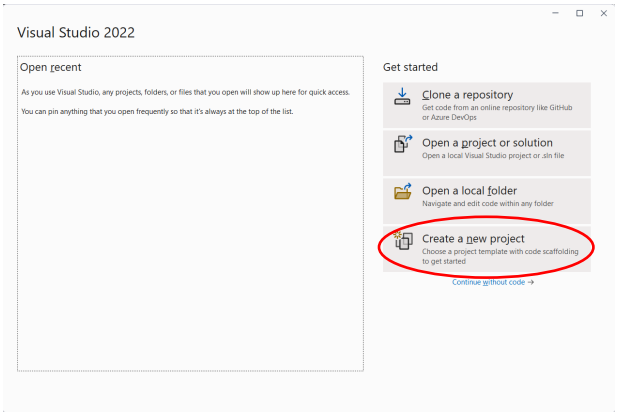
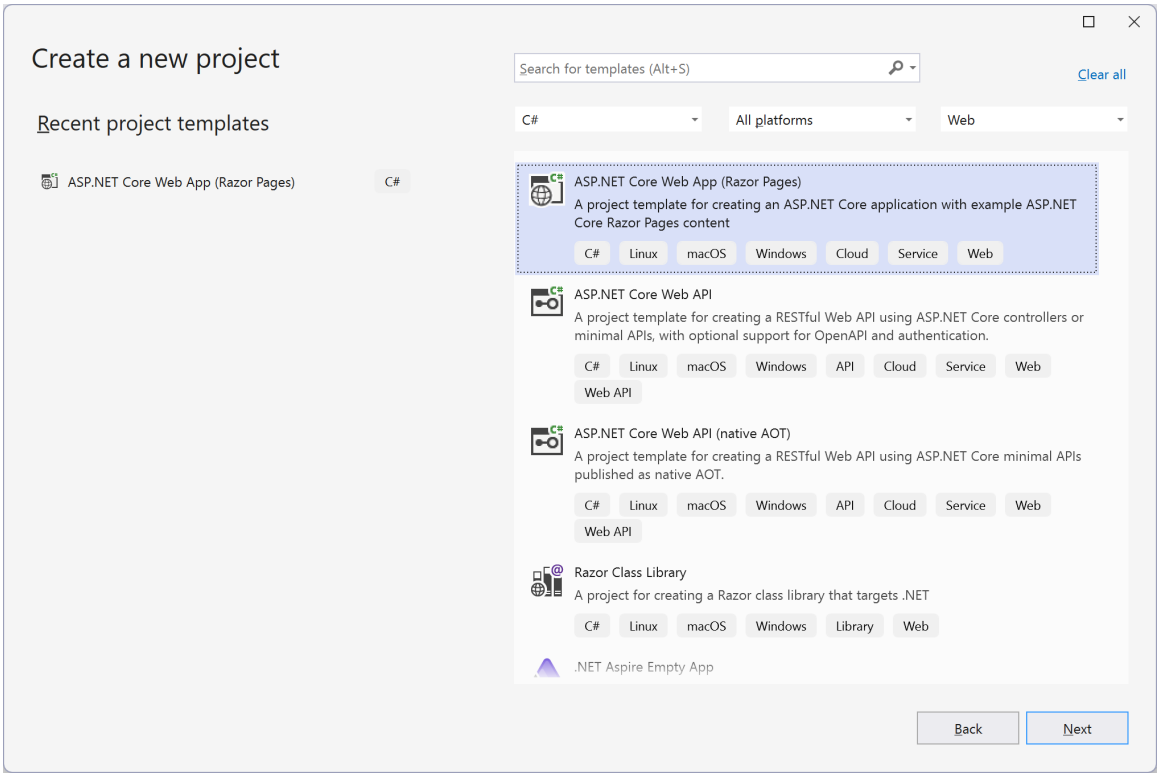
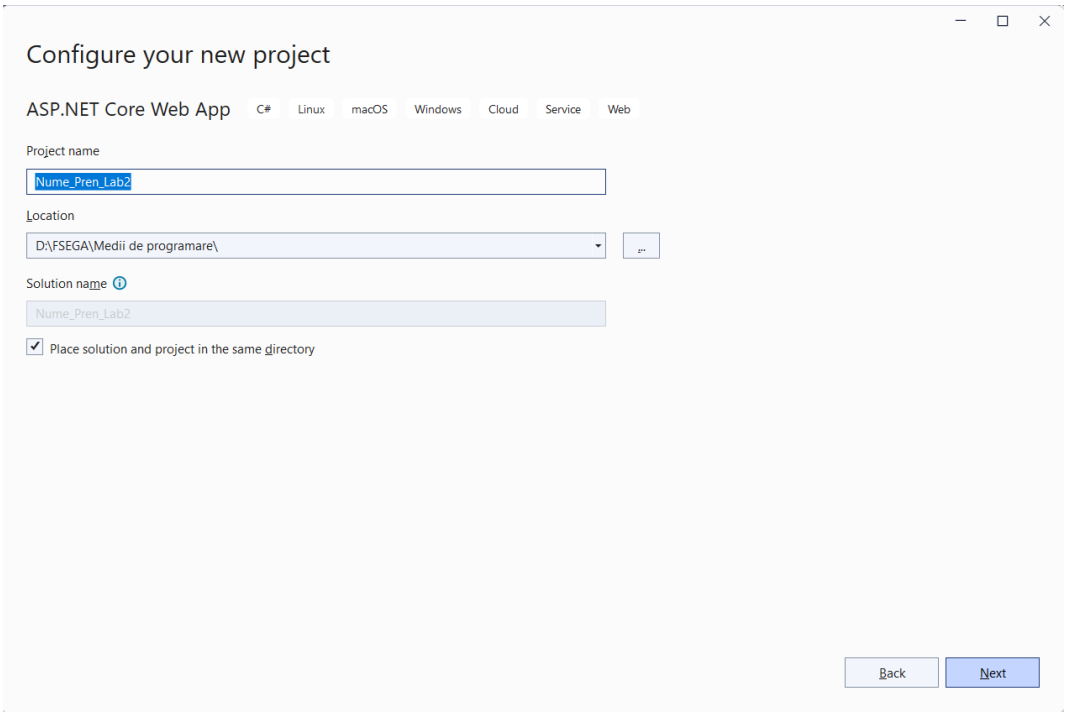
Laborator 2 – Aplicatii Web cu ASP.NET Core si Entity Framework Code First

