")
    # Przypadek pozytywny – poprawne id
    response = requests.get(f"{BASE_URL}/teacher-details/1", headers=HEADERS)
    print("Pozytywny wynik:", response.status_code, response.json())

    # Przypadek negatywny – nieistniejący student
    response = requests.get(f"{BASE_URL}/teacher-details/999", headers=HEADERS)
    print("Negatywny wynik:", response.status_code, response.json())

def test_book_lesson():
    print("\n### TEST /book-lesson ###")
    # Przypadek pozytywny
    payload = {
        "id": 1,
        "id_studenta": 1,
        "id_nauczyciela": 1,
        "data_lekcji": "2024-12-18 10:00",
    }
    response = requests.post(f"{BASE_URL}/book-lesson", headers=HEADERS, json=payload)
    print("Pozytywny wynik:", response.status_code, response.json())

    # Przypadek negatywny – zajęty termin
    response = requests.post(f"{BASE_URL}/book-lesson", headers=HEADERS, json=payload)
    print("Negatywny wynik:", response.status_code, response.json())
```

```

def test_add_teacher():
    print("\n### TEST /add-teacher ###")
    # Przypadek pozytywny
    payload = {
        "imie": "Adam",
        "nazwisko": "Nowakowski",
        "prowadzone_przedmioty": "matematyka, fizyka",
        "opis": "Nauczyciel z wieloletnim doświadczeniem.",
        "ocena_nauczyciela": 4.9,
        "numer_telefonu": "123456788",
        "stawka": 100,
        "waluta": "PLN",
        "email": "a.nowakowski@example.com",
        "id": 3 # Pole id oznacza id kalendarza
    }
    response = requests.post(f"{BASE_URL}/add-teacher", headers=HEADERS, json=payload)
    print("Pozytywny wynik:", response.status_code, response.json())

    # Przypadek negatywny – brak wymaganych danych
    payload.pop("imie")
    response = requests.post(f"{BASE_URL}/add-teacher", headers=HEADERS, json=payload)
    print("Negatywny wynik:", response.status_code, response.json())

```

```

def test_get_lessons():
    print("\n### TEST /get-lessons ###")

    # Przypadek pozytywny
    params = {
        "id_studenta": 1,
        "data_początkowa": "2024-12-10 08:00",
        "data_końcowa": "2024-12-18 18:00"
    }
    response = requests.get(f"{BASE_URL}/get-lessons", headers=HEADERS, params=params)
    print("Pozytywny wynik:", response.status_code, response.text)

    # Przypadek negatywny – nieistniejący student
    params["id_studenta"] = 999
    response = requests.get(f"{BASE_URL}/get-lessons", headers=HEADERS, params=params)
    print("Negatywny wynik:", response.status_code)
    if response.text:
        print("Treść błędu", response.text)
    else:
        print("Prawidłowa pusta odpowiedź.")

if __name__ == "__main__":
    # Testowanie endpointów
    test_teacher_list()
    test_teacher_details()
    test_book_lesson()
    test_add_teacher()
    test_get_lessons()

```

3. Wyniki skierowania żądań do endpoint'ów

Odpowiedź na skrypt testowy:

```
(venv) piotrgutowski@MacBook-Pro-Piotr lab4arius % python skrypt.py
### TEST /teacher-list ###
Wynik: 200 [{"id_nauczyciela": 1, 'imie': 'Jan', 'nazwisko': 'Kowalski', 'przedmioty': 'matematyka,fizyka'}, {'id_nauczyciela': 2, 'imie': 'Anna', 'nazwisko': 'Nowak', 'przedmioty': 'chemia,biologia'}, {'id_nauczyciela': 3, 'imie': 'Piotr', 'nazwisko': 'Zielinski', 'przedmioty': 'historia'}, {'id_nauczyciela': 4, 'imie': 'Katarzyna', 'nazwisko': 'Wisniewska', 'przedmioty': 'matematyka,biologia'}, {'id_nauczyciela': 5, 'imie': 'Tadeusz', 'nazwisko': 'Lewandowski', 'przedmioty': 'fizyka,chemia'}]

