

Anggota Kelompok : **Muhammad Rheza Pramuditha Dewangga (123210150)**

Naufal Hafizh Akbar (123210070)

Brilliant Cahya Dewa (123210065)

LAPORAN TUGAS API (FAST API)

Penjelasan struktur dan fungsi masing-masing endpoint

1. Endpoint /consultations (GET):

- **Struktur:** `@app.get("/consultations",response_model=List[DoctorConsultation])`
- **Fungsi:** Endpoint ini bertujuan untuk menyediakan akses ke daftar lengkap jadwal konsultasi yang tersedia dalam basis data.
- **Penjelasan:** Saat permintaan GET diterima ke `/consultations`, server akan mengembalikan daftar seluruh buku yang tersedia dalam bentuk JSON. Data jadwal konsultasi tersebut akan disesuaikan dengan struktur model **Consultation**.

2. Endpoint /consultations/{consultation_id} (GET):

- **Struktur:** `@app.get("/consultations/{consultation_id}", response_model = DoctorConsultation)`
- **Fungsi:** Endpoint ini dimaksudkan untuk memberikan detail lengkap dari sebuah jadwal konsultasi berdasarkan ID konsultasi yang diberikan.
- **Penjelasan:** Saat permintaan GET diterima ke `/consultations/{consultation_id}`, server akan mencari jadwal konsultasi dengan ID yang sesuai dalam basis data. Jika jadwal konsultasi ditemukan, detailnya akan dikembalikan dalam bentuk JSON sesuai dengan struktur model **Consultation**. Jika jadwal konsultasi tidak ditemukan, akan dibangkitkan `HTTPException` dengan status code 404.

3. Endpoint /consultations (POST):

- **Struktur:** `@app.post("/consultations",response_model=DoctorConsultation)`
- **Fungsi:** Endpoint ini bertujuan untuk memperbarui daftar jadwal konsultasi dengan menambahkan entri baru.
- **Penjelasan:** Saat permintaan POST diterima ke `/consultations` dengan data jadwal konsultasi baru dalam format JSON, server akan memeriksa apakah jadwal konsultasi dengan ID yang sama sudah ada dalam basis data. Jika tidak, jadwal konsultasi baru akan ditambahkan ke basis data dan detailnya akan dikembalikan dalam bentuk JSON sesuai dengan struktur model **Consultation**. Jika jadwal konsultasi dengan ID yang sama sudah ada, akan dibangkitkan `HTTPException` dengan status code 400.

4. Endpoint /consultations/{consultation_id} (PUT):

- **Struktur:** `@app.put("/consultations/{consultation_id}",response_model=DoctorConsultation)`

- **Fungsi:** Endpoint ini digunakan untuk memperbarui detail dari sebuah jadwal konsultasi yang sudah ada berdasarkan ID konsultasi yang diberikan.
- **Penjelasan:** Saat permintaan PUT diterima ke `/consultations/{consultation_id}` dengan data jadwal konsultasi yang diperbarui dalam format JSON, server akan mencari jadwal konsultasi dengan ID yang sesuai dalam basis data. Jika jadwal konsultasi ditemukan, detailnya akan diperbarui dengan data baru yang diterima dan detail konsultasi yang diperbarui akan dikembalikan dalam bentuk JSON sesuai dengan struktur model `Consultation`. Jika jadwal konsultasi tidak ditemukan, akan dibangkitkan `HTTPException` dengan status code 404.

5. Endpoint `/consultations/{consultation_id}` (DELETE):

- **Struktur:** `@app.delete("/consultations/{consultation_id}", response_model=dict)`
- **Fungsi:** Endpoint ini bertujuan untuk menghapus sebuah jadwal konsultasi berdasarkan ID konsultasi yang diberikan.
- **Penjelasan:** Saat permintaan DELETE diterima ke `/consultations/{consultation_id}`, server akan mencari jadwal konsultasi dengan ID yang sesuai dalam basis data. Jika jadwal konsultasi ditemukan, jadwal konsultasi tersebut akan dihapus dari basis data dan pesan konfirmasi akan dikembalikan dalam bentuk JSON. Jika jadwal konsultasi tidak ditemukan, akan dibangkitkan `HTTPException` dengan status code 404.