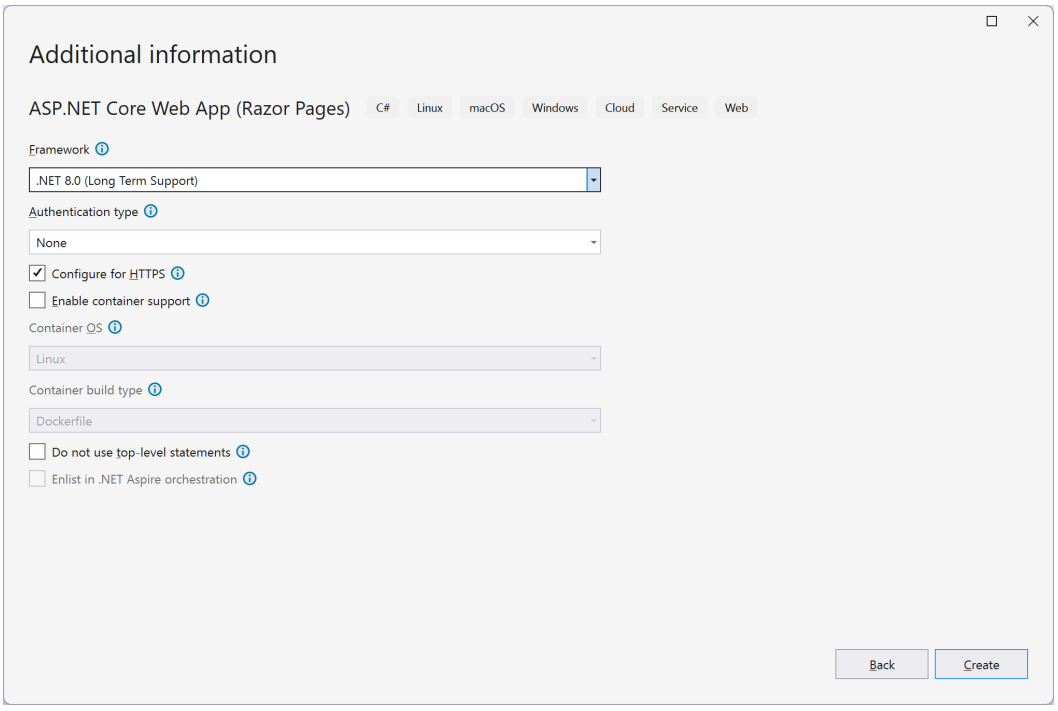
1. Se creează un nou proiect de tipul **ASP.NET Core Web App** si se denumeste **Nume_Pren_Lab2**



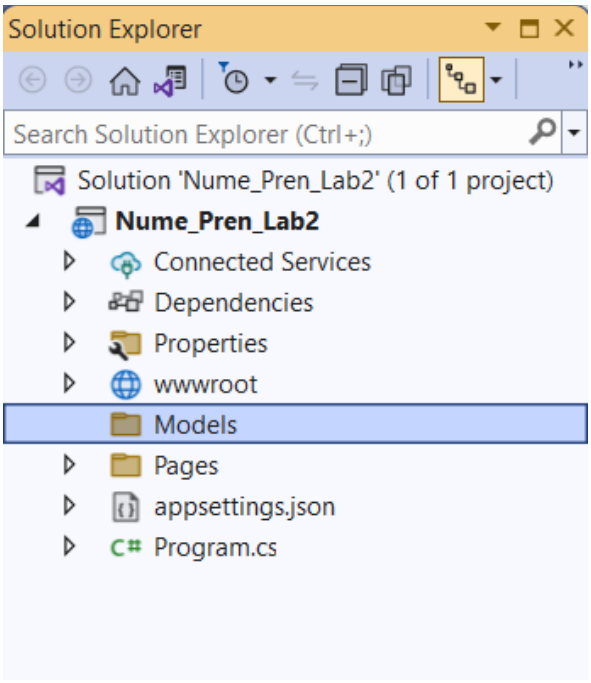




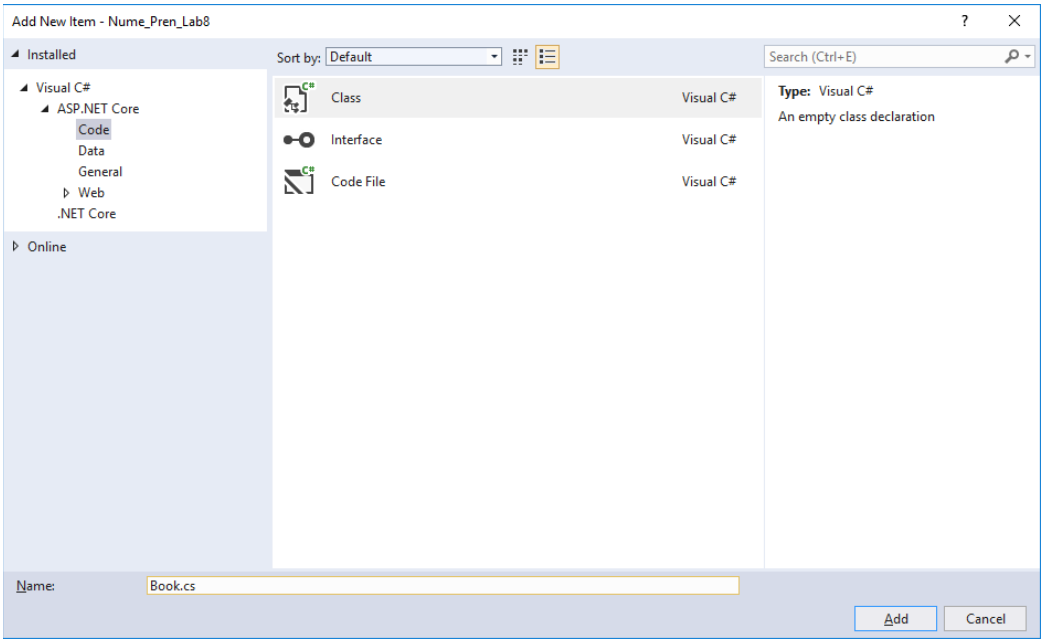
2. In fereastra urmatoare lasam optiunile implicite si apasam butonul Create



