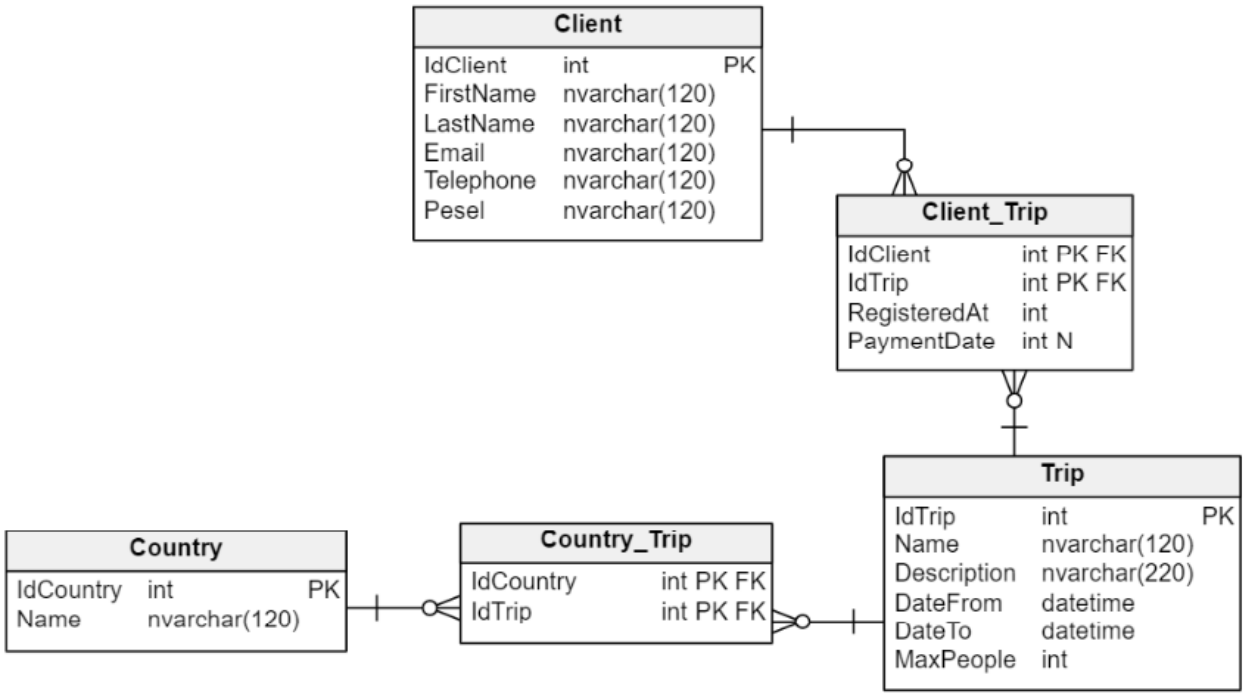
In this task, we use the Entity Framework Core - Database First approach to implement the requirements.



We are creating a new application using the above database.

- Create a new REST application
- Generate the necessary classes using the EF Database First approach
- Prepare endpoints to implement the following functionalities

1. Endpoint for HTTP GET request /api/trips
    1. The endpoint should return a list of trips sorted in descending order by the trip start date.
    2. Add an optional ability to paginate the results using query string and the parameters page and pageSize. We can assume the default pageSize is 10.
    3. Example response:

```
{
    "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. Prepare an endpoint to delete a client
    1. Requests should be accepted at the HTTP DELETE address /api/clients/{idClient}
    2. The server should first check if the client has any assigned trips. If the client has at least one trip, we return an

appropriate error code with a readable message.

3. Prepare an endpoint that allows assigning a client to a trip
    1. Requests should be accepted at the HTTP POST address /api/trips/{idTrip}/clients
    2. The server should, during the request processing:
        1. Check if a client with the given PESEL number already exists - if so, return an error.
        2. Check if a client with the given PESEL number is already registered for the given trip - if so, return an error.
        3. Check if the given trip exists and if DateFrom is in the future. We cannot register for a trip that has already occurred.
        4. PaymentDate can be NULL for clients who have not yet paid for the trip. Additionally, RegisteredAt in the Client_Trip table should match the time the request was received by the server.
    3. Example parameters sent in the request (the date may be sent in a different format):

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

- Remember to use correct names, and separation between UI/infrastructure/logic. Remember to use appropriate models.