ITR UTEC
Centro Sur
Universidad Tecnológica

TECNÓLOGO
EN INFORMÁTICA

Taller de sistemas de información .NET Tarea – 02_C1 Web App con Razor pages

Autor: Lucas Techera

Docente: Gabriel Aramburu

CI: 5.295.981-3

Contacto: fidel.techera@estudiantes.utec.edu.uy

Consigna Nº 1.

1) Crear un proyecto utilizando la plantilla correspondiente
2) Implementar operaciones CRUD de un catálogo de Libros.
   a) De cada libro se conoce su isbn, título, autor y fecha de publicación.
   b) No implementar la capa de persistencia (solo guardar el catálogo en memoria)
   c) Utilice los conceptos utilizados en clase.
   d) Recordar usar la herramienta de generación de código (Scaffolding)

Solución:

Se comienza creando un proyecto nuevo utilizando la siguiente plantilla: Aplicación web ASP.NET Core (Modelo-Vista-Controlador), se indica nombre y ubicación del proyecto y por último se verifica que el framework sea .NET 8.0.

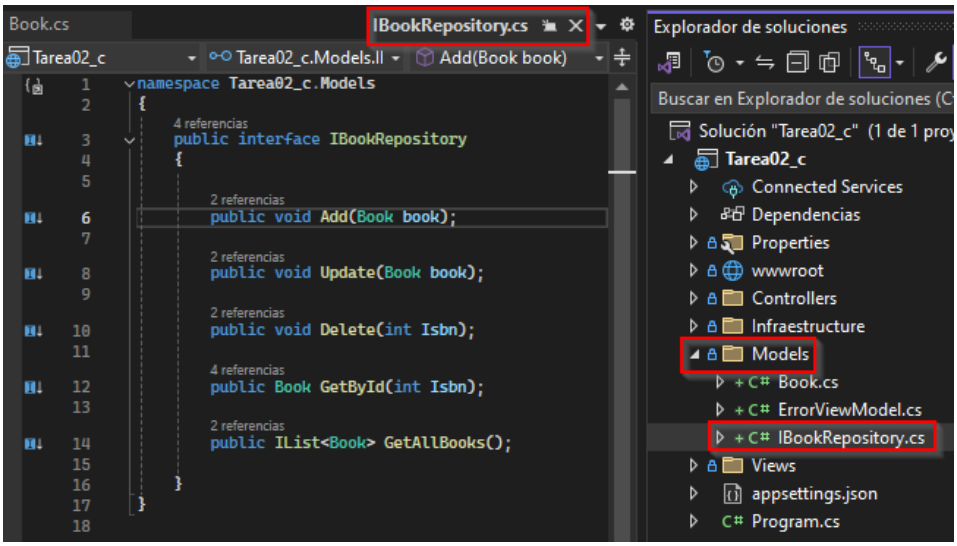
A continuación, dentro de la carpeta Modelos, se creará la clase de dominio libro, la cual contará con los atributos solicitados en la consigna. La misma se verá así:

Book.cs:

```
namespace Tarea02_c.Models
{
    44 referencias
    public class Book
    {
        private int _Isbn;

        25 referencias
        public int Isbn
        {  //propiedad que define setter y getter para mi atributo
            get { return _Isbn; }
            set { _Isbn = value; }
        }


        13 referencias
        public string? Title { get; set;}

        13 referencias
        public string? Author { get; set;}

        11 referencias
        public DateOnly CreationDate {  get; set;}

        0 referencias
        public Book()
        {

        }

        6 referencias
        public Book(int isbn, string title, string author, DateOnly creationDate)
        {
            this._Isbn = isbn;
            this.Title = title;
            this.Author = author;
            this.CreationDate = creationDate;
        }
    }
```

