



W niniejszym zadaniu przygotuj aplikację w oparciu o podejście EF code first.

Zaimplementuj następujące końcówki:

### Dodawanie nowej recepty

- Końcówkę pozwalającą na wystawienie nowej recepty. Końcówka powinna przyjmować jako element żądania informacje o pacjencie, receptie i informacje o lekarz wystawionych na danej receptie.
- Jeśli pacjent przekazany w żądaniu nie istnieje, wstawiamy nowego pacjenta do tabeli Pacjent.
- Jeśli lek podany na receptie nie istnieje, zwracamy błąd.
- Recepta może obejmować maksymalnie 10 leków. W innym wypadku zwracamy błąd.
- Musimy sprawdzić czy  $\text{DueDate} \geq \text{Date}$

**Przykład żądania:**

```
{
  "patient": {
    "IdPatient": 1,
    "FirstName": "Jan",
    //...
  },
  "medicaments": [
    {"idMedicament": 1, "Dose": 3, "Description":
"Some desc..." }
  ],
  "Date": "2012-01-01",
  "DueDate": "2012-01-01"
}
```

### Dane na temat pacjenta:\*\*

Przygotuj końcówkę pozwalającą na wyświetlenie wszystkich danych na temat konkretnego pacjenta, wraz z listę recept i leków, które pobrał. Chcielibyśmy, żeby odpowiedź uwzględniała wszystkie informacje na temat leków i lekarzy. Dane na temat recept powinny być posortowane po polu DueDate.

### Przykład:

```
{
  "IdPatient": 1,
  "FirstName": "Jan",
  //...
  "Prescriptions": [
    {
      "IdPrescription": 1,
      "Date": "2012-01-01",
      "DueDate": "2012-01-01",
      "Medicaments": [
        {
          "IdMedicament": 1,
          "Name": "AAA",

```

```
        "Dose": 10,  
        "Description": "AAA"  
    }  
]  
"Doctor": {  
    "IdDoctor": 1,  
    "FirstName": "AAA"  
}  
}  
]
```

Spróbuj przygotować testy jednostkowe sprawdzające twoje końcówki.

### **Upewnij się, że:**

- nazewnictwo jest poprawne
- kod został podzielony na osobne warstwy i jest testowalny
- pamiętaj o zasadach REST, SOLID, DRY, YAGNI, KISS
- pamiętaj o optymistycznym/pesymistycznym blokowanie (jeśli jest ono potrzebne)