**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 6

**З дисципліни:** *“Конструювання програмного забезпечення”*

**Лектор:**

доцент каф. ПЗ

Сердюк П.В.

**Виконав:**

ст. гр. ПЗ-32

Шмідт Р.Т.

**Прийняв:**

доцент каф. ПЗ

Сердюк П.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2021 р.

∑= \_\_\_\_\_ .

Львів – 2021

**Тема:** ADO.Net ORM. Data access through ADO.Net EF.

**Мета:** Лабораторна робота полягає у вивченні різних видів даних у C#: класів, інтерфейсів.

**Теоретичні відомості**

**ADO.NET** – це набір класів, які надають служби доступу до даних програмістів, які використовують платформу .NET Framework. ADO.NET має багатий набір компонентів для створення розподілених додатків, які спільно використовують дані.

Це невід'ємна частина платформи .NET Framework, яка надає доступ до реляційних даних, XML-даних та даних додатків. ADO.NET задовольняє різні потреби розробників, включаючи створення клієнтських додатків баз даних, а також бізнес-об'єктів середнього рівня, що використовуються програмами, засобами, мовами та браузерами.

**ORM –** технологія [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка зв'язує [бази даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) з концепціями [об'єктно-орієнтованих](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) мов програмування, створюючи «віртуальну об'єктну базу даних».

Вирішення проблеми зберігання даних існує - це реляційні системи управління базами даних.

Використання реляційної бази даних для зберігання об'єктно-орієнтованих даних призводить до семантичного розриву, змушуючи програмістів писати програмне забезпечення, яке має вміти обробляти дані в об'єктно-орієнтованому вигляді, а зберігати ці дані у реляційній формі. Ця постійна необхідність перетворення між двома різними формами даних як сильно знижує продуктивність, а й створює труднощі для програмістів, оскільки обидві форми даних накладають обмеження друг на друга.

**Завдання**

* вивести на UI дані із таблиць. Дані вибирати відповідно до списку сценаріїв та планованого графічного інтерфейсу
* Реалізувати додавання, оновлення та видалення даних за допомогою фреймворків ADO.Net EF
* Реалізувати підходи DB First та Code First
* Реалізувати міграцію для Code First. Для цього змінити модель і згенерувати міграцію