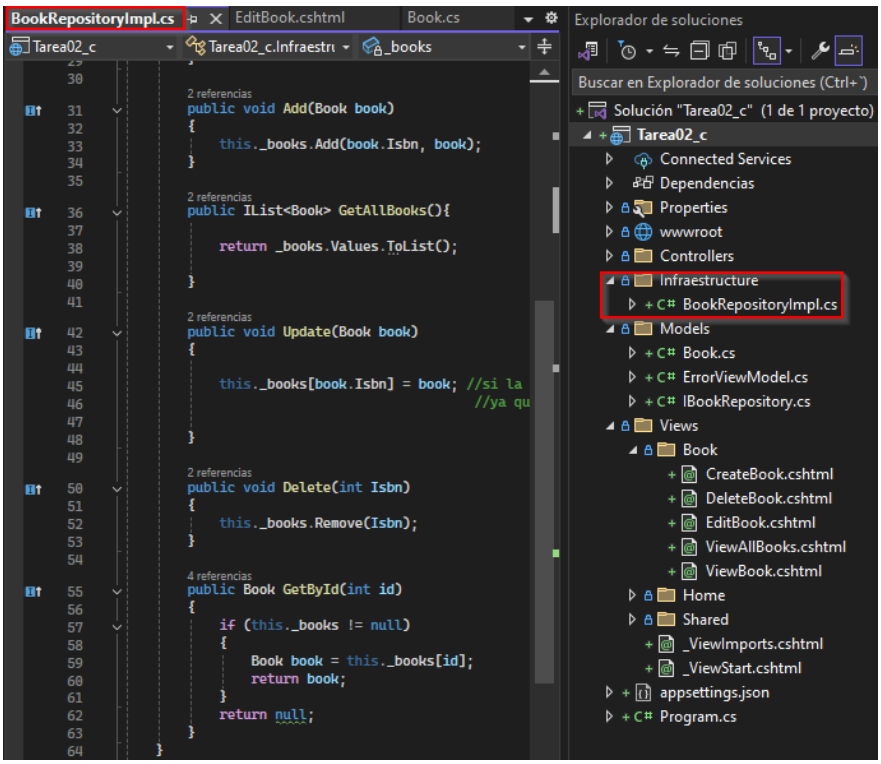
Siguiendo la consigna se realizará una interfaz repositorio de libro y la implementación de la misma con operaciones sencillas para este CRUD. Dicha interfaz se encuentra dentro de la carpeta Models junto con la clase de dominio creada previamente.

IBookRepository.cs:



Para la implementación de esta interfaz se crea una carpeta llamada **Infraestructure**, en la cual pondremos nuestra implementación.

BookRepositoryImpl.cs:

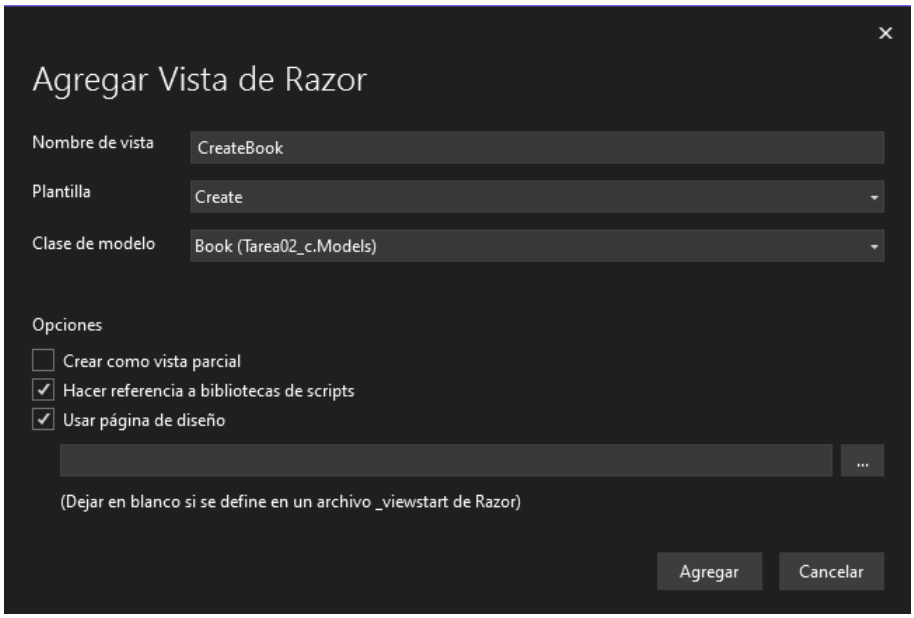


En dicha implementación también se tiene un diccionario, cuya clave será un número entero (isbn) y guardará un objeto libro. A dicho diccionario se agregan varios datos(libros) de prueba, los cuales se encuentran en el constructor de la clase y posteriormente serán inyectados en un controlador, creando los objetos al ejecutar la aplicación.