3. In continuare vom crea modelul EF. Cream un nou director in proiectul nostru astfel: In fereastra **Solution Explorer**, facem click dreapta pe numele proiectului **Nume_Pren_Lab2**, iar din meniul contextual selectam **Add -> New Folder.** Vom denumi directorul creat **Models.**



4. Facem click dreapta pe numele directorului Models, selectam **Add -> Class.** Vom denumi clasa **Book.**

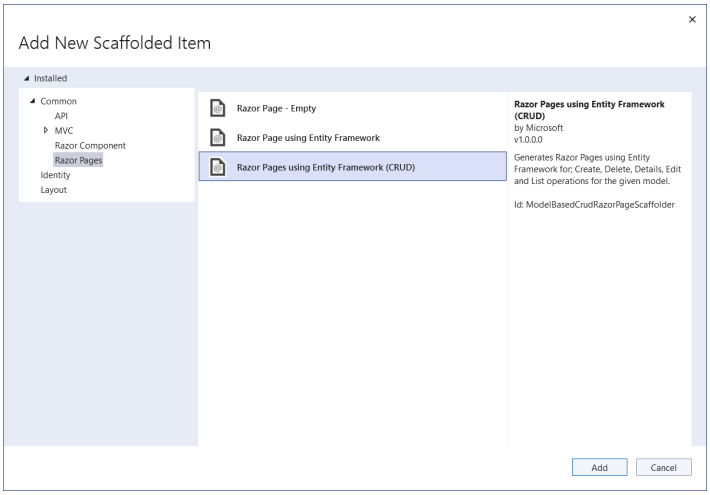


5. In clasa **Book** adaugam urmatoarele proprietati:

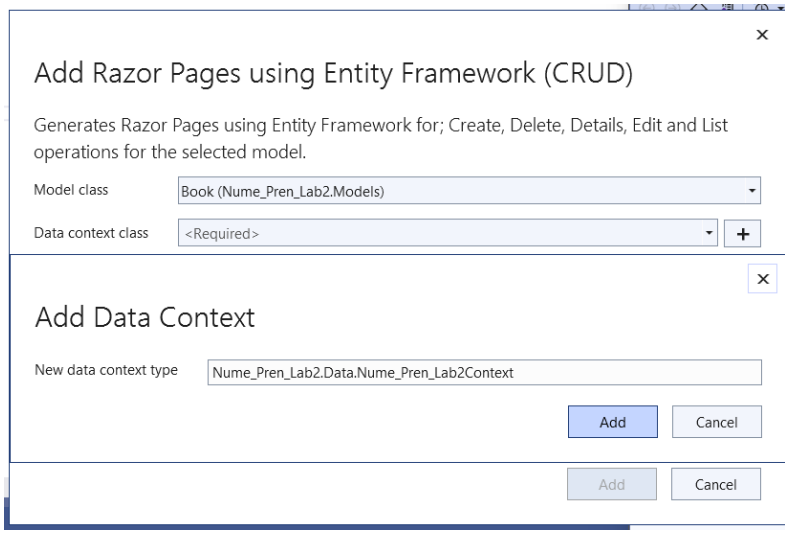
```
namespace Nume_Pren_Lab2.Models
{
    public class Book
    {
        public int ID { get; set; }
        public string Title { get; set; }
```

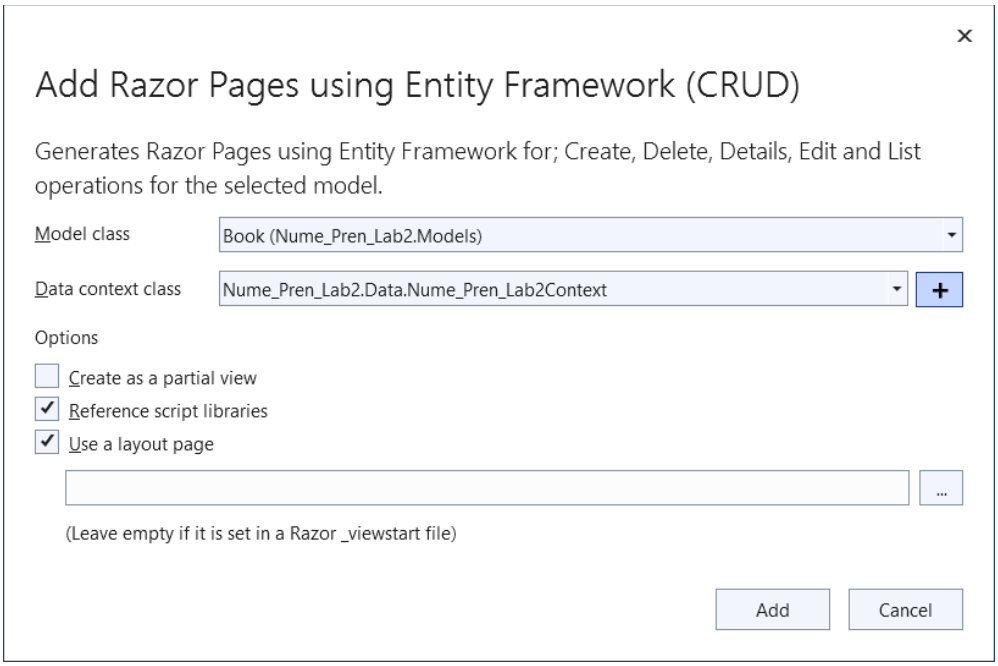
```
        public string Author { get; set; }
        public decimal Price { get; set; }
    }
}
```

6. In ferestra **Solution Explorer**, adaugam un subdirector la directorul **Pages** apasand click dreapta pe numele directorului Pages, selectam **Add ->New Folder.** Denumim noul subdirector **Books**
7. Facem click dreapta pe subdirectorul **Pages/Books** si selectam **Add > New Scaffolded Item** si selectam **Razor Pages using Entity Framework(CRUD)**