### TEST /teacher-details ###
Pozytywny wynik: 200 {'email': 'jan.kowalski@example.com', 'id_nauczyciela': 1, 'imie': 'Jan', 'nazwisko': 'Kowalski', 'numer_telefonu': '123456789', 'ocena': 4.8, 'opis': 'Specjalista w naukach ścisłych', 'przedmioty': 'matematyka,fizyka', 'stawka': 50, 'waluta': 'PLN'}
Negatywny wynik: 404 {'error': 'Nie znaleziono nauczyciela'}

### TEST /book-lesson ###
Pozytywny wynik: 201 {'message': 'Lekcja została zarezerwowana'}
Negatywny wynik: 409 {'error': 'Termin jest już zajęty'}

### TEST /add-teacher ###
Pozytywny wynik: 201 {'id_nauczyciela': 6, 'message': 'Nauczyciel został dodany'}
Negatywny wynik: 400 {'error': 'Brak wymaganego pola: imie'}
```

```
### TEST /get-lessons ###
Pozytywny wynik: 200 [
  {
    "data_lekcji": "2024-12-10 14:00",
    "id_lekcji": 5,
    "id_nauczyciela": 1,
    "id_przedmiotu": 1,
    "id_studenta": 1,
    "imie_nauczyciela": "Jan",
    "nazwisko_nauczyciela": "Kowalski",
    "przedmiotu": "matematyka"
  },
  {
    "data_lekcji": "2024-12-12 12:00",
    "id_lekcji": 10,
    "id_nauczyciela": 5,
    "id_przedmiotu": 2,
    "id_studenta": 1,
    "imie_nauczyciela": "Tadeusz",
    "nazwisko_nauczyciela": "Lewandowski",
    "przedmiotu": "fizyka"
  },
  {
    "data_lekcji": "2024-12-14 15:00",
    "id_lekcji": 13,
    "id_nauczyciela": 4,
    "id_przedmiotu": 5,
    "id_studenta": 1,
    "imie_nauczyciela": "Katarzyna",
    "nazwisko_nauczyciela": "Wisniewska",
    "przedmiotu": "biologia"
  },
  {
    "data_lekcji": "2024-12-18 10:00",
    "id_lekcji": 17,
    "id_nauczyciela": 1,
    "id_przedmiotu": 1,
    "id_studenta": 1,
    "imie_nauczyciela": "Jan",
    "nazwisko_nauczyciela": "Kowalski",
    "przedmiotu": "matematyka"
  }
]
```

Negatywny wynik: 404

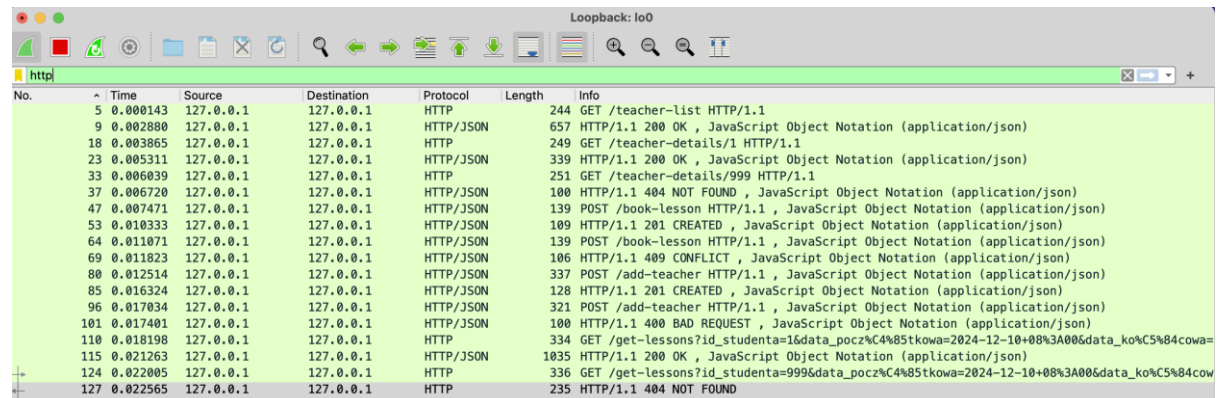
Prawidłowa pusta odpowiedź.