BookRepositoryImpl.cs:

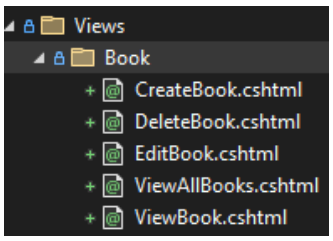
```
public class BookRepositoryImpl : IBookRepository
{

    private IDictionary<int, Book> _books;

    0 referencias
    public BookRepositoryImpl()
    {
        _books = new Dictionary<int, Book>();

        //isbn, titulo, autor, fecha creacion
        Book book1 = new Book(0, "Voces anonimas - Inframundo", "Guillermo Lockhart", new DateOnly(2023, 7, 21));
        Book book2 = new Book(1, "Misterio en el Cabo Polonio", "Helen Velando", new DateOnly(2021, 5, 17));
        Book book3 = new Book(2, "Felipe", "Susana Olaondo", new DateOnly(2019, 1, 23));
        Book book4 = new Book(3, "El poder de seis", "Pitacus Lore", new DateOnly(2016, 6, 12));
        Book book5 = new Book(4, "Soy el numero cuatro", "Pitacus Lore", new DateOnly(2014, 5, 1));
        Book book6 = new Book(5, "Voces anonima - Juegos Prohibidos", "Guillermo Lockhart", new DateOnly(2015, 11, 30));


        this._books.Add(book1.Isbn, book1);
        this._books.Add(book2.Isbn, book2);
        this._books.Add(book3.Isbn, book3);
        this._books.Add(book4.Isbn, book4);
        this._books.Add(book5.Isbn, book5);
        this._books.Add(book6.Isbn, book6);
    }
```

A continuación, se crearán vistas las cuales utilizan la herramienta de generación de código scaffolding, a continuación, se enseña la creación de una de las vistas, al ser un proceso muy similar para la creación de las otras solo se mostrará una (el resto se encontrará en un repositorio en GitHub).



En la imagen se puede observar como se agrega una vista de razor, esto se logra realizando lo siguiente: Dado que nuestro proyecto se creo con una plantilla especifica el mismo ya viene con una carpeta Views(Vistas) en la cual crearemos otra carpeta dentro para guardar todas las vistas relacionadas con nuestra clase libro. A continuación, haremos click derecho en dicha carpeta libro y seleccionaremos en agregar una vista razor, luego, se rellena la información solicitada en la página anterior, un nombre para nuestra vista, la plantilla (uso que se le va a dar) y una clase modelo, la cual es la clase de dominio creada al inicio de este documento.

Después de repetir estos pasos para crear todas las vistas para nuestras operaciones, se obtendrán las siguientes vistas:



En la imagen se pueden observar todas las vistas de nuestras operaciones básicas, ahora debemos editar cada una de ellas y adaptarla a nuestras necesidades, a continuación, se muestra la vista **ViewAllBooks.cshtml.**

```
@model IEnumerable<Tarea02_c.Models.Book>

@{

}

<h1>List of registred books</h1>

<p>
    <a asp-action="CreateBook">Create new book</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Isbn)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Author)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.CreationDate)
            </th>
        </tr>
    </thead>
    <tbody>
```

```
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Isbn)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Author)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.CreationDate)
            </td>
            <td>
                    <a asp-action="EditBook" asp-route-id="@item.Isbn">Edit</a> |
                    <a asp-action="ViewBook" asp-route-id="@item.Isbn">Details</a> |
                    <a asp-action="Delete" asp-route-id="@item.Isbn">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
```

En esta vista se cargarán los objetos que se encuentren en el diccionario creado previamente.

Habiendo realizado una vista para la operación que obtendrá todos los libros, se creará el controlador que expondrá los endpoints y se implementará el mismo, utilizando la vista previamente presentada.