**Виконання**

1. Реалізувати підходи DB First та Code First

# 

Рис. 1. Попереднє створення бази даних для підходу DatabaseFirst



Рис. 2 Запуск команди для побудови класів відповідно до БД

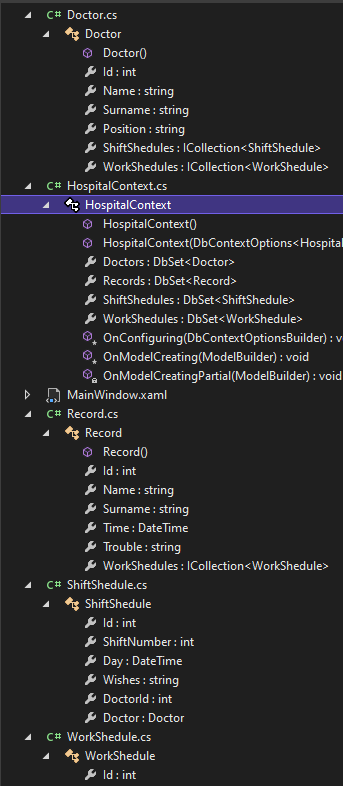


Рис. 3 Результат виконання даної команди

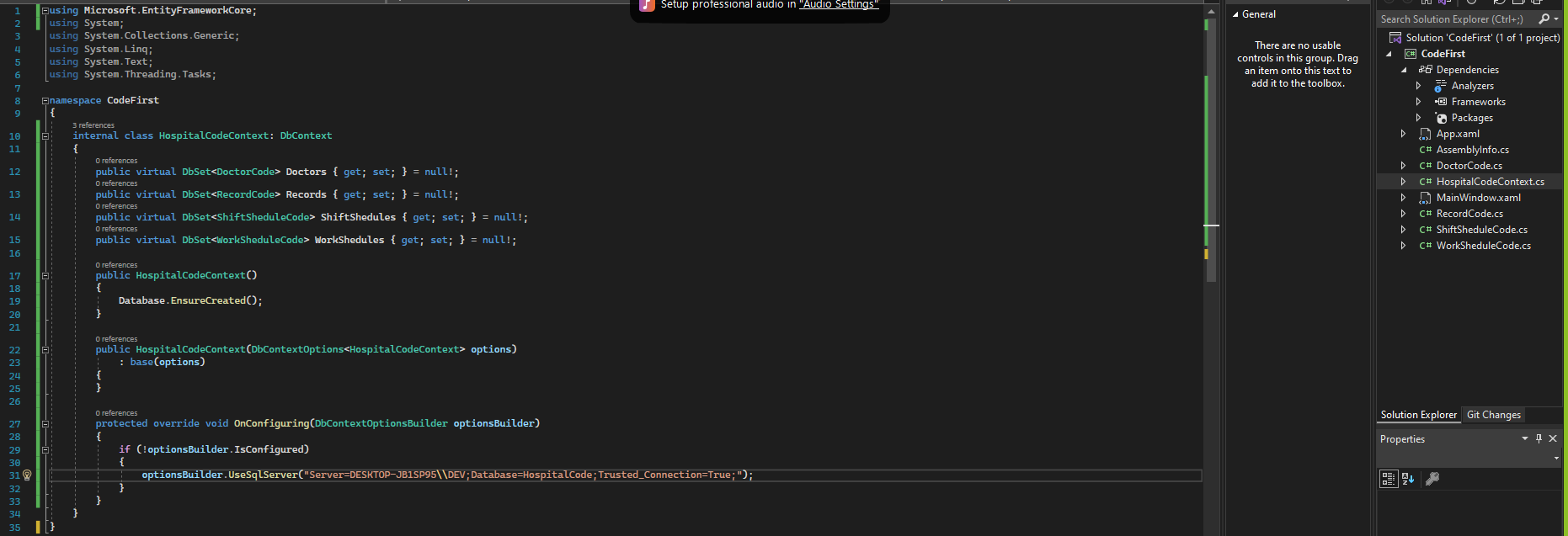


Рис. 4 Результат створення контекст та класів для підходу CodeFirst

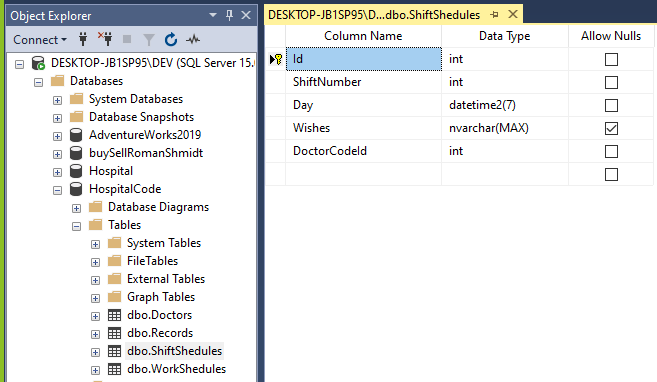


Рис. 5 Результат створення БД за підходом CodeFirst

1. Cтворення міграції для CodeFirst

