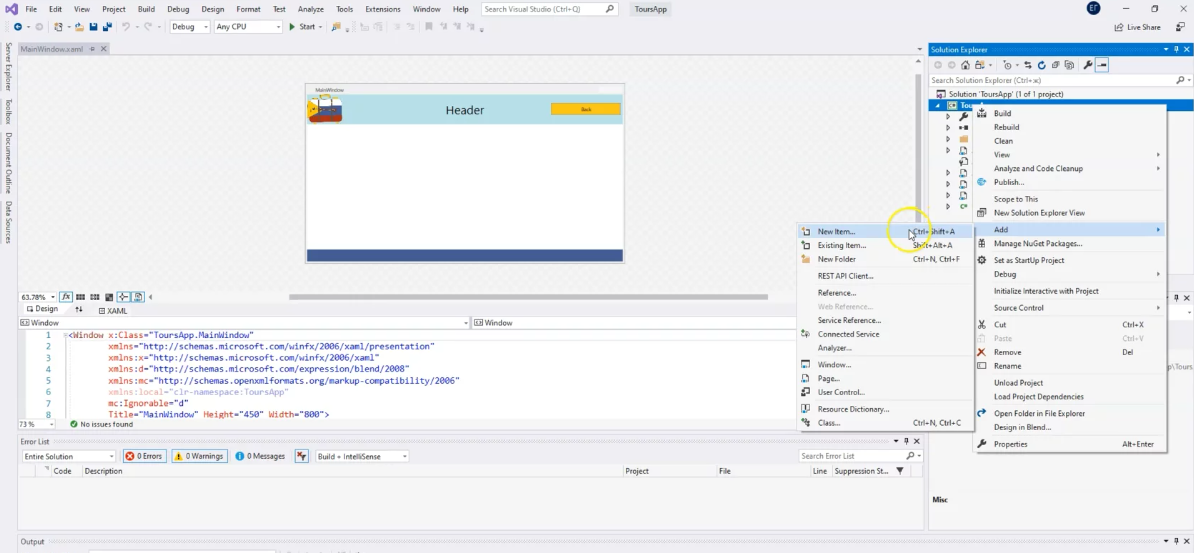
**РАБОТА С БАЗОЙ ДАННЫХ В ПРИЛОЖЕНИИ: ЧТЕНИЕ, ДОБАВЛЕНИЕ, РЕДАКТИРОВАНИЕ, УДАЛЕНИЕ ДАННЫХ (ЧАСТЬ 1)**

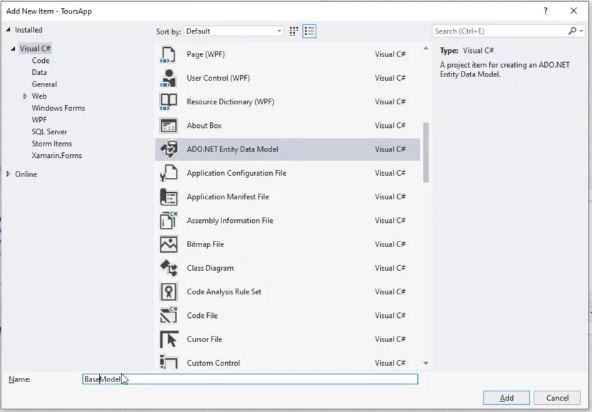
Тему работы с базой данных в приложении будем рассматривать на примере отображения, добавления, редактирования и удаления данных. Для отображения определенных данных система обращается к базе данных, получает ответ, а затем преобразует его в удобный для пользователя вид, который настраивает разработчик. Для работы с базой данных в приложении мы будем использовать Entity Framework, который позволяет работать с базой данных через объектно-ориентированный подход. Он предоставляет ряд существенных преимуществ: вам не нужно беспокоиться о коде доступа к данным, а также знать детали работы СУБД SQL Server и синтаксиса языка структурированных запросов SQL. Вместо этого вы работает с таблицами базы данных как с классами С#, а с полями этих таблиц, как со свойствами классов, используя вместо SQL запросов более удобный подход — LINQ. Entity Framework берет на себя обязанности по преобразованию кода C# в SQL инструкции