BookController.cs:

```
[Route("Book")]
3 referencias
public class BookController : Controller
{

    private readonly ILogger<BookController> _logger;

    private readonly IBookRepository _bookRepository;

    private readonly IWebHostEnvironment _webRootPath; //en caso de necesitar a

    0 referencias
    public BookController(ILogger<BookController> logger,
      IBookRepository repo,
      IWebHostEnvironment path)
    {
        _logger = logger;
        _bookRepository = repo;
        _webRootPath = path;
    }


    0 referencias
    public IActionResult Index()
    {
        return View();
    }

    [HttpGet]
    [Route("GetAllBooks")]
    0 referencias
    public IActionResult GetAllBooks()
    {
        _logger.LogInformation("Devolviendo lista de libros del sistema.");
        return View("ViewAllBooks", this._bookRepository.GetAllBooks());
    }
```

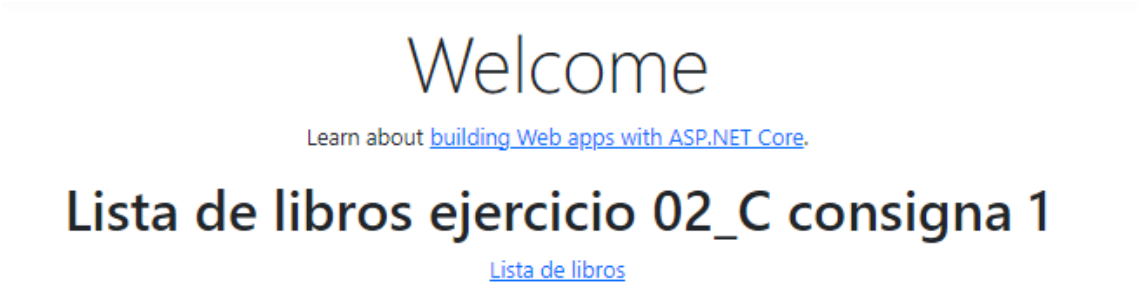
```
Tarea02_c
  Connected Services
  Dependencias
  Properties
  wwwroot
  Controllers
    BookController.cs
    HomeController.cs
  Infraestructure
  Models
  Views
    Book
      CreateBook.cshtml
      DeleteBook.cshtml
      EditBook.cshtml
      ViewAllBooks.cshtml
      ViewBook.cshtml
    Home
    Shared
      _ViewImports.cshtml
      _ViewStart.cshtml
  appsettings.json
  Program.cs
```

Se puede observar en esta imagen el controlador que se utilizará para exponer los endpoints, pero vamos a explicar como se llego a esto. Para empezar en la carpeta Controllers, previamente creada dado la plantilla utilizada, se creará un controlador MVC en blanco.

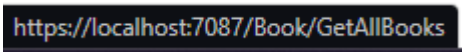
Ahora con nuestro controlador creado procedemos a darle una ruta de acceso general, en este caso “Book”, y a continuación, se realiza la inyección de dependencias de varios componentes entre ellos el Repositorio implementado previamente.

Ahora implementaremos el endpoint de obtener todos los libros, dado que previamente creamos su vista. El mismo es de tipo **Get** y su ruta es **GetAllBooks,** a continuación, dentro de nuestro método se imprime un mensaje el cual se visualizará en la consola utilizando un logger, y se retornará la función View(), dicha función contiene como primer parámetro la vista que se devolverá y como segundo parámetro la función que nos permite obtener los datos de los libros, esto nos devolverá un objeto ViewResult, utilizando el viewName(primer parámetro) y un modelo(Segundo parámetro).

Al ejecutar el proyecto se podrá observar la siguiente vista:



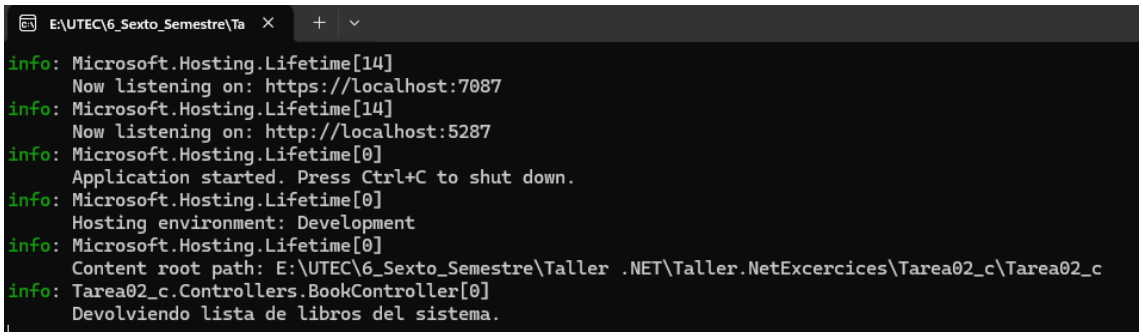
Al pasar el cursor sobre el vínculo Lista de libros en la esquina inferior izquierda se podrá observar la hacia donde nos redirige ese vínculo.