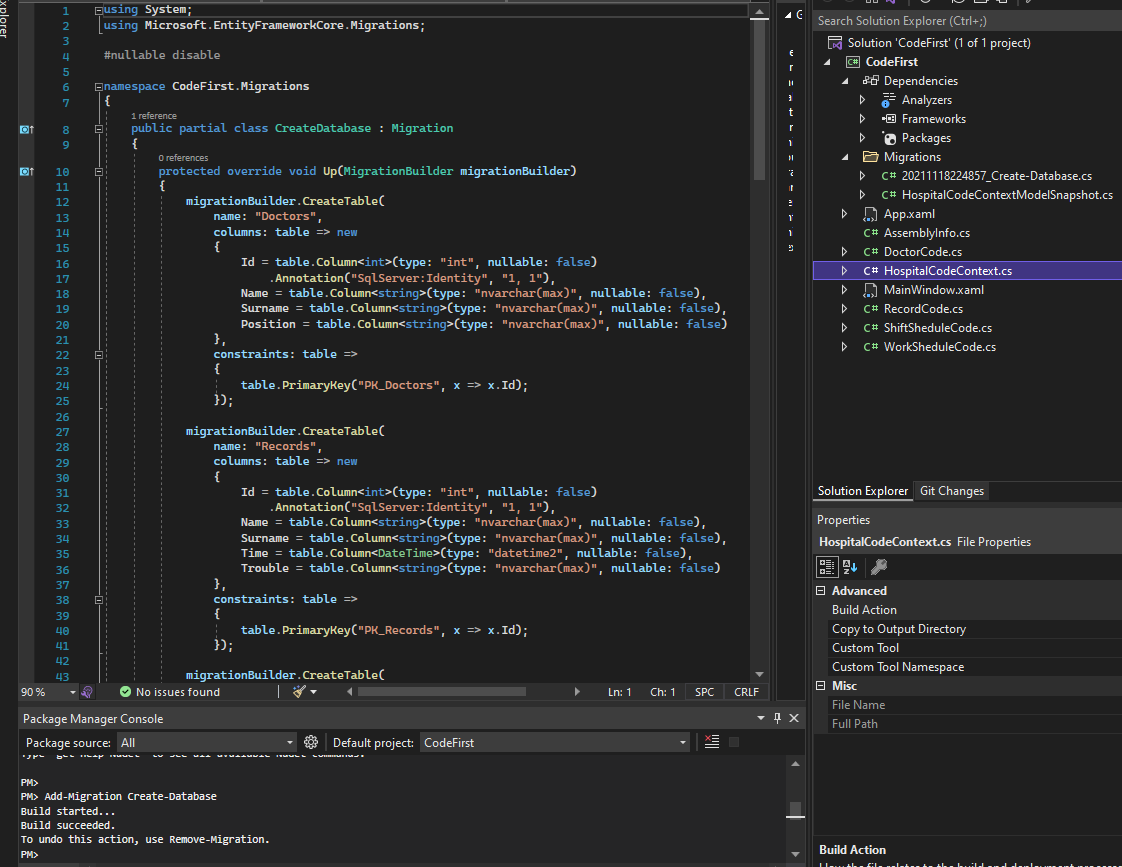


Рис. 6 Створення першої міграції для побудови snaphot-у бази

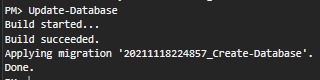


Рис. 7 Затвердження міграції

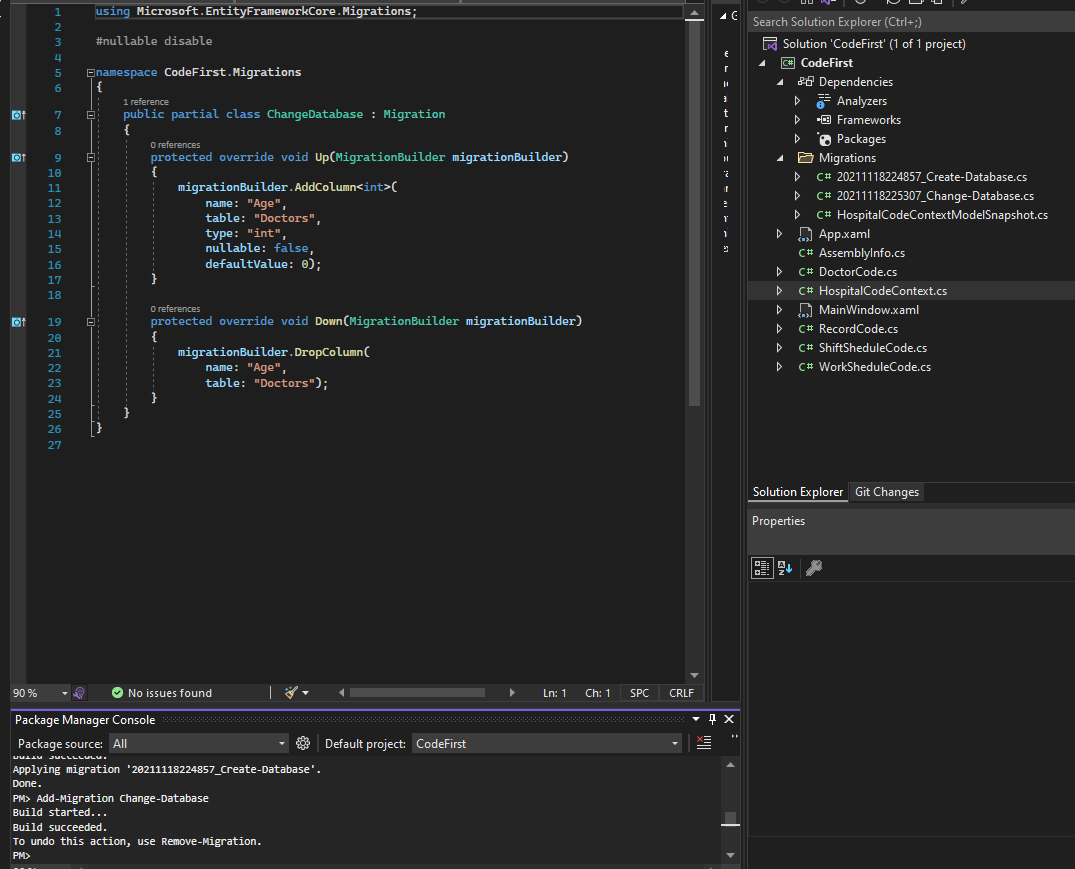


Рис. 8 Створення міграції з новими змінами до бази

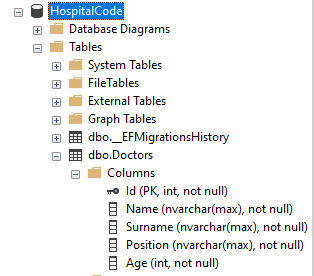


Рис. 9 Результат міграції

1. Реалізував вивід на UI даних з таблиць відповідно до можливих сценаріїв, пов’язаних з індивідуальним завданням

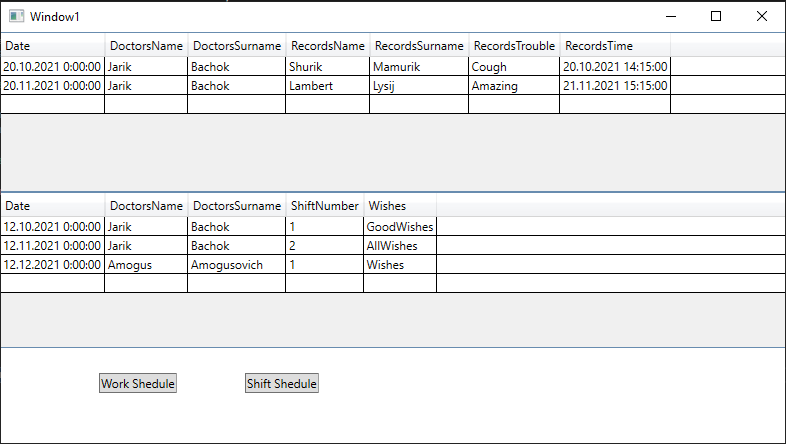


Рис. 10 Результат виведення даних на UI