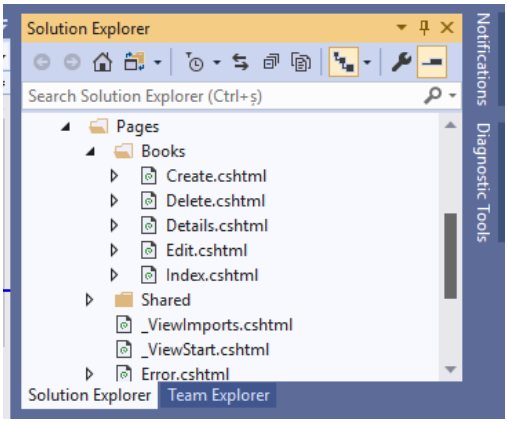


8. In fereastra urmatoare selectam din lista derulanta de la **Model Class** – Book(Nume_Pren_Lab2.Models). La **Data context class**, selectam semnul “+” si lasam numele generat Nume_Pren_Lab2**.Data.**Nume_Pren_Lab2Context

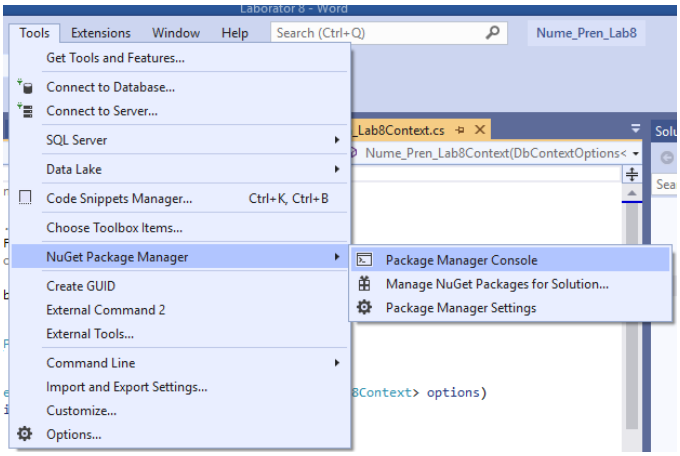




9. Dupa apasarea Add se vor crea in directorul Data fisierul Nume_Pren_Lab2Context iar in subdirectorul Pages/Books urmatoarele pagini:



10. Vom crea baza de date pe baza modelului creat anterior, utilizand optiunea de migrare din Package Manager Console(PMC). Astfel selectam din meniul **Tools**, optiunea **NuGet Package Manager ->Package Manager Console**



11. In PMC introducem urmatoarele instructiuni

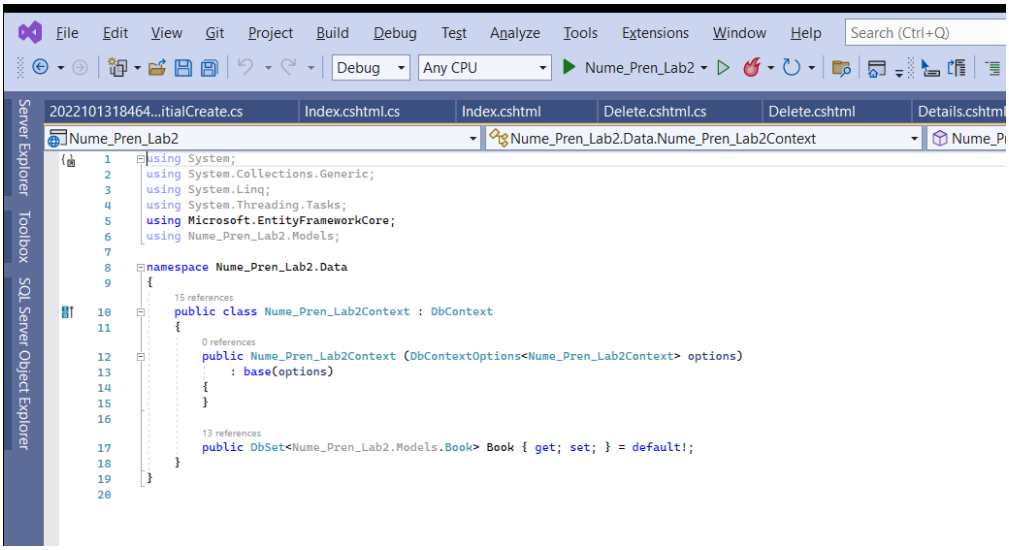
```
Add-Migration InitialCreate
Update-Database
```

Argumentul `InitialCreate` este utilizat pentru a da un nume migrarii. Putem folosi orice nume.

Comenzile vor genera urmatorul warning: "No type was specified for the decimal column 'Price' on entity type 'Book'. This will cause values to be silently truncated if they do not fit in the default precision and scale. Explicitly specify the SQL server column type that can accommodate all the values using 'HasColumnType()'."

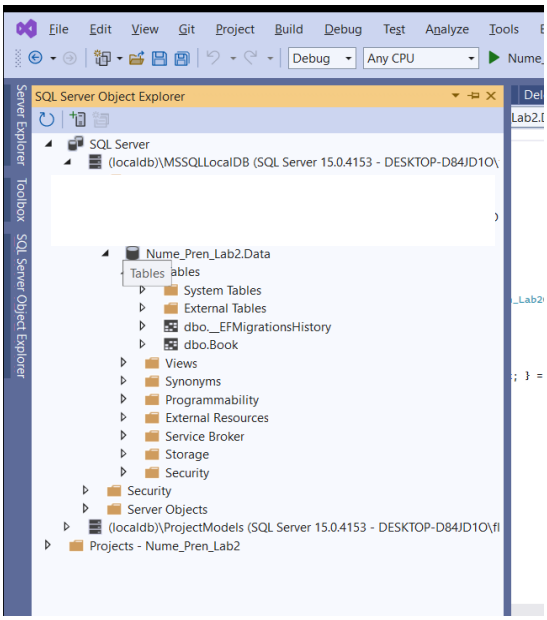
Ignoram momentan warning-ul.