Dicho vínculo contiene la ruta general y la ruta del endpoint vistos previamente.

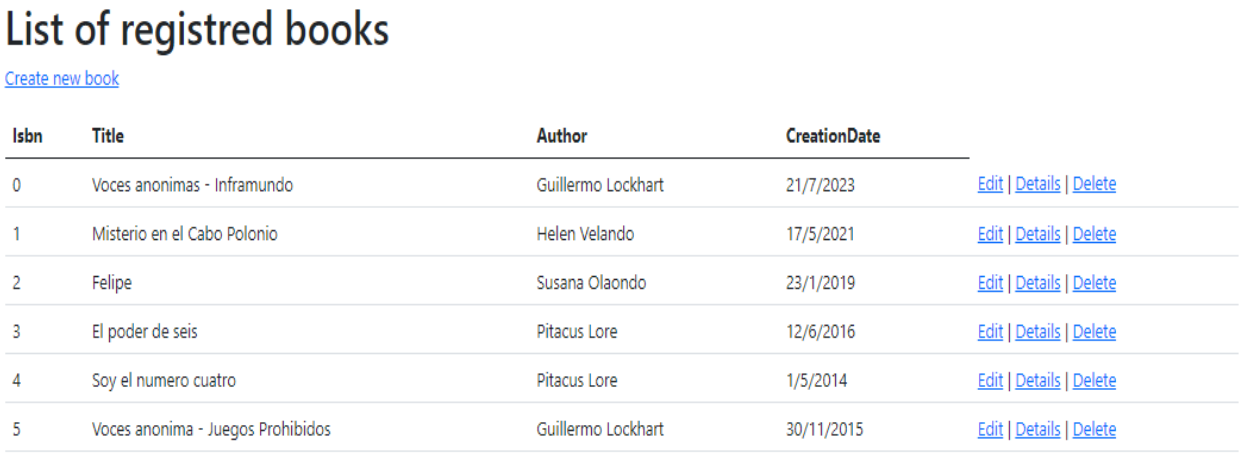
Al presionar dicho vínculo se accede a la lista de libros, por lo tanto, ejecutando el endpoint.

Ejecución:



En la consola se puede observar el mensaje impreso por el logger, al revisar el navegador nos encontraremos con la siguiente vista:

Ejecución ViewAllBooks.cshtml:



Se puede observar la vista creada utilizando los datos de prueba creados previamente dichos datos son recuperados por la función **GetAllBooks(),** implementada en el repositorio.

Con esto queda demostrado el correcto funcionamiento del endpoint que nos permite obtener todos los libros del sistema.

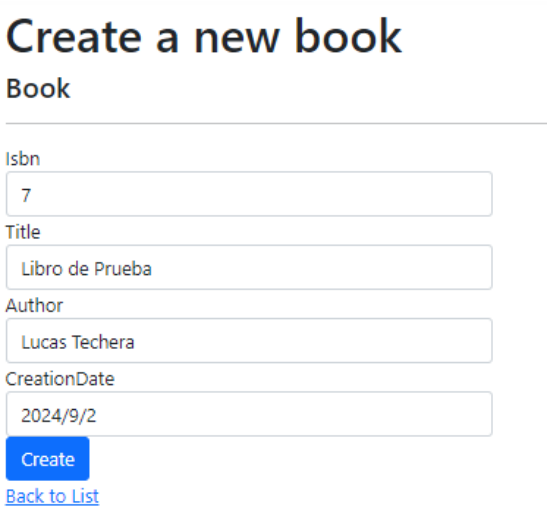
A continuación, se explicará y demostrará el funcionamiento de las operaciones de creación, actualización y eliminación.