Kode lengkap dengan comment/dokumentasi:

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List, Optional

# Inisialisasi aplikasi FastAPI
app = FastAPI()

# Model data untuk jadwal konsultasi dokter
class DoctorConsultation(BaseModel):

    id: int
    doctor_name: str
    patient_name: str
    appointment_time: str
    appointment_date: str
    description: Optional[str] = None

# In-memory database untuk menyimpan daftar jadwal konsultasi dokter
doctor_consultation_db = []

# Endpoint untuk mendapatkan daftar seluruh jadwal konsultasi dokter
@app.get("/consultations", response_model=List[DoctorConsultation])
async def get_consultations():
    """
    Get all doctor consultation schedules available in the database.
    """
```

```

    return doctor_consultation_db

# Endpoint untuk mendapatkan detail jadwal konsultasi dokter berdasarkan ID
@app.get("/consultations/{consultation_id}", response_model=DoctorConsultation)
async def get_consultation(consultation_id: int):
    """
    Get details of a doctor consultation schedule based on its ID.
    """
    for consultation in doctor_consultation_db:
        if consultation.id == consultation_id:
            return consultation

    # Jika jadwal konsultasi dokter tidak ditemukan, raise HTTPException dengan status 404
    raise HTTPException(status_code=404, detail="Consultation schedule not found")

# Endpoint untuk menambahkan jadwal konsultasi dokter baru ke dalam database
@app.post("/consultations", response_model=DoctorConsultation)
async def create_consultation(consultation: DoctorConsultation):
    """
    Create a new doctor consultation schedule entry in the database.
    """
    # Periksa apakah jadwal konsultasi dokter dengan ID yang sama sudah ada di database
    for existing_consultation in doctor_consultation_db:
        if existing_consultation.id == consultation.id:
            # Jika jadwal konsultasi dokter dengan ID yang sama sudah ada, raise HTTPException dengan status 400
            raise HTTPException(status_code=400, detail="Consultation schedule with this ID already exists")

    # Tambahkan jadwal konsultasi dokter baru ke dalam database
    doctor_consultation_db.append(consultation)
    return consultation

# Endpoint untuk memperbarui detail jadwal konsultasi dokter berdasarkan ID
@app.put("/consultations/{consultation_id}", response_model=DoctorConsultation)
async def update_consultation(consultation_id: int, updated_consultation: DoctorConsultation):
    """
    Update details of a doctor consultation schedule based on its ID.
    """
    for index, consultation in enumerate(doctor_consultation_db):
        if consultation.id == consultation_id:
            # Jika jadwal konsultasi dokter dengan ID yang sesuai ditemukan, perbarui detailnya
            doctor_consultation_db[index] = updated_consultation
            return updated_consultation

    # Jika jadwal konsultasi dokter tidak ditemukan, raise HTTPException dengan status 404

```

```

        raise HTTPException(status_code=404, detail="Consultation schedule not found")

