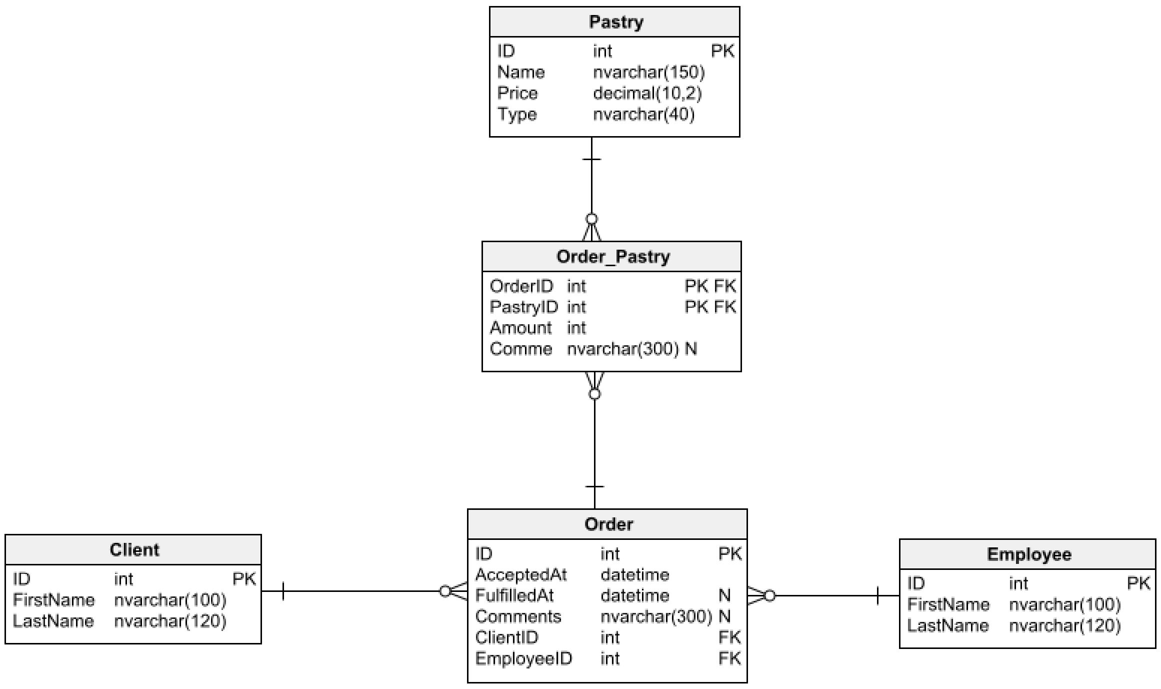
Example test 2

The tasks should be performed in a Web API project using the ORM framework - **EntityFrameworkCore**.

Database

The diagram of the database used in the following tasks is presented below.

Make sure to add sample data to the appropriate tables.



Endpoints

1. Design an endpoint that will return a list of orders placed by a customer with the given name. If the customer's name is not submitted, return a list containing the orders of all customers. The resulting list does not need to include information about the employee who was responsible for taking the order or the customer's personal information. Instead, it must include what was included in a given order (what confectionery products were selected for it). Remember to return correct error codes.

   The endpoint should respond to a query to the `api/orders` address e.g.
   `HTTP GET http://localhost:5000/api/orders`
   `HTTP GET http://localhost:5000/api/orders?clientLastName=Kowalski`

   Examples of the data that will be returned:

```json
[
  {
    "id": 1,
    "acceptedAt": "2024-05-28T00:00:00",
    "fulfilledAt": "2024-05-29T00:00:00",
    "comments": "Lorem ipsum ...",
    "pastries": [
      {
        "name": "Drożdzówka",
        "price": 3.3,
        "amount": 3
      },
      {
        "name": "Jagodzianka",
        "price": 7.2,
        "amount": 4
      }
    ]
  },
  {
    "id": 2,
    "acceptedAt": "2024-05-31T00:00:00",
    "fulfilledAt": "2024-06-01T00:00:00",
    "comments": "Lorem ipsum ...",
    "pastries": [
      {
        "name": "Drożdzówka",
        "price": 3.3,
        "amount": 12
      },
      {
        "name": "Babka cytrynowa",
        "price": 21.23,
        "amount": 2
      }
    ]
  }
]
```

2. Design an endpoint to add a new order. It should accept the basic information needed to add an order and the collection of confectionery products that should be part of it. If a product that is not in the database is specified, the processing of the request should be aborted and the corresponding error status should be returned. The ID of the customer making the assumption should be given as part of the URL segment. The entire task should be completed using a single transaction.

   The endpoint should respond to a query to the `api/clients` address e.g. `HTTP POST http://localhost:5000/api/clients/1/orders`
   Example of the data that is sent with the request:

```
{
  "employeeId": 1,
  "acceptedAt": "2024-05-26",
  "comments": "Lorem ipsum dolor sit amet",
  "pastries": [
    {
      "name": "Jagodzianka",
      "amount": 5,
      "comments": null
    }
  ]
}
```

Scoring

Each endpoint is calculated at **10 points**. That is, **20 points** can be earned for the total.

Points can be deducted for:

- Incorrect or illegible variable names: **-2 pts**
- Incorrect HTTP codes: **-4 pts**
- Failure to separate database communication into a separate file/service/repository: **-10 pts**
- Missing dependency injection: **-8 pts**

Remarks

- The test should be solved using the **CodeFirst** approach
- Solving the test without database tables - **0 pts**
- Solving the test without sample data **-50% points**
- Handing in an uncompiled test - **0 pts**
- Detection of plagiarism - **2 from the subject** without the possibility of correcting it