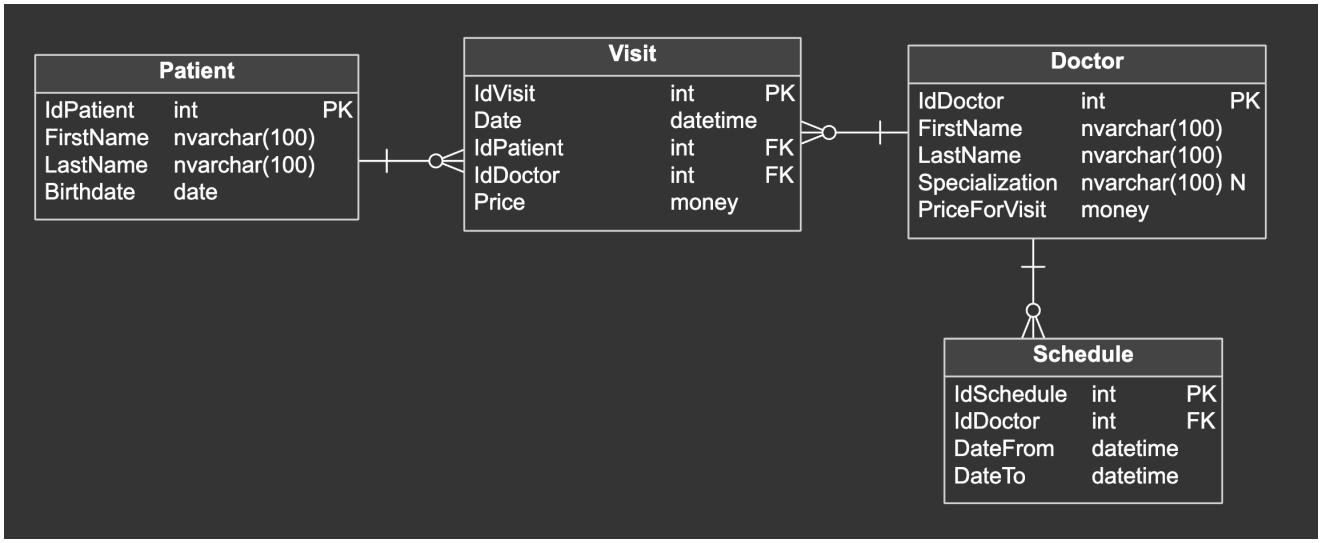


# 1. Pobranie danych pacjenta wraz z wizytami

1. Przygotowujemy końcówkę, która jako zwróci nam dane pacjenta wraz z informacjami na temat przypisanych do niego wizyt.

```
{
     "firstName": "Jan",
     "lastName": "Kowalski",
     "birthdate": "1980-01-01", //zwracany format daty może być inny, można użyć typu
danych datetime
     "totalAmountMoneySpent": "1200 zł",
     "numberOfVisit": 10,
     "visits": [
             {
                     "IdVisit": 1,
                     "Doctor": "John Doe",
                     "Date": "2024-03-12 12:30",
                     "Price": "100 zł"
             },
             ...
     ]
}
```

3. Kolumna totalAmountMoneySpent zwraca sumę wydanych przez klienta pieniędzy.
4. Kolumna numberOfVisit zwraca liczbę wizyt przypisanych do pacjenta.

# 2. Wstawienie nowej wizyty

1. Implementujemy końcówką, która pozwala na wstawienie nowej wizyty.
2. Końcówka powinna przyjmować następujące parametry: IdPatient, IdDoctor i Date.
3. Powinniśmy upewnić się, że:
    1. Pacjent istnieje.
    2. Doktor istnieje.
    3. Przed przypisaniem upewniamy się, że klient nie ma już umówionej żadnej innej wizyty. Sprawdzamy jest w bazie mamy jakąkolwiek inną wizytę z Date>Now.
    4. Sprawdzamy czy wybrany lekarz pracuje danego dnia w odpowiednich godzinach na podstawie tabeli Schedule. Jeśli doktor nie jest dostępny, zwracamy odpowiedni kod błędu.
    5. Sprawdzamy czy doktor nie ma tego dnia więcej niż 5 wizyt. Jeśli tak, to zwracamy odpowiedni błąd.
    6. Przypisujemy cenę do wizyty na podstawie ceny za wizytę u danego doktora. Jeśli pacjent odbył już więcej niż 10 wizyt - przyznajemy mu rabat 10%. Rabat uwzględniamy w wartości wstawianej do kolumny price w tabelce Visit.
    7. Wstawiamy rekord do tabeli Visit.
    8. Zwracamy wygenerowaną wartość Id.
    9. Pamiętamy o zwracaniu odpowiednich kodów błędu.

**Dodatkowe informacje:**

- Korzystamy z Entity Framework.
- Możesz skorzystać zarówno z podejścia Code First lub Database First. Podejście Database First jest punktowane nieco niżej - maksymalnie na ocenę 4.5.
- Pamiętaj o zasadach związanych z projektowaniem aplikacji typu REST.
- Pamiętaj o dobrych praktykach związanych z architekturę aplikacji.
- Upewnij się, że aplikacja się kompiluje.
- Pamiętaj o zwracaniu odpowiednich kodów HTTP i obsłudze błędów.
- W pierwszej kolejności ważne jest działanie aplikacji, w drugiej kolejności jakość kodu (oddzielenie logiki biznesowej, prezentacji i infrastruktury).