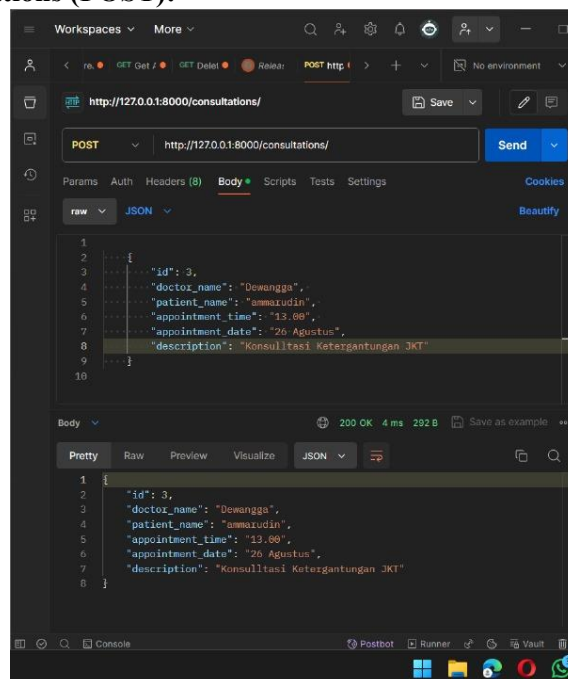
# Endpoint untuk menghapus jadwal konsultasi dokter berdasarkan ID
@app.delete("/consultations/{consultation_id}", response_model=dict)
async def delete_consultation(consultation_id: int):
    """
    Delete a doctor consultation schedule from the database based on its ID.
    """
    for index, consultation in enumerate(doctor_consultation_db):
        if consultation.id == consultation_id:
            # Jika jadwal konsultasi dokter dengan ID yang sesuai ditemukan, hapus
            # jadwal tersebut
            del doctor_consultation_db[index]
            return {"message": "Consultation schedule deleted"}
        # Jika jadwal konsultasi dokter tidak ditemukan, raise HTTPException dengan
        # status 404
        raise HTTPException(status_code=404, detail="Consultation schedule not found")

# Jalankan aplikasi jika file ini dijalankan secara langsung
if __name__ == '__main__':
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)

```

RESPONSE JSON: Kami memanfaatkan software Postman untuk menguji API-nya.

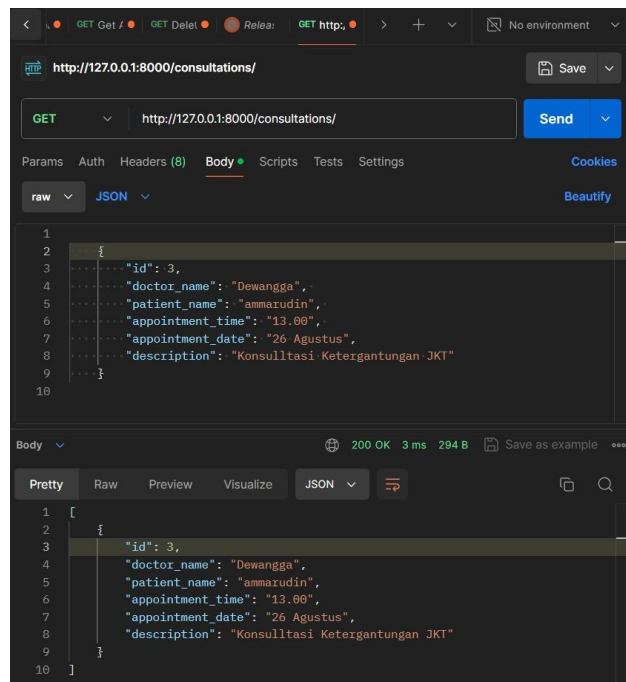
1. Endpoint /consultations (POST):



- Response JSON saat mengirim 2 data konsultasi ke AP

2. Endpoint /consultations (GET)

- Response JSON saat meminta seluruh data jadwal konsultasi dari API menggunakan GET dengan IP <http://127.0.0.1:8000/consultations/>



The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8000/consultations/`. The response is a JSON array containing one consultation object. The status is 200 OK, and the response time is 3 ms.

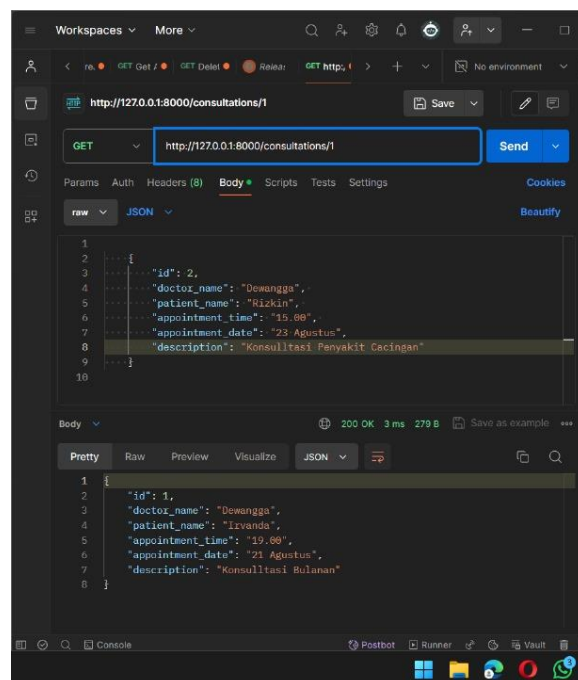
```
1 {
2   "id": 3,
3   "doctor_name": "Dewangga",
4   "patient_name": "ammarudin",
5   "appointment_time": "13.00",
6   "appointment_date": "26 Agustus",
7   "description": "Konsultasi Ketergantungan JKT"
8 }
9
10
```

Body: 200 OK 3 ms 294 B