**Использование Windows Presentation Foundation (WPF) для создания интерактивных настольных приложений**

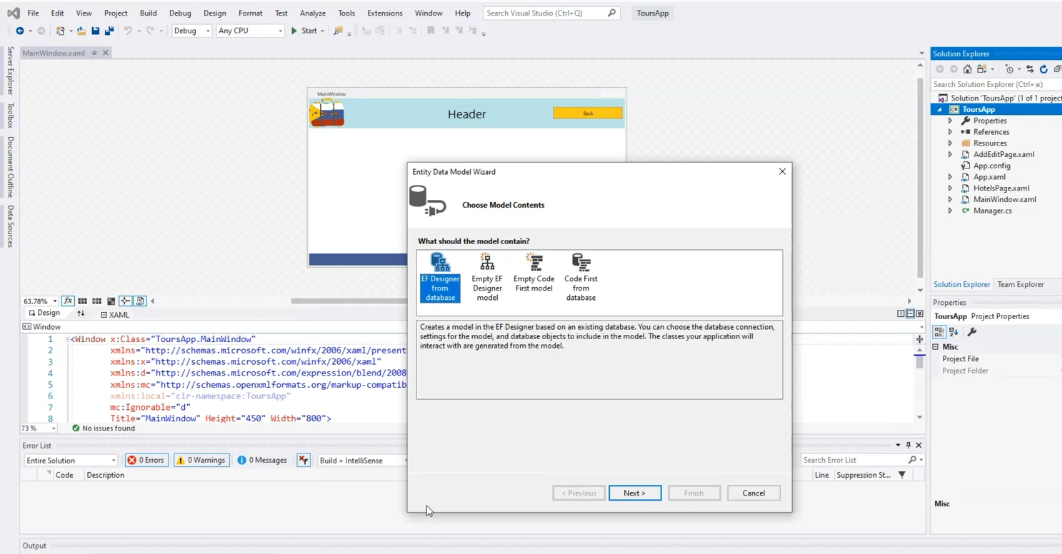
**1.** New Item…



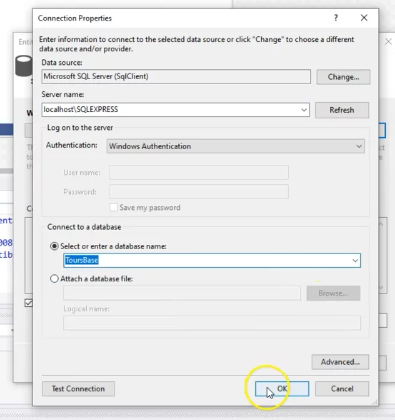
**2.** Выбираем ADO.NET Entity Data Model и даем ей название



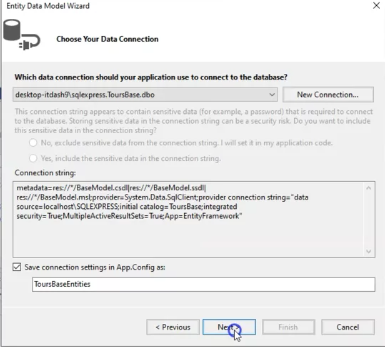
**3.** Выбираем EF Designer from database



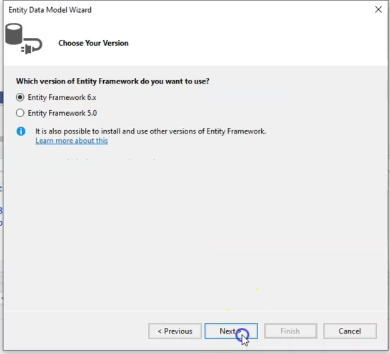
**4.** Создаем новое подключение и выбираем базу данных