Comenzile vor genera schema bazei de date initiale, bazandu-se pe modelul specificat in clasa Nume_Pren_Lab2Context care mosteneste DbContext si specifica ce entitati sa fie incluse in model

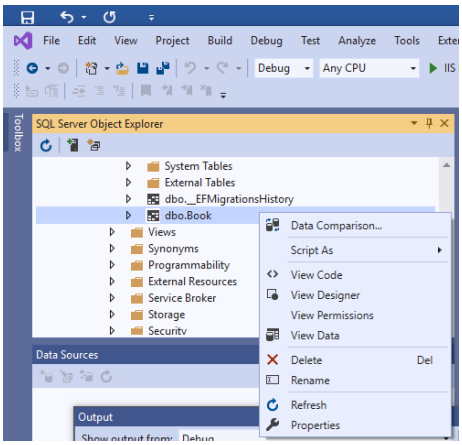


Codul de mai sus creeaza o proprietate DBSet<Book> reprezentand un entity set. In terminologia Entity Framework un entity set corespunde cel mai frecvent unui table din baza de date. O entitate corespunde unui rand din baza de date.

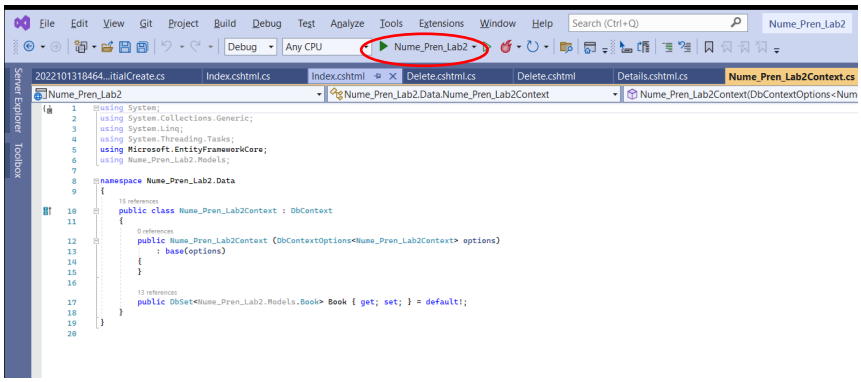
12. Pentru a vedea baza de date creată din meniul **View** deschidem **SQL Server Object Explorer**

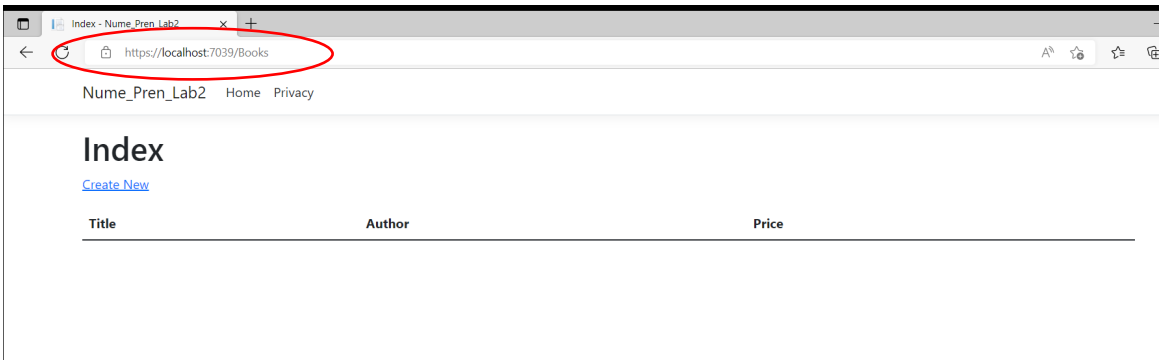


13. Pentru a vedea structura create a tabelului Book, facem click dreapta pe numele tabelului si alegem **View Designer**



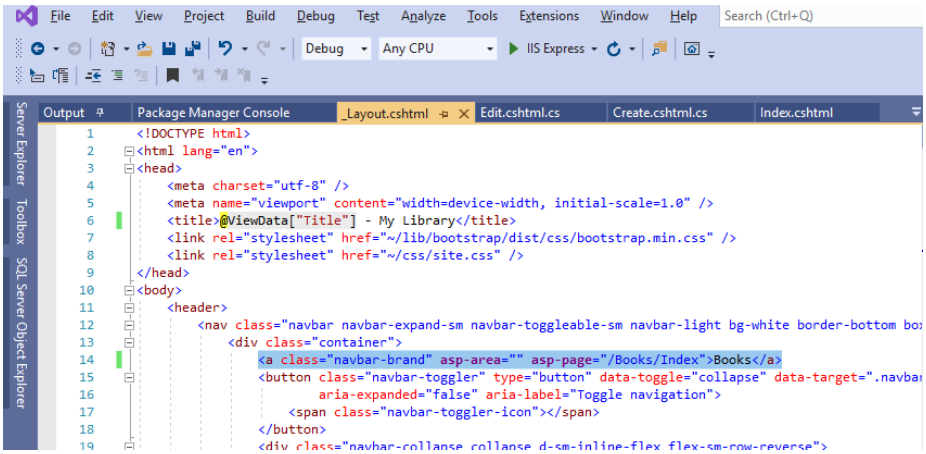
14. Rulam aplicatia, iar in browser adaugam in bara de adrese /Books



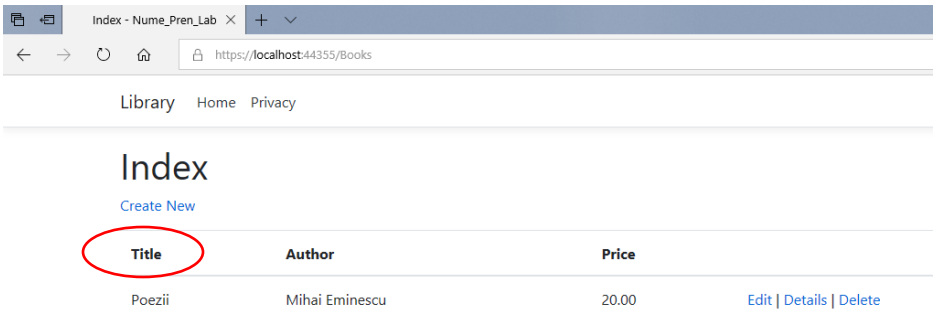


Testam link-ul **Create new** apoi link-urile **Edit**, **Details** si **Delete**.