1. Реалізував додавання, зчитування, оновлення та видалення даних

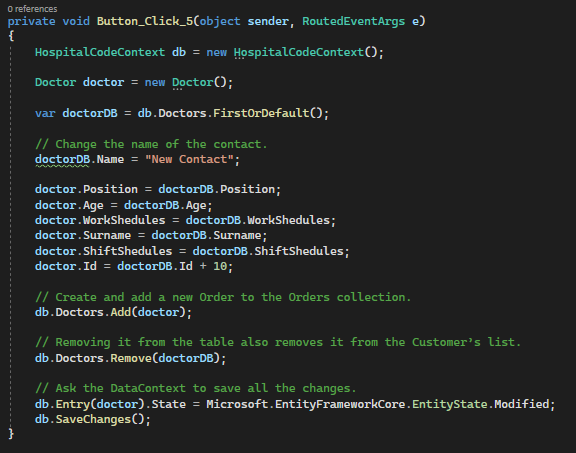


Рис. 11 CRUD-операції

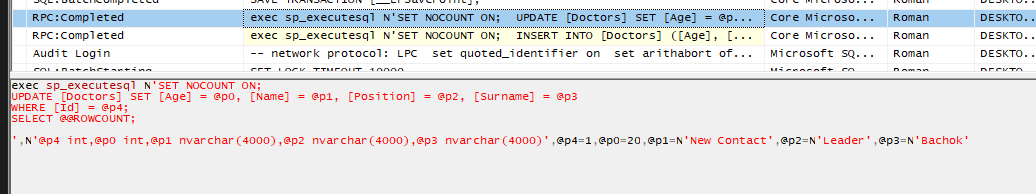


Рис. 12 Показ запиту Update в SQL Profiler

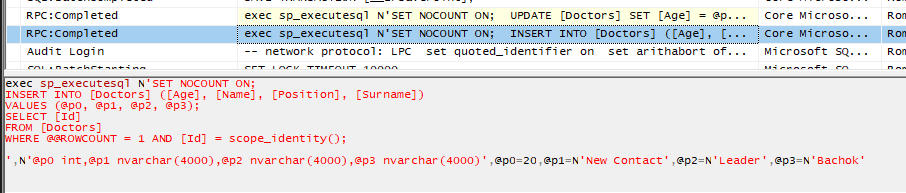


Рис. 13 Insert в SQL Profiler

**Висновки**

Виконуючи дану лабораторну роботу я ознайомився із фреймворком ADO.NET EF; створив моделі згідно індивідуального завдання з баз даних, в тому числі використовуючи підхід code first; реалізував можливість створення, зчитування, оновлення та видалення.