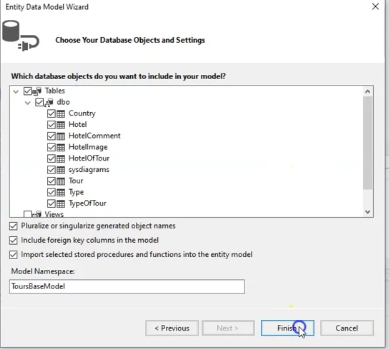
**5.** Нажимаем Next



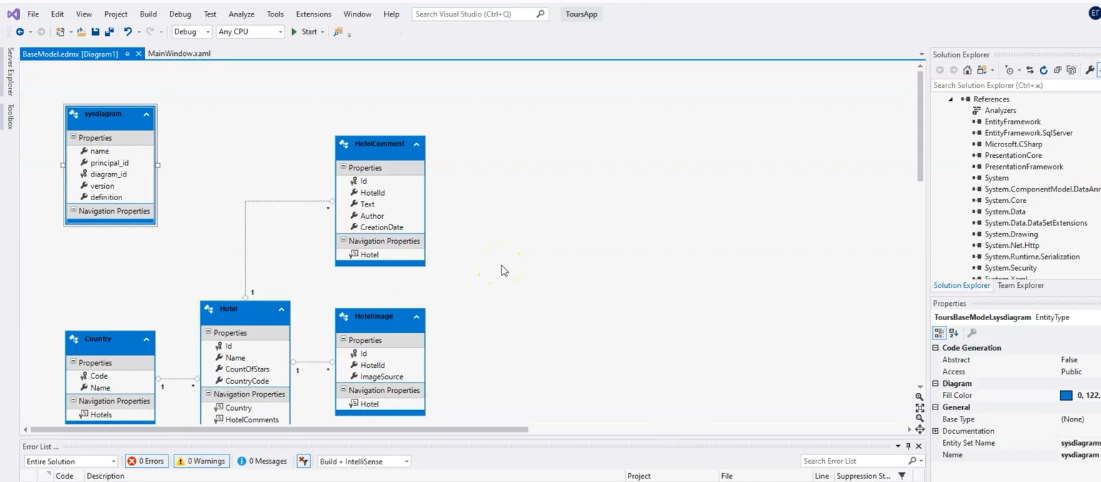
**6.** Нажимаем Next



**7.** Выбираем все таблицы из списка



**8.** Модель готова

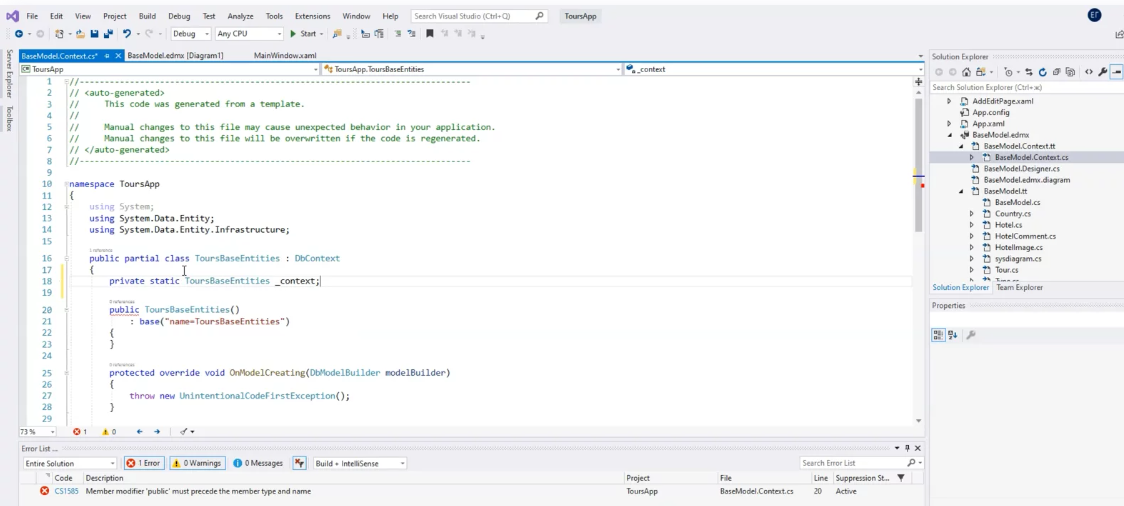


**Обращение к модели данных. Паттерн SingleTone. Чтение, добавление, редактирование, удаление данных**

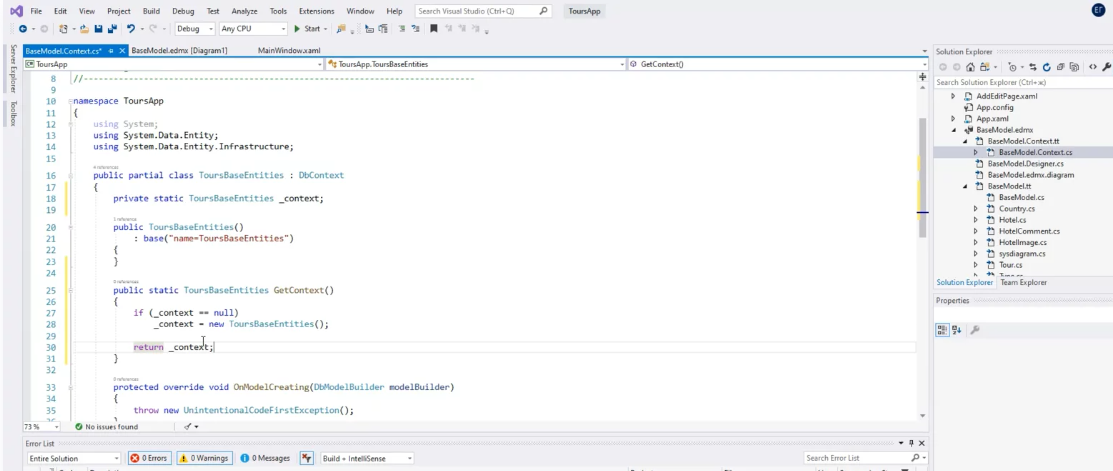
**1.** Открываем файл BaseModel.Context.cs



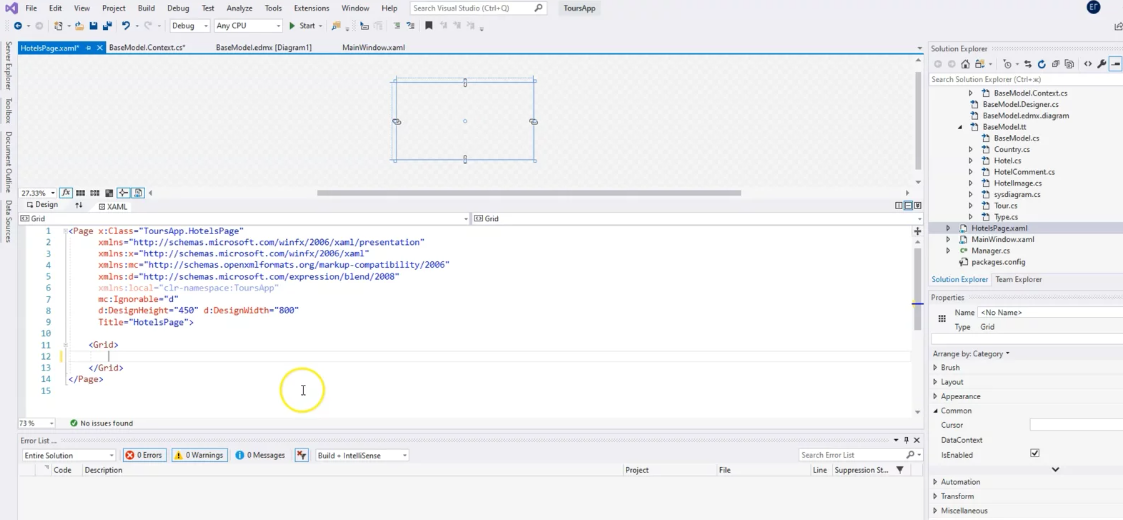
**2.** Добавляем приватное статичное поле, которое будет контекстом



