Exercises: Code First

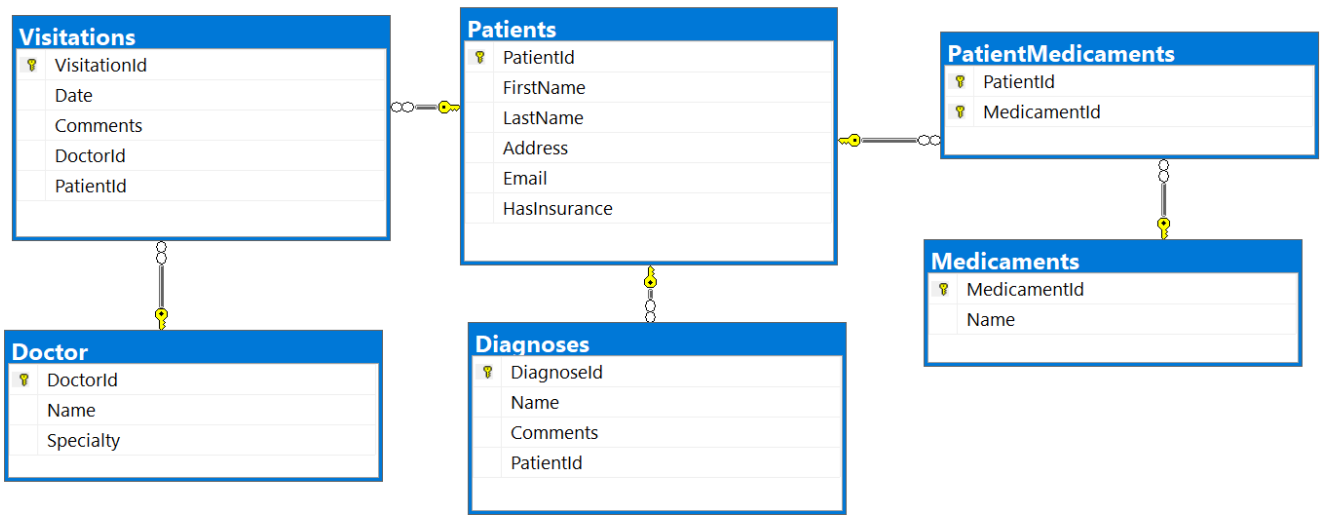
This document defines the **exercise assignments** for the "Databases Advanced – EF Core" course @ Software University.

1. Hospital Database

You went to your GP for your annual exam and you told him that you've started work as a Junior Database App Developer. It turned out he was looking for someone to make an app, which he could use to manage and store data about his patients.

Your task is to design a database using the **Code First** approach. The GP needs to keep information about his **patients**. Each patient has **first name**, **last name**, **address**, **email**, information whether he has **medical insurance** or not and should keep history about all his **visitations**, **diagnoses** and **prescribed medicaments**. Each visitation has **date** and **comments**. Each **diagnose** has **name** and **comments** for it. Each **medicament** has **name**. **Validate** all data before inserting it in the database.

Your Database should look something like this:
Remember! With Entity Framework Core you can have **different column names** from your **classes' property names**!



Constraints

Your **namespaces** should be:

- **`P01_HospitalDatabase`** – for your Startup class, if you have one
- **`P01_HospitalDatabase.Data`** – for your DbContext
- **`P01_HospitalDatabase.Data.Models`** – for your models

Note: **Do not** use separated projects, because Judje will return Compile Time Error.

Your **classes** should be:

- **`HospitalContext`** – your DbContext
- **`Patient`**:
  - PatientId

    - FirstName (up to 50 characters, unicode)
    - LastName (up to 50 characters, unicode)
    - Address (up to 250 characters, unicode)
    - Email (up to 80 characters, not unicode)
    - HasInsurance
- **`Visitation`**:
    - VisitationId
    - Date
    - Comments (up to 250 characters, unicode)
    - Patient
- **`Diagnose`**:
    - DiagnoseId
    - Name (up to 50 characters, unicode)
    - Comments (up to 250 characters, unicode)
    - Patient
- **`Medicament`**:
    - MedicamentId
    - Name (up to 50 characters, unicode)
- **`PatientMedicament`** – mapping class between Patients and Medicaments

The **collections** of mapping classes (**`ICollection<PatientMedicament>`**) must be named **Prescriptions**!

Note: Don't forget to remove the **Tools** package before uploading your project to Judge, if you have used it!
Don't use version of Entity Framework Core above 2.2.0!

Bonus Task

Make a console-based user interface, so the doctor can easily use the database.