Operaciones para crear libros:

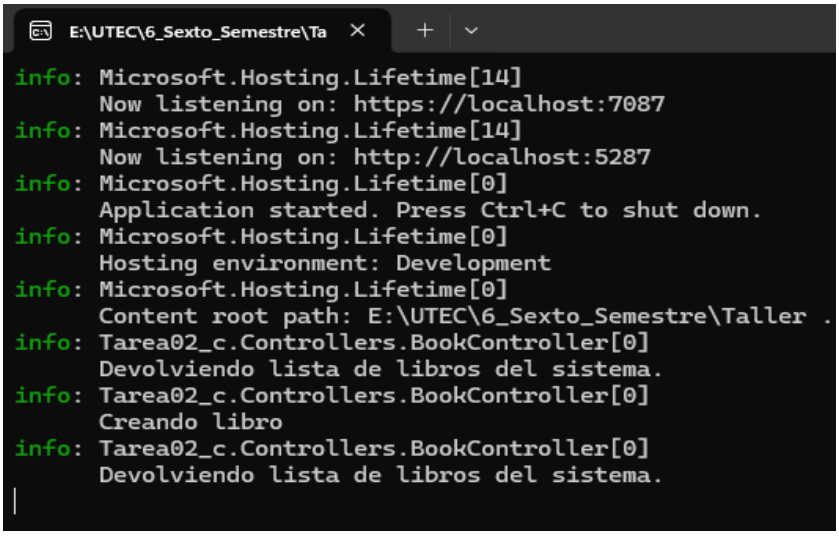
```
[HttpGet]
[Route("CreateBook")]
0 referencias
public IActionResult CreateBook()
{
    return View(); //cuando devuelvo esto el controllador asume que quiero devolver una
}

[HttpPost]
[Route("CreateBook")] //puedo tener dos endpoints con mismo nombre siempre y cuando uno
0 referencias
public IActionResult CreateBook([Bind("Isbn,Title,Author,CreationDate")] Book book) //
{

    _logger.LogInformation("Creando libro");
    this._bookRepository.Add(book);
    return RedirectToAction("GetAllBooks", "Book");
}
```

En esta imagen se puede apreciar dos endpoints de igual nombre, la diferencia radica en su tipo, uno es GET y el otro es POST, y en la cantidad de parámetros recibidos. El primer endpoint se encarga de cargar la vista de un formulario en el navegador, mientras que el segundo endpoint se encarga de enviar los datos del formulario de la vista a la función de **Add(),** posteriormente se nos redirecciona a la vista que contiene todos los libros donde se puede observar el libro agregado.

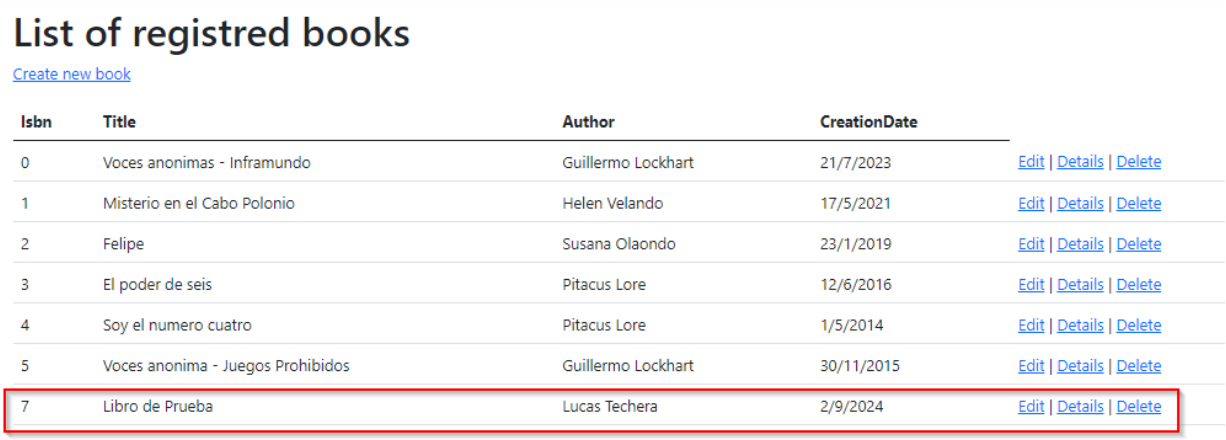
Ejecución:



Vista del formulario para crear un libro.