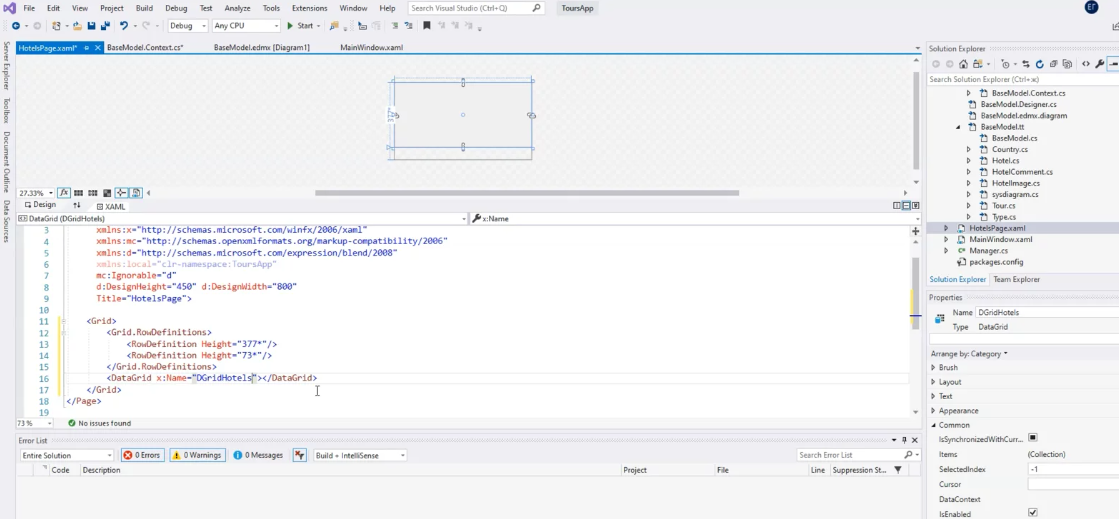
**3.** Добавляем метод получения экземпляра этого контекста



**4.** Для отображения данных открываем страницу HotelPage.xaml

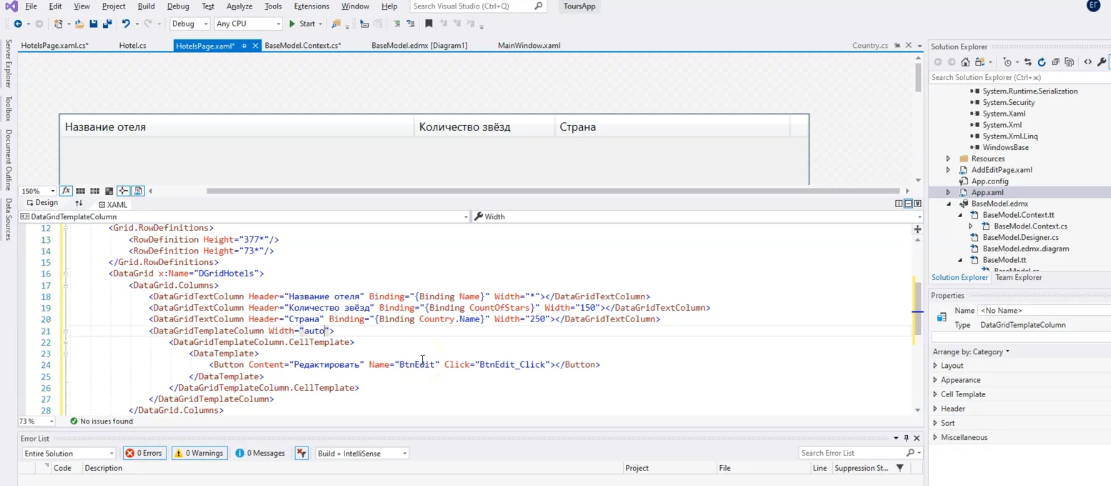


**5.** Размечаем Grid на две части



**6.** Cписок данных выводится по столбцам, которые прописывает разработчик. Устанавливаем их с помощью свойства DataGrid.Columns, которое и описывает набор столбцов.