```
(venv) piotrgutowski@MacBook-Pro-Piotr lab4arius %
```


Widok z perspektywy serwera:

```
127.0.0.1 - - [18/Dec/2024 20:29:54] "GET /teacher-list HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2024 20:29:54] "GET /teacher-details/1 HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2024 20:29:54] "GET /teacher-details/999 HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2024 20:29:54] "POST /book-lesson HTTP/1.1" 201 -
127.0.0.1 - - [18/Dec/2024 20:29:54] "POST /book-lesson HTTP/1.1" 409 -
/Users/piotrgutowski/Desktop/lab4arius/lab4server.py:274: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    kalendarz = KalendarzNauczyciela.query.get(data['id'])
127.0.0.1 - - [18/Dec/2024 20:29:54] "POST /add-teacher HTTP/1.1" 201 -
127.0.0.1 - - [18/Dec/2024 20:29:54] "POST /add-teacher HTTP/1.1" 400 -
/Users/piotrgutowski/Desktop/lab4arius/lab4server.py:324: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    student = Student.query.get(id_studenta)
127.0.0.1 - - [18/Dec/2024 20:29:54] "GET /get-lessons?id_studenta=1&data_poczatkowa=2024-12-10+08:00&data_koncowa=2024-12-18+18:00 HTTP/1.1" 200 -
127.0.0.1 - - [18/Dec/2024 20:29:54] "GET /get-lessons?id_studenta=999&data_poczatkowa=2024-12-10+08:00&data_koncowa=2024-12-18+18:00 HTTP/1.1" 404 -
■
```

Dane przechwycone przez program Wireshark (na interfejsie loopback, po ustawieniu filtra http w celu odfiltrowania pakietów innych warstw np.tcp):



No.	Time	Source	Destination	Protocol	Length	Info
5	0.000143	127.0.0.1	127.0.0.1	HTTP	244	GET /teacher-list HTTP/1.1
9	0.002880	127.0.0.1	127.0.0.1	HTTP/JSON	657	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)
18	0.003865	127.0.0.1	127.0.0.1	HTTP	249	GET /teacher-details/1 HTTP/1.1
23	0.005311	127.0.0.1	127.0.0.1	HTTP/JSON	339	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)
33	0.006039	127.0.0.1	127.0.0.1	HTTP	251	GET /teacher-details/999 HTTP/1.1
37	0.006720	127.0.0.1	127.0.0.1	HTTP/JSON	100	HTTP/1.1 404 NOT FOUND, JavaScript Object Notation (application/json)
47	0.007471	127.0.0.1	127.0.0.1	HTTP/JSON	139	POST /book-lesson HTTP/1.1, JavaScript Object Notation (application/json)
53	0.010333	127.0.0.1	127.0.0.1	HTTP/JSON	109	HTTP/1.1 201 CREATED, JavaScript Object Notation (application/json)
64	0.011071	127.0.0.1	127.0.0.1	HTTP/JSON	139	POST /book-lesson HTTP/1.1, JavaScript Object Notation (application/json)
69	0.011823	127.0.0.1	127.0.0.1	HTTP/JSON	106	HTTP/1.1 409 CONFLICT, JavaScript Object Notation (application/json)
80	0.012514	127.0.0.1	127.0.0.1	HTTP/JSON	337	POST /add-teacher HTTP/1.1, JavaScript Object Notation (application/json)
85	0.016324	127.0.0.1	127.0.0.1	HTTP/JSON	128	HTTP/1.1 201 CREATED, JavaScript Object Notation (application/json)
96	0.017034	127.0.0.1	127.0.0.1	HTTP/JSON	321	POST /add-teacher HTTP/1.1, JavaScript Object Notation (application/json)
101	0.017401	127.0.0.1	127.0.0.1	HTTP/JSON	100	HTTP/1.1 400 BAD REQUEST, JavaScript Object Notation (application/json)
110	0.018198	127.0.0.1	127.0.0.1	HTTP	334	GET /get-lessons?id_studenta=1&data_pocz%C4%85tkowa=2024-12-10+08%3A00&data_ko%C5%B4cowa=
115	0.021263	127.0.0.1	127.0.0.1	HTTP/JSON	1035	HTTP/1.1 200 OK, JavaScript Object Notation (application/json)
124	0.022005	127.0.0.1	127.0.0.1	HTTP	336	GET /get-lessons?id_studenta=999&data_pocz%C4%85tkowa=2024-12-10+08%3A00&data_ko%C5%B4cow
127	0.022565	127.0.0.1	127.0.0.1	HTTP	235	HTTP/1.1 404 NOT FOUND

Pełny zapis bez filtra znajduje się w załączniku.