Al presionar el botón Create, se ejecuta el endpoint y se puede observar en la consola el mensaje de creación del libro, y a continuación se puede visualizar el mismo:



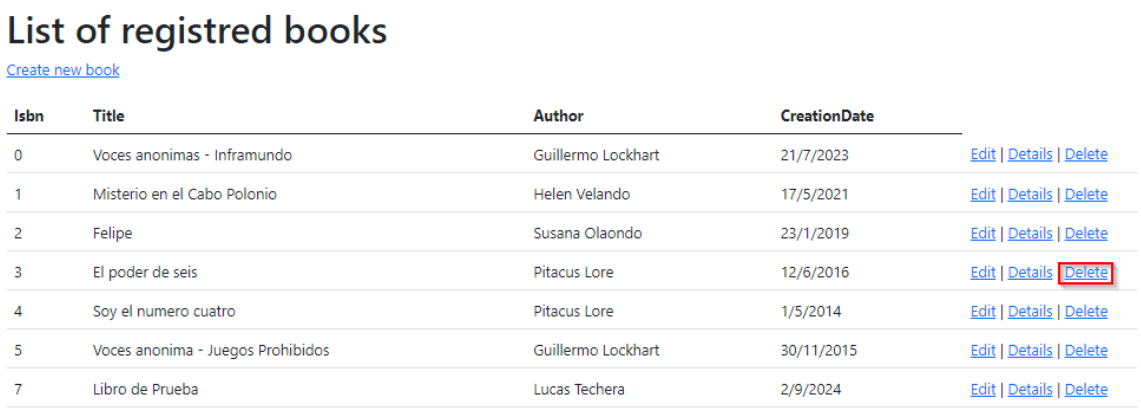
Al presionar el vínculo Details se puede acceder a la vista de obtener libro por Id y visualizarlo en detalle.

Los siguientes endpoints tienen un funcionamiento similar, un endpoint se encargará de cargar una vista y el segundo ejecutará la acción, por lo tanto, se omitirá una explicación y se mostrará su ejecución.

Operaciones eliminar libros:

```
[HttpGet]
[Route("Delete/{Isbn}")]
0 referencias
public IActionResult Delete(int Isbn)
{
    if (Isbn != null)
    {
        _logger.LogInformation("Estoy mostrando libro a eliminar Isbn: " + Isbn);

        Book book = _bookRepository.GetById(Isbn);

        return View("DeleteBook", book);
    }

    return null;

}

[HttpPost]
[Route("DeleteConfirmed")]
0 referencias
public IActionResult DeleteConfirmed(int Isbn)
{
    _logger.LogInformation("Eliminando libro ....");
    this._bookRepository.Delete(Isbn);

    return RedirectToAction("GetAllBooks", "Book");

}
```

Lista de libros registrados:



En la imagen se observa el vínculo, que redireccionará a la vista de eliminar libro.

Delete Book

Are you sure you want to delete this?

Book

Isbn	3
Title	El poder de seis
Author	Pitacus Lore

Delete | Back to List

A continuación, se presiona el botón y se vuelve a la lista de libros.

List of registred books

Create new book

Isbn	Title	Author	CreationDate	
0	Voces anonimas - Inframundo	Guillermo Lockhart	21/7/2023	Edit | Details | Delete
1	Misterio en el Cabo Polonio	Helen Velando	17/5/2021	Edit | Details | Delete
2	Felipe	Susana Olaondo	23/1/2019	Edit | Details | Delete
4	Soy el numero cuatro	Pitacus Lore	1/5/2014	Edit | Details | Delete
5	Voces anonima - Juegos Prohibidos	Guillermo Lockhart	30/11/2015	Edit | Details | Delete
7	Libro de Prueba	Lucas Techera	2/9/2024	Edit | Details | Delete