15. Observam ca in meniu apare numele solutiei Nume_Pren_Lab2. Pentru a afisa un nume mai prietenos vom modifica fisierul _Layout.cshtml conform imaginii de mai jos:



16. In continuare vom vedea cum putem modifica paginile implicit generate. De ex. dorim ca in pagina Books in loc de Title sa apara Book Title



17. Deschidem fisierul Models/Books.cs si adaugam urmatoarele linii:

```
namespace Nume_Pren_Lab2.Models
{
    public class Book
    {
        public int ID { get; set; }

        [Display(Name = "Book Title")]
        public string Title { get; set; }
        public string Author { get; set; }

        [Column(TypeName = "decimal(6, 2)")]
        public decimal Price {get; set; }
    }
```

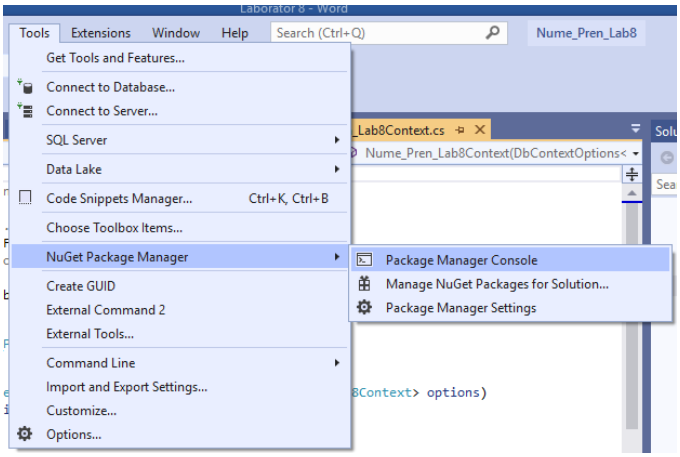
Dataannotations sunt clase atribut care se regasesc in namespace-ul System.ComponentModel.**DataAnnotations** pe care le putem utiliza pentru a seta constrangeri sau valori predefinite.

Adnotarea [Column(TypeName = "decimal(6, 2)")] permite Entity Framework Core sa mapeze corect proprietatea price Price , pemitand valori cu doua zecimale. Atributul Display specifica modul in care dorim sa afisam numele unui camp.

18. In fisierul Models/Book.cs adaugam o noua proprietate pentru clasa Book

```
public class Book
    {
        public int ID { get; set; }

        [Display(Name = "Book Title")]
        public string Title { get; set; }

        public string Author { get; set; }

        [Column(TypeName = "decimal(6, 2)")]
        public decimal Price { get; set; }

        [DataType(DataType.Date)]
        public DateTime PublishingDate { get; set; }
     }
```

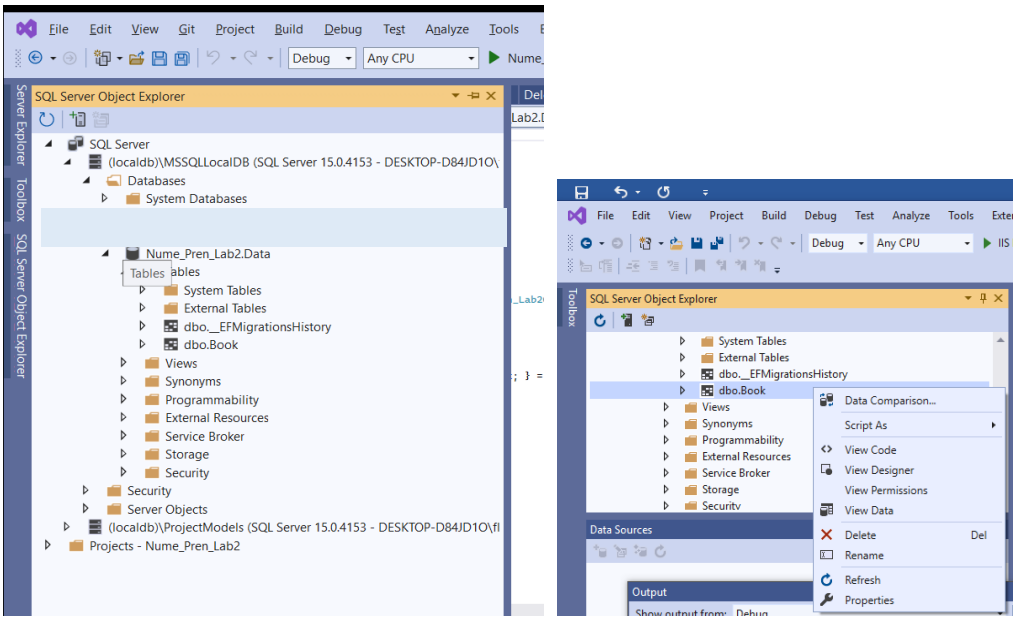
19. Vom actualiza baza de date pe baza modelului actualizat anterior, utilizand optiunea de migrare din Package Manager Console(PMC). Astfel selectam din meniul **Tools**, optiunea **NuGet Package Manager ->Package Manager Console**



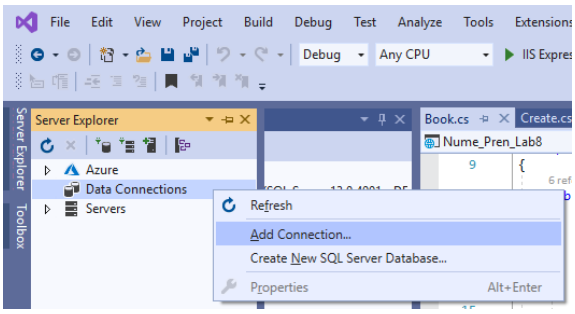
20. In PMC introducem urmatoarele instructiuni

```
Add-Migration PublishingDate
Update-Database
```

21. Observam ca baza de date a fost actualizata, selectand din meniul **View** deschidem **SQL Server Object Explorer**, apoi facem click dreapta pe numele tabelului si alegem **View Designer**



(Sau din ferestra Server Explorer – creem o noua conexiunea alegand fisierul cu extensia .mdf creat in urma migrarii (numele acestuia este : Nume_Pren_Lab2Context……mdf) . Atunci cand se executa migrarea fisierul este creat implicit in C:/Users/User(numele userului cu care va logati pe Windows)