2. Hospital Database Modification

Your GP bragged around in the hospital about the cool software you made for him. Now the hospital administration wants to modify your program so they can use it too. They want to store information about the **doctors** (**name** and **specialty**). Each doctor can perform **many visitations**. Make the necessary changes in the **database** to satisfy the new needs of the hospital administration.

Constraints

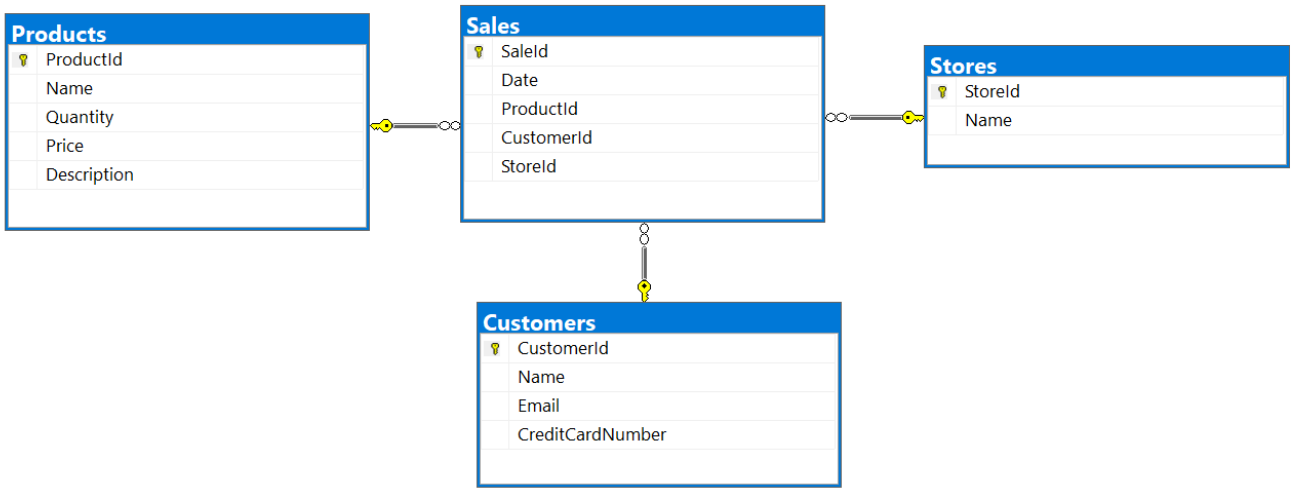
Keep the **namespaces** from the previous task and only add the class **Doctor** and change the class **Visitation** accordingly. The doctor's **name** and **specialty** should be up to 100 **characters** long, **unicode**.

**Bonus Task

Make an authentication system for doctors. Each **doctor** should be able to log in with his **email** and **password**. Choose what information each doctor should have access to and hide the rest.

3. Sales Database

Create a database for storing data about sales using the Code First approach. The database should look like this:



Constraints

Your **namespaces** should be:

- **`P03_SalesDatabase`**
- **`P03_SalesDatabase.Data`**
- **`P03_SalesDatabase.Data.Models`**

Your **classes** should be:

- **`SalesContext`** – your DbContext
- **`Product`**:
  - ProductId
  - Name (up to 50 characters, unicode)
  - Quantity (real number)
  - Price
  - Sales
- **`Customer`**:
  - CustomerId
  - Name (up to 100 characters, unicode)
  - Email (up to 80 characters, not unicode)
  - CreditCardNumber (string)
  - Sales
- **`Store`**:
  - StoreId
  - Name (up to 80 characters, unicode)
  - Sales
- **`Sale`**:
  - SaleId
  - Date
  - Product

- Customer
- Store

Bonus Task

Write a **seed method** that fills the database with sample data (randomly generated).

4. Products Migration

To apply migrations, you will need two packages:

- EntityFrameworkCore.Tools 2.2.0
- EntityFrameworkCore.SqlServer.Design 1.1.6

For table **`Products`** add string column **`Description`**, up to 250 symbols. Use migrations. The migration should be named: **`"ProductsAddColumnDescription"`**. Add a default value for the description property: **`"No description"`**.

Note: Don't forget to remove the **Tools** and **Design** packages before uploading your project to Judge!

5. Sales Migration

For table **`Sales`** make **`Date`** column with default value **`GETDATE()`** function, called from the database, not the application. Use explicit migration. Do **not** use **`DateTime.Now`**! Name the migration "**`SalesAddDateDefault`**".

After that, open your table data and see if the default value is applied or not.

Keep your **namespaces** from **Task 3** and do both **Task 4** and **Task 5** before testing your project in **Judge**. Make sure to upload your **migrations** too!

Note: Don't forget to remove the **Tools** and **Design** packages before uploading your project to Judge!