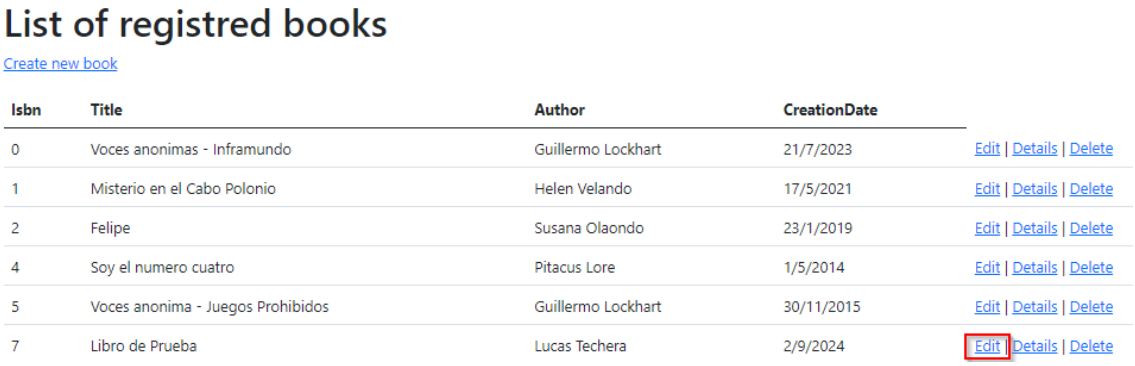
Se observa la ausencia del libro con Isbn 3. Se corrobora en la consola el funcionamiento utilizando la información impresa por el logger.

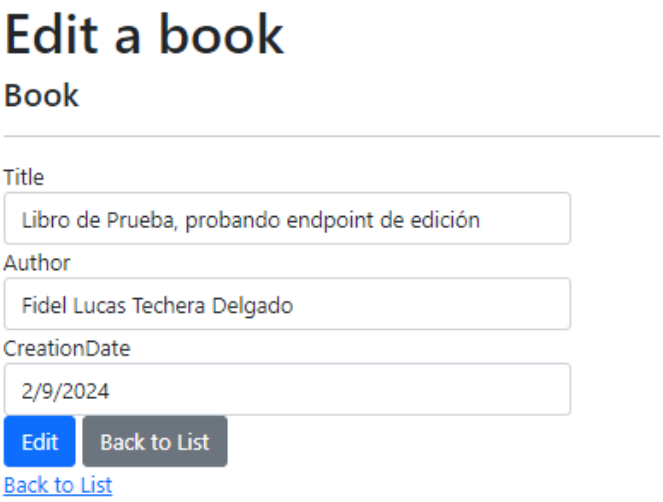
```
E:\UTEC\6_Sexto_Semestre\Ta
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7087
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5287
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: E:\UTEC\6_Sexto_Semestre\Taller .NET\Taller.NetExcercices\Tarea02_c\Tarea02_c
info: Tarea02_c.Controllers.BookController[0]
      Devolviendo lista de libros del sistema.
info: Tarea02_c.Controllers.BookController[0]
      Creando libro
info: Tarea02_c.Controllers.BookController[0]
      Devolviendo lista de libros del sistema.
info: Tarea02_c.Controllers.BookController[0]
      Estoy devolviendo un libro con id: 7
info: Tarea02_c.Controllers.BookController[0]
      Devolviendo lista de libros del sistema.
info: Tarea02_c.Controllers.BookController[0]
      Estoy mostrando libro a eliminar Isbn: 3
info: Tarea02_c.Controllers.BookController[0]
      Eliminando libro ....
info: Tarea02_c.Controllers.BookController[0]
      Devolviendo lista de libros del sistema.
```

Se observa la cadena de sucesos: Mostrar libro, eliminar y devolver a la vista con la lista de todos los libros.

Operaciones editar libros:

```
[HttpGet]
[Route("EditBook/{Isbn}")]
0 referencias
public IActionResult Edit (int Isbn)
{

    _logger.LogInformation("Estoy mostrando libro a editar Isbn: " + Isbn);

    Book book = this._bookRepository.GetById(Isbn);

    return View("EditBook", book);
}

[HttpPost]
[Route("EditConfirmed")]
0 referencias
public IActionResult EditConfirmed([Bind("Isbn,Title,Author,CreationDate")] Book book)
{
    _logger.LogInformation("Editando libro ....");
    this._bookRepository.Update(book);

    return RedirectToAction("GetAllBooks", "Book");

}
```