22. Actualizam paginile astfel incat sa fie prezenta si data publicarii (proprietatea PublishingDate)
In pagina Pages/Books/Index.cshtml adaugam :

```
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].Author)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].Price)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].PublishingDate)
            </th>
    …
@foreach (var item in Model.Book) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Author)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Price)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PublishingDate)
            </td>
```

23. In paginile Pages/Books/Create.cshtml si Pages/Books/Edit.cshtml adaugam :

```
    <div class="form-group">
                <label asp-for="Book.Price" class="control-label"></label>
                <input asp-for="Book.Price" class="form-control" />
                <span asp-validation-for="Book.Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Book.PublishingDate" class="control-label"></label>
                <input asp-for="Book.PublishingDate" class="form-control" />
                <span asp-validation-for="Book.PublishingDate" class="text-
    danger"></span>
```

```
    </div>
```

24. In pagina Pages/Books/Details.cshtml adaugam :

```
    <dt class="col-sm-2">
                @Html.DisplayNameFor(model => model.Book.Price)
            </dt>
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.Book.Price)
            </dd>
            <dt class="col-sm-2">
                @Html.DisplayNameFor(model => model.Book.PublishingDate)
            </dt>
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.Book.PublishingDate)
            </dd>
```

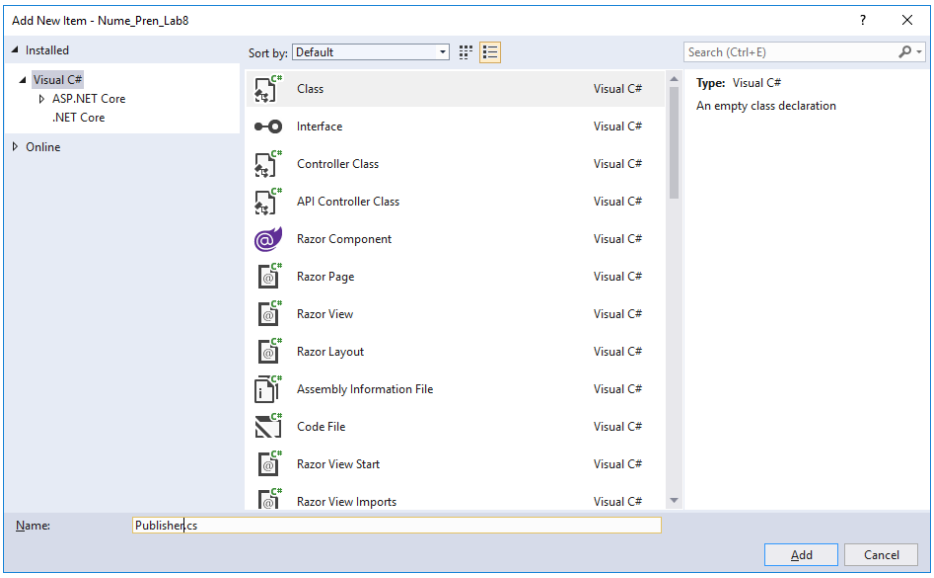
25. In pagina Pages/Books/Delete.cshtml adaugam :

```
    <dt class="col-sm-2">
                @Html.DisplayNameFor(model => model.Book.Price)
            </dt>
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.Book.Price)
            </dd>
            <dt class="col-sm-2">
                @Html.DisplayNameFor(model => model.Book.PublishingDate)
            </dt>
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.Book.PublishingDate)
            </dd>
```

26. Rulam aplicatia si testam toate functionalitatile cu noua proprietate adaugata – PublishingDate.
27. In continuare vom creste complexitatea modelului. Adaugam o noua entitate Publisher precum si o noua proprietate in clasa Book

```
public class Book
    {
        public int ID { get; set; }

        [Display(Name = "Book Title")]
        public string Title { get; set; }

        public string Author { get; set; }

        [Column(TypeName = "decimal(6, 2)")]
        public decimal Price { get; set; }


        public DateTime PublishingDate { get; set; }

        public int? PublisherID { get; set; }

       public   Publisher? Publisher { get; set; }  } //navigation property
```

28. Pentru a creea o noua entitate, in ferestra Solution Explorer, facem click dreapta pe folderul Models->Add->New Item si alegem Class. Denumim noua entiate Publisher



29. In clasa creata adaugam urmatoarele proprietati:

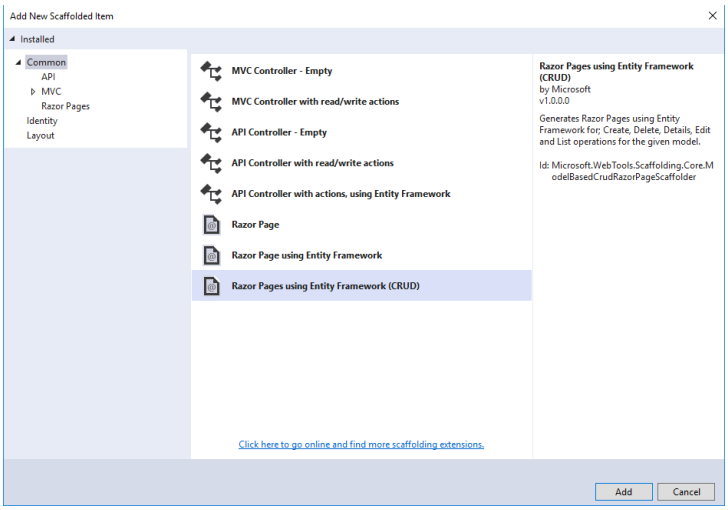
```
public class Publisher
    {
        public int ID { get; set; }

        public string PublisherName { get; set; }

        public ICollection<Book>? Books { get; set; }  //navigation property
      }
```

Cheia primara ID va fi cheie straina pentru entitatea Book. Entity framework identifica automat cheia primara daca acesta poarta denumirea ID sau numeclasaID. In cazul in care dorim ca numele cheii primare sa fie altul atunci utilizam atributul [Key] pentru a indentifica proprietatea respectiva ca fiind cheie primara.

