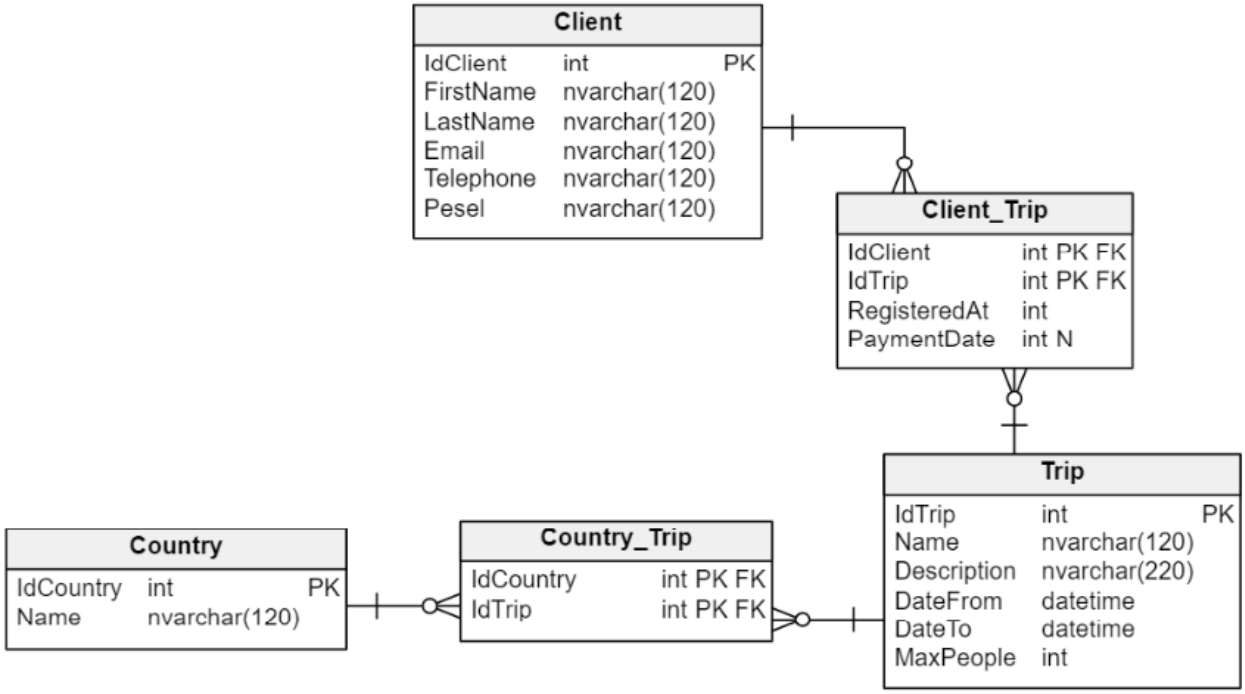
W tym zadaniu wykorzystujemy podejście Entity Framework Core - Database First do implementacji wymagań.



Tworzymy nową aplikację wykorzystującą powyższą bazę danych.

- Stwórz nową aplikację REST
- Wygeneruj niezbędna klasy z pomocą podejścia EF database first
- Przygotuj końcówki realizujące następujące funkcje

1. Endpoint dla żądania HTTP GET /api/trips
    1. Endpoint powinien zwrócić listę wycieczek posortowanych malejącą po dacie rozpoczęcia wycieczki.
    2. Dodaj opcjonalną możliwość stronicowania wyniku z pomocą query stringa i parametrów page i pageSize. Możemy założyć, że domyślny pageSize to 10.
    3. Przykład odpowiedzi:

```
{
    "pageNum": 1,
```

```
    "pageSize": 10,
    "allPages": 20,
    "trips": [
        {
            "Name": "ABC",
            "Description": "Lorem ipsum...",
            "DateFrom": "",
            "DateTo": "",
            "MaxPeople": 20,
            "Countries": [
                {
                    "Name": "Poland"
                },
                {
                    "Name": "Germany"
                }
            ],
            "Clients": [
                {
                    "FirstName": "John",
                    "LastName": "Smith"
                },
                {
                    "FirstName": "Jake",
                    "LastName": "Doe"
                }
            ]
        }
    ]
}
```

2. Przygotuj endpoint pozwalający na usunięcie klienta
    1. Żądania powinny być przyjmowane na adres HTTP DELETE /api/clients/{idClient}
    2. Serwer powinien najpierw sprawdzić czy klient nie ma przypisanych żadnych wycieczek. Jeśli klient ma co

najmniej jedną wycieczkę - zwracamy odpowiedni kod błędy z czytelną wiadomością

3. Przygotuj endpoint, który pozwala na przypisanie klienta do wycieczki
    1. Żądanie powinny być przyjmowane na adres HTTP POST /api/trips/{idTrip}/clients
    2. Serwer powinien w trakcie ralizacji żądania:
        1. Sprawdzić czy klient o takim numerze PESEL już istnieje - jeśli tak zwracamy błąd
        2. Czy klient o takim numerze PESEL jest już zapisany na daną wycieczkę - jeśli tak, zwracamy błąd
        3. Sprawdzamy czy dana wycieczka istnieje i czy DateFrom jest w przyszłości. Nie możemy zapisać się na wycieczkę, która już się odbyła.
        4. PaymentDate może mieć wartość NULL dla tych klientów, którzy jeszcze nie zapłacili za wycieczkę. Ponadto RegisteredAt w tabeli Client_Trip powinna być zgodna z czasem przyjęcia żądania przez serwer.
    3. Przykład parametrów wysłanych w żądaniu (data może być wysłana w innym formacie):

```
{
    "FirstName": "John",
    "LastName": "Doe",
    "Email": "doe@wp.pl",
    "Telephone": "543-323-542",
    "Pesel": "91040294554",
    "IdTrip": 10,
    "TripName": "Rome",
    "PaymentDate": "4/20/2021"
}
```

- Pamiętać o poprawnych nazwach, separacji między UI/infrastrukturę/logiką. Pamiętaj o wykorzystywaniu odpowiednich modeli.