```
1 [
2   {
3     "id": 3,
4     "doctor_name": "Dewangga",
5     "patient_name": "ammarudin",
6     "appointment_time": "13.00",
7     "appointment_date": "26 Agustus",
8     "description": "Konsultasi Ketergantungan JKT"
9   }
10 ]
```

3. Endpoint /consultations/{consultation_id} (GET):

- Response JSON saat meminta data jadwal konsultasi dengan ID = 2 dari API menggunakan URL <http://127.0.0.1:8000/consultations/2>



The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8000/consultations/1`. The response is a JSON object representing a consultation with ID 1. The status is 200 OK, and the response time is 3 ms.

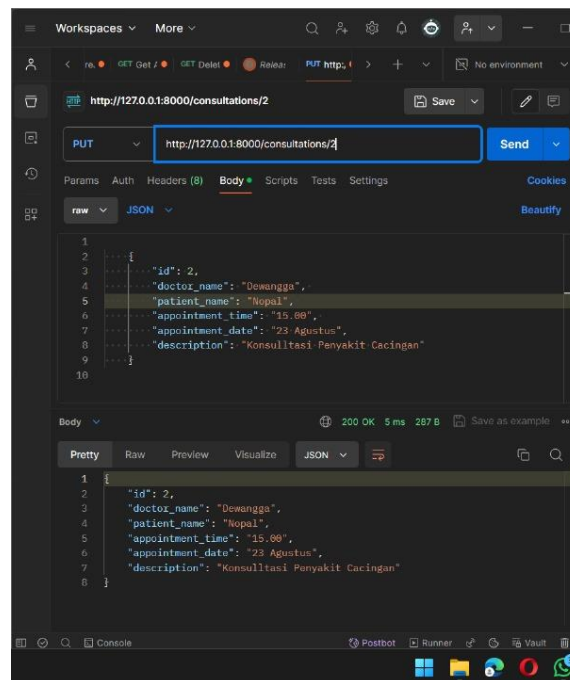
```
1 {
2   "id": 2,
3   "doctor_name": "Dewangga",
4   "patient_name": "Rizkin",
5   "appointment_time": "15.00",
6   "appointment_date": "23 Agustus",
7   "description": "Konsultasi Penyakit Cacingan"
8 }
9
10
```

Body: 200 OK 3 ms 279 B

```
1 {
2   "id": 1,
3   "doctor_name": "Dewangga",
4   "patient_name": "Irvanda",
5   "appointment_time": "19.00",
6   "appointment_date": "21 Agustus",
7   "description": "Konsultasi Bulanan"
8 }
```

4. Endpoint /consultations/{consultation_id} (PUT):

- Response JSON saat mengupdate buku dengan ID = 2, dan mengubah data sesuai yang diinginkan



5. Endpoint /consultations/{consultation_id} (DELETE):

- Response JSON saat menghapus jadwal konsultasi berdasarkan ID nya