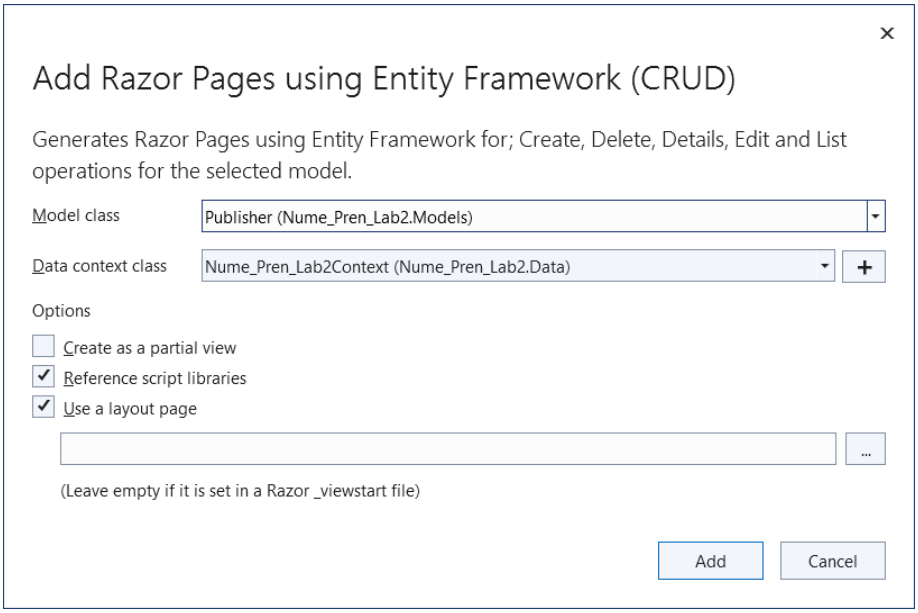
Cand o entitate Book are o entiate relationata Publisher, fiecare va avea o referinta catre cealalta in navigation property

30. Realizati operatia de migrare (vezi pct. 20)
31. In ferestra **Solution Explorer**, adaugam un subdirector la directorul **Pages** apasand click dreapta pe numele directorului Pages, selectam **Add ->New Folder.** Denumim noul subdirector **Publishers**

32. Facem click dreapta pe subdirectorul **Pages/Publishers** si selectam **Add > New Scaffolded Item** si selectam **Razor Pages using Entity Framework(CRUD)**



33. In fereastra urmatoare selectam din lista derulanta de la **Model Class** – Publisher(Nume_Pren_Lab2.Models). La **Data context class** alegem clasa context creata anterior Nume_Pren_Lab2**.Data.**Nume_Pren_Lab2Context



34. Modificam paginile CRUD din folderul Books astfel incat sa apara numele editurii pentru fiecare entiate Book. Astfel in Index.cshtml adaugam:

```
    <th>
                @Html.DisplayNameFor(model => model.Book[0].PublishingDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].Publisher.PublisherName)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
```

```
@foreach (var item in Model.Book) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Author)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Price)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PublishingDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Publisher.PublisherName)
            </td>
            <td>
                <a asp-page="./Edit" asp-route-id="@item.ID">Edit</a> |
                <a asp-page="./Details" asp-route-id="@item.ID">Details</a> |
                <a asp-page="./Delete" asp-route-id="@item.ID">Delete</a>
            </td>
```

35. In Index.cshtml.cs modificam interogarea LINQ din metoda OnGetAsync adaugand:

```
public async Task OnGetAsync()
    {
      Book = await _context.Book
        .Include(b=>b.Publisher)
        .ToListAsync();
    }
```

36. In folderul Books, in fisierele Create.cshtml si Edit.cshtml adaugam un element label si o lista al carei continut il setam folosind tag-helper-ul asp-items si dictionarul ViewBag:

```
<div class="form-group">
          <label asp-for="Book.PublishingDate" class="control-label"></label>
          <input asp-for="Book.PublishingDate" class="form-control" />
          <span asp-validation-for="Book.PublishingDate" class="text-danger"></span>
        </div>
<div class="form-group">
        <label asp-for="Book.PublisherID" class="control-label">Publisher Name</label>
        <select asp-for="Book.PublisherID" class="form-control" asp-items="ViewBag.PublisherID">
</select>
        <span asp-validation-for="Book.PublisherID" class="text-danger"></span>
</div>
        <div class="form-group">
          <input type="submit" value="Create" class="btn btn-primary" />
        </div>
      </form>
```

37. In fisierul Create.cshtml.cs umplem dictionarul ViewData pentru cheia PublisherID utilizand Dbset-ul Publisher:

```
public IActionResult OnGet()
    {
      ViewData["PublisherID"] = new SelectList(_context.Set<Publisher>(), "ID",
"PublisherName");
      return Page();
    }
```

38. Umplem dictionarul ViewData si in fisierul Edit.cshtml.cs:

```
public async Task<IActionResult> OnGetAsync(int? id)
    {
      if (id == null)
      {
        return NotFound();
      }
      Book = await _context.Book.FirstOrDefaultAsync(m => m.ID == id);

      if (Book == null)
      {
        return NotFound();
      }
      Book=book;
      ViewData["PublisherID"] = new SelectList(_context.Set<Publisher>(), "ID",
"PublisherName");
      return Page();
    }
```

39. Sarcina laborator:
    - Actualizati fisierul Layout astfel incat sa se poata naviga spre sectiunea Publishers din bara de navigare (vezi pct. 15). Rulati aplicatia si adaugati minim 3 edituri (Publishers) (Ex. Humanitas, Nemira, Arthur etc.) si 2 carti.
    - Modificati Entity data model al aplicatiei astfel incat sa existe o clasa Authors cu proprietatile ID, FirstName, LastName. Modificati clasa Books astfel: stergeti proprietatea Auhors, adaugati o cheie straina si navigation property catre Authors.
    - Realizati operatia de migrare astfel incat modelul modificat sa corespunda cu structura bazei de date
    - Actualizati fisierul Layout astfel incat sa se poata naviga spre sectiunea Authors din bara de navigare

- In folderul Books, actualizati pagina Index astfel incat autorul sa poata fi ales dintr-un dropdownlist. Actualizati paginile Create, Edit si Delete astfel incat sa afiseze numele autorului utilizand entitatea Authors nou creata.