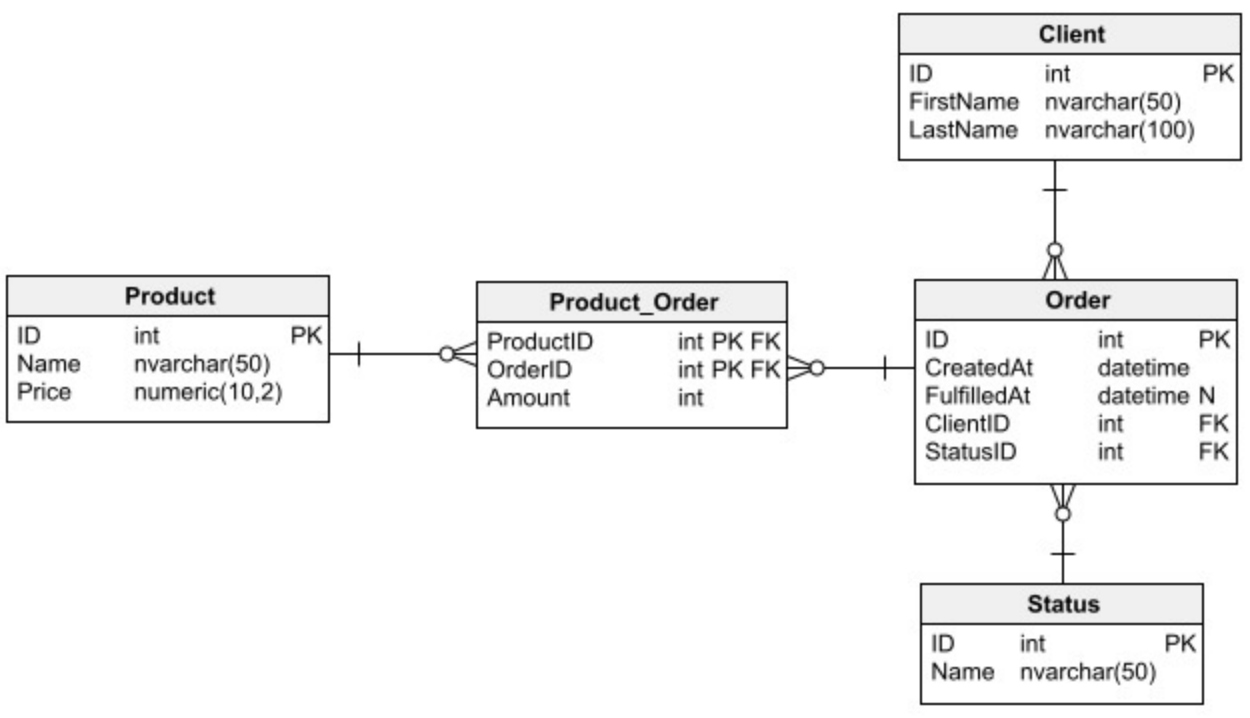
Kolokwium 2

Kolokwium powinno zostać zrealizowane na bazie lokalnej lub uczelnianej przy pomocy Entity Framework w wersji Core.

Baza danych

Poniżej zaprezentowany jest diagram bazy danych wykorzystywanej w poniższych zadaniach.



Należy zadbać o dodanie **przykładowych danych** do odpowiednich tabel.

Końcówki

1. Zaprojektuj końcówkę, która będzie zwracać informacje o zamówieniach danego klienta. Każde zwrócone zamówienie powinno zawierać swoje ID, informację o czasie utworzenia, status zamówienia oraz listę zawartych w nim produktów.
   - Jeśli podany klient nie istnieje powinien zostać zwrócony odpowiedni błąd HTTP.

   Końcówka powinna reagować na zapytanie pod adres `api/clients` np.

   `HTTP GET http://localhost:5000/api/clients/1/orders`

   Przykładowe dane jakie zostaną zwrócone:

```
[
  {
    "orderID": 1,
    "clientsLastName": "Kowalski",
    "createdAt": "2023-05-03T09:37:52.0951584",
    "fulfilledAt": "2023-05-03T09:37:52.0951587",
    "status": "Created",
    "products": [
      {
        "name": "Pączek",
        "price": 10.2,
        "amount": 3
      }
    ]
  },
  {
    "orderID": 2,
    "clientsLastName": "Kowalski",
    "createdAt": "2023-05-03T09:37:52.0951594",
    "fulfilledAt": null,
    "status": "Finished",
    "products": [
      {
        "name": "Gniazko Poznańskie",
        "price": 11.2,
        "amount": 5
      }
    ]
  }
]
```

2. Zaprojektuj końcówkę, która pozwoli dodać zamówienie dla danego klienta wraz z przesłanymi produktami. Przy dodawaniu nowego zamówienia do bazy, ustaw wartość pola "StatusID" numerem statusu odpowiadającego statusowi "Utworzone".
   - W przypadku podania niepełnych lub niepoprawnych danych powinien zostać zwrócony odpowiedni kod HTTP.
   - Po pomyślnym dodaniu zamówienia zwróć odpowiedni kod HTTP.
   - Jeśli podany klient nie istnieje powinien zostać zwrócony odpowiedni błąd HTTP.
   - Jeżeli status `"Utworzone"` nie istnieje powinien zostać zwrócony odpowiedni błąd HTTP.
   - Pole `fulfilledAt` jest opcjonalne
   - Pamiętaj o wykorzystaniu **transakcji**!

   Końcówka powinna reagować na zapytanie pod adres `api/clients/` np.

   `HTTP POST http://localhost:5000/api/clients/1/orders`

   Przykładowe dane jakie są przesłane wraz z żądaniem:

```
{
  "createdAt": "2023-05-19T10:00:01.001",
  "fulfilledAt": "2023-05-24T10:00:01.001",
  "products": [
    {
      "id": 1,
      "amount": 3
    },
    {
      "id": 2,
      "amount": 1
    }
  ]
}
```

Punktacja

Kontekst bazy danych jest oceniany na **8 punktów**.
Każda końcówka jest wyliczana na **6 punktów**.
Czyli za całość można zdobyć **20 punktów**.
Punkty mogą być odjęte za:

- Niepoprawne lub nieczytelne nazwy zmiennych: **max -3 pkt**
- Brak gitignore: **-1 pkt**
- Wypushowanie zbędnych plików do repozytorium: **-1 pkt**
- Niepoprawne kody HTTP: **max -4 pkt**
- Brak wydzielenia komunikacji bazy danych do oddzielnego pliku/serwisu/repozytorium: **max 50% punktów za końcówkę**
- Brak wstrzykiwania zależności: **-8 pkt**

Uwagi

- Kod należy spushować na Github Classrooma. Rozwiązanie powinno zostać umieszczone na platformie **max. 5 min** po zakończeniu zajęć. Projekty umieszczone po czasie nie będą brane pod uwagę podczas sprawdzania (!)
- Kolokwium powinno być rozwiązane przy użyciu podejścia **Code-First**
- Rozwiązanie kolokwium bez tabel w bazie danych - **0 pkt**
- Rozwiązanie kolokwium bez przykładowych danych - **-50% punktów**
- Oddanie niekompilującego się kolokwium - **0 pkt**
- Wykrycie plagiatu - **2 z przedmiotu** bez możliwości poprawy