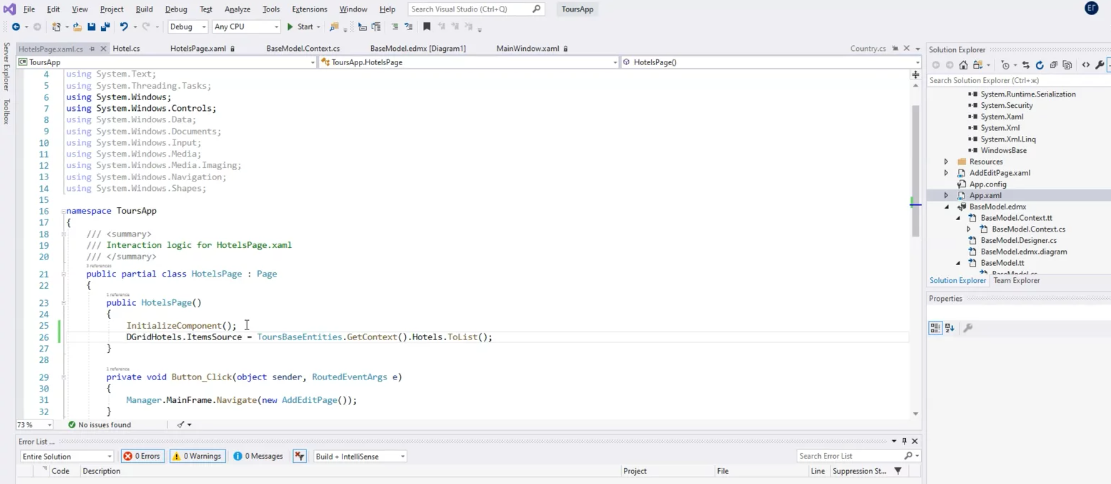
* DataGridTextColumn для текстовых столбцов
* DataGridTemplateColumn для более сложного представления данных, например, кнопки



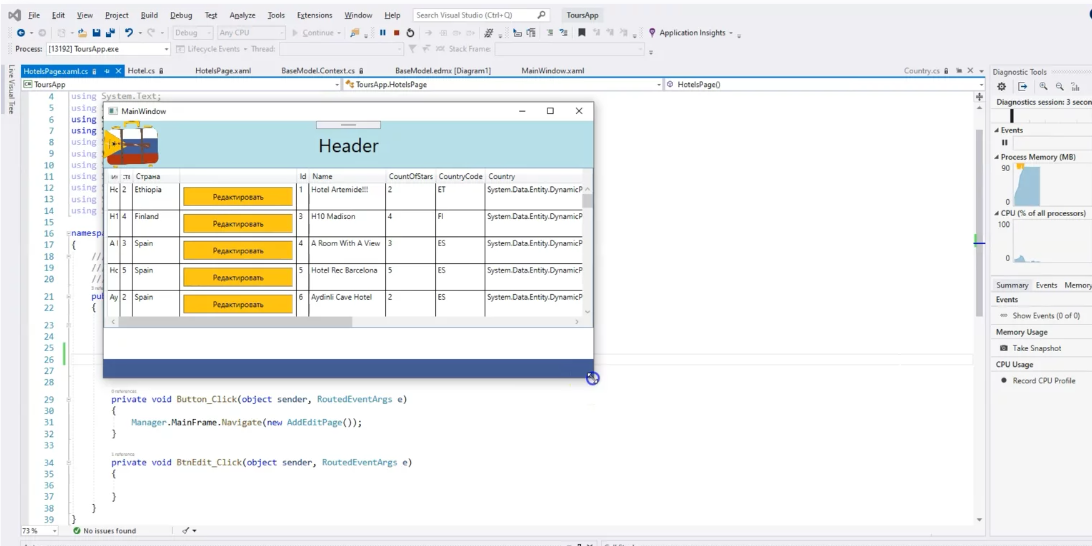
**7.** Далее загрузим список отелей в коде в таблицу:

a) Нажимаем F7

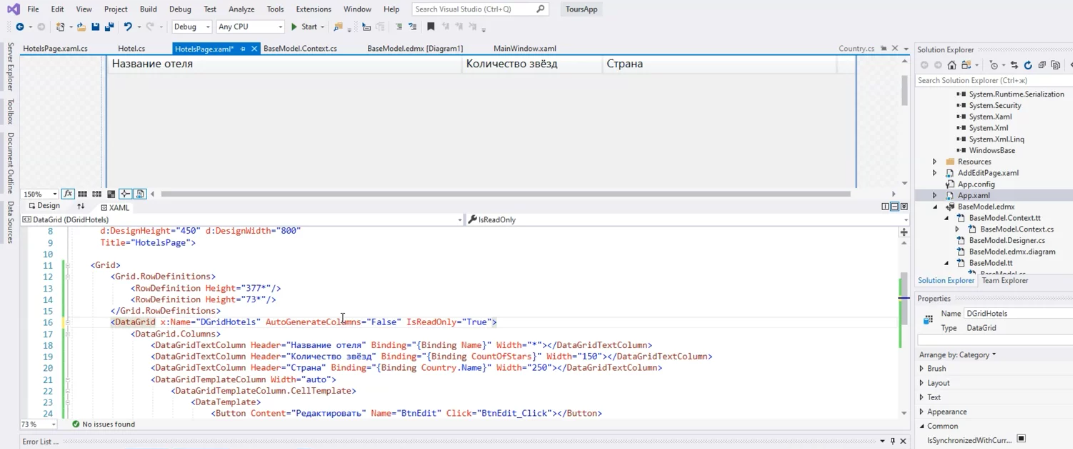
b) Обращаемся к контексту модели



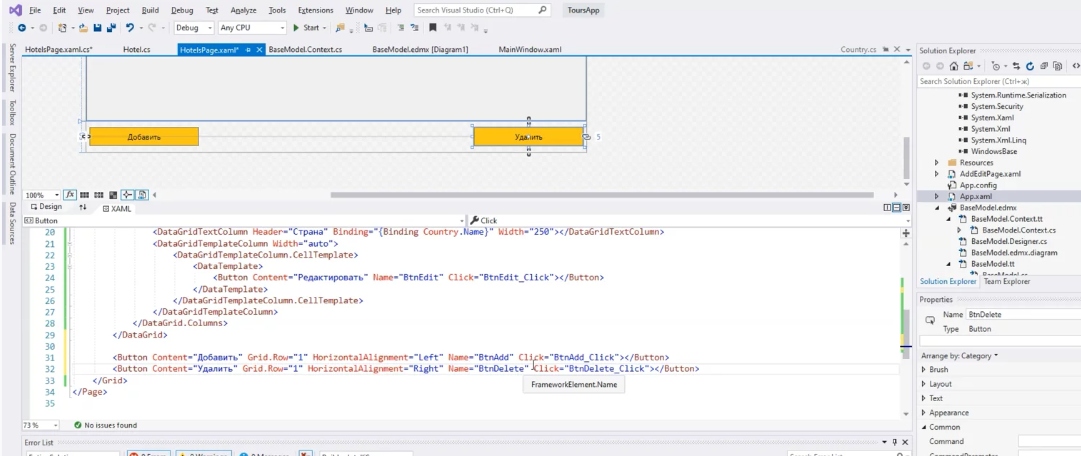
**8.** Пробуем запустить программу и видим набор данных уже в приложении



**9.** Для отключения загрузки всех свойств объектов, необходимо прописать AuroGenerateColumns="False" и IsReadOnly="True"



**10.** Также добавим кнопки для добавления и удаления во второй строке сетки. Дадим им имена и обработаем нажатия

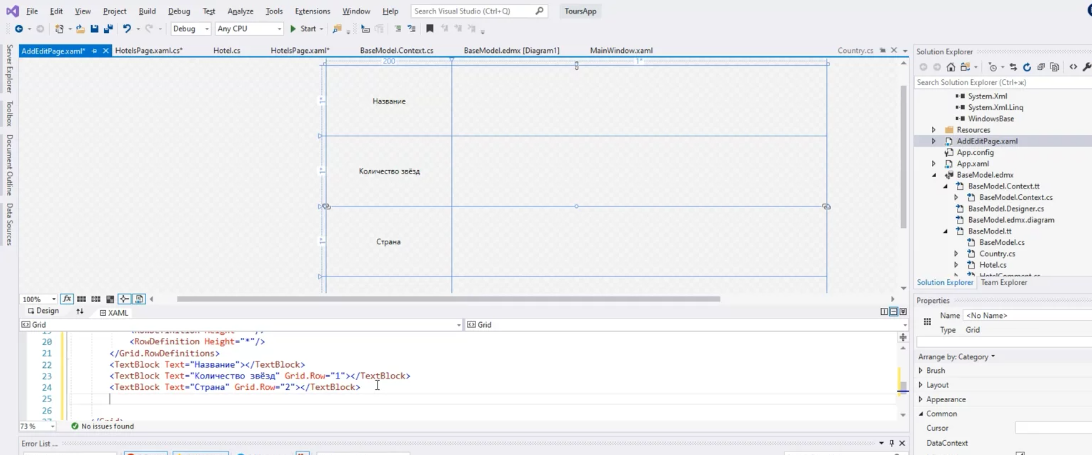


**11.** Информация об отелях может меняться, поэтому важно реализовать функции добавления, редактирования и удаления. Для этого используем уже созданную нами вторую страницу, добавив необходимые элементы управления

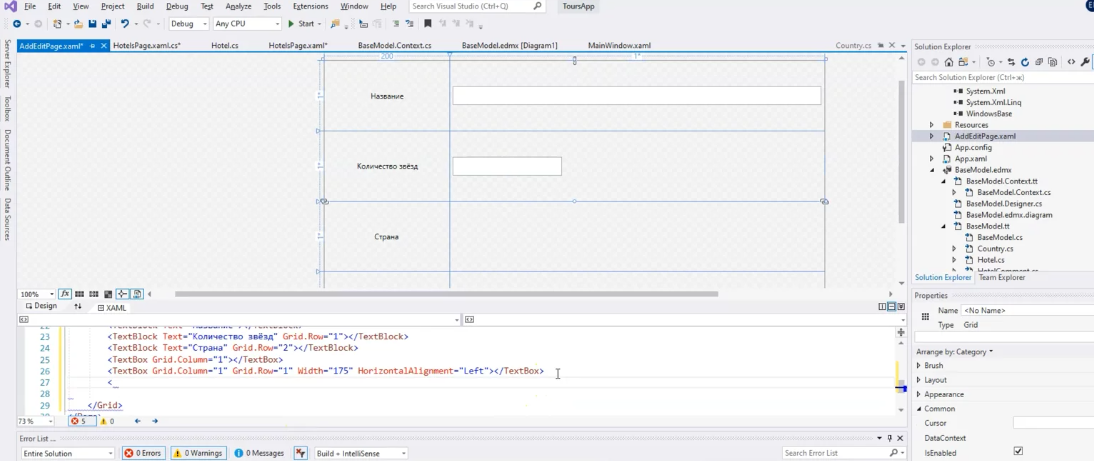
a) Сверстаем сетку, состоящую из трех строк и двух столбцов

b) Разместим элементы для ввода данных:

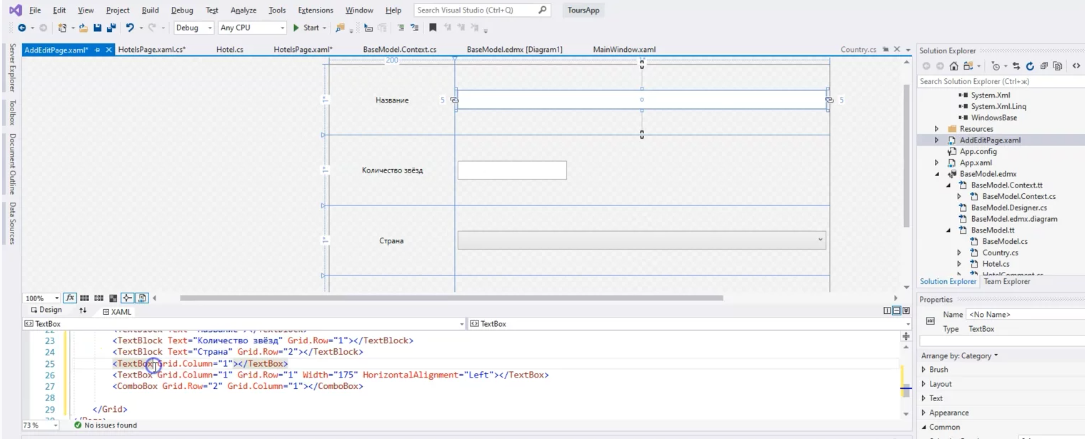
* Текстовые блоки для отображения подсказок, что именно вводить



* TextBox для ввода

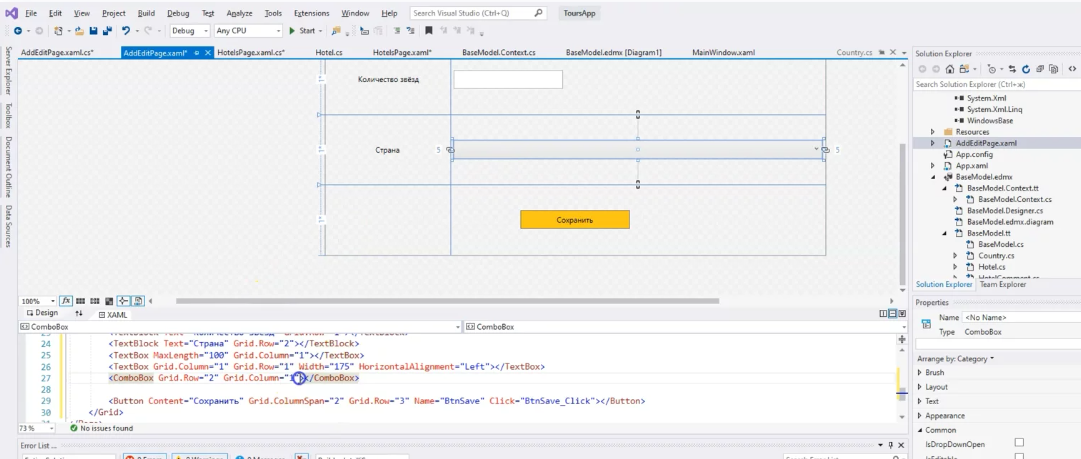


* ComboBox для выпадающего списка стран

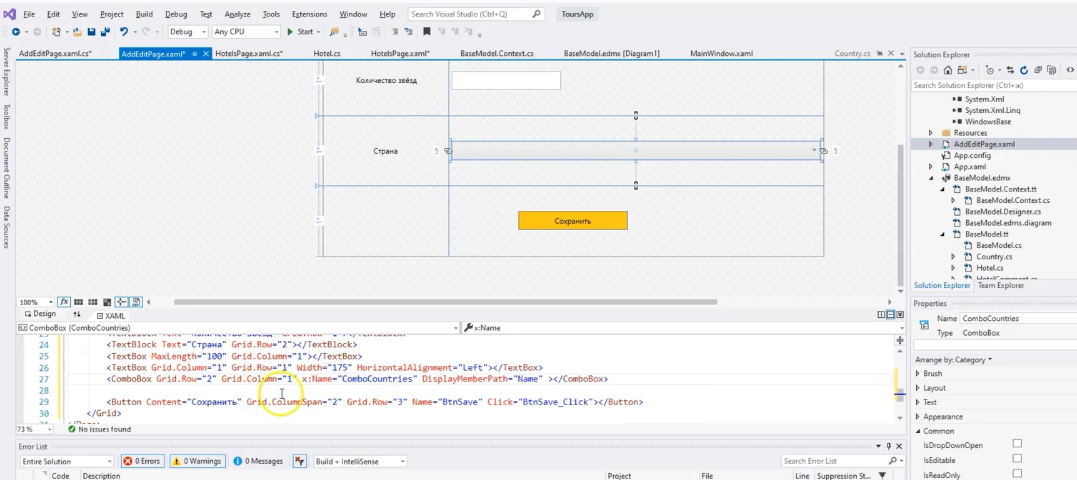


c) Устанавливаем максимальное число символов для текст-боксов равных максимальному числу символов в базе данных

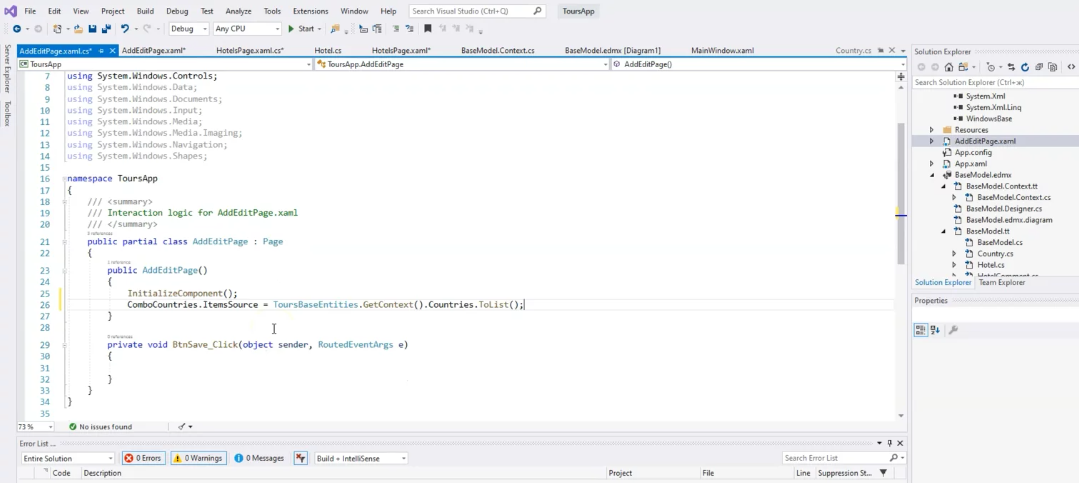
d) Добавим кнопку для сохранения изменений, дадим ей имя и обработаем нажатие



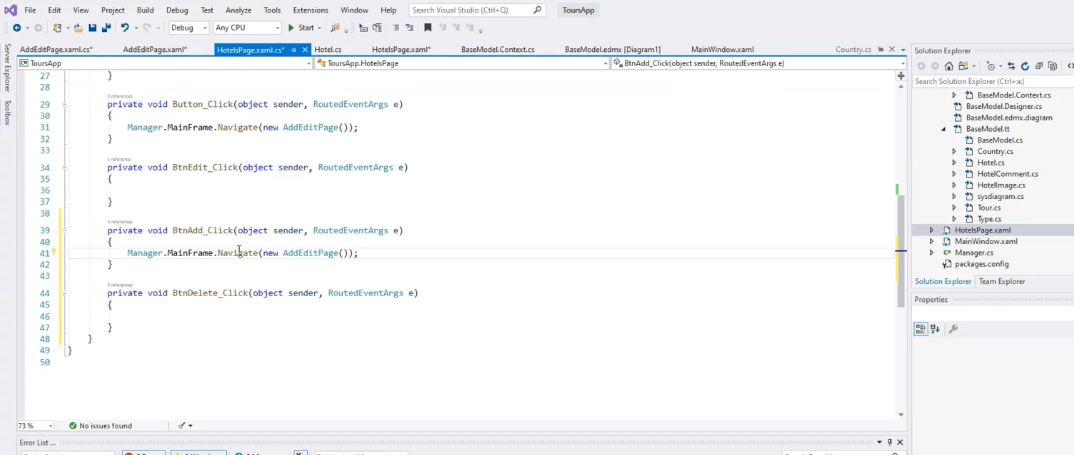
e) Доработаем комбо-бокс для выпадающего списка: дадим ему имя, укажем отображаемое свойство



И загрузим список стран (для этого в коде прописываем соответствующую команду)



**12.**И, наконец, сделаем переход на страницу добавления со страницы списка отелей



**13.** Проверяем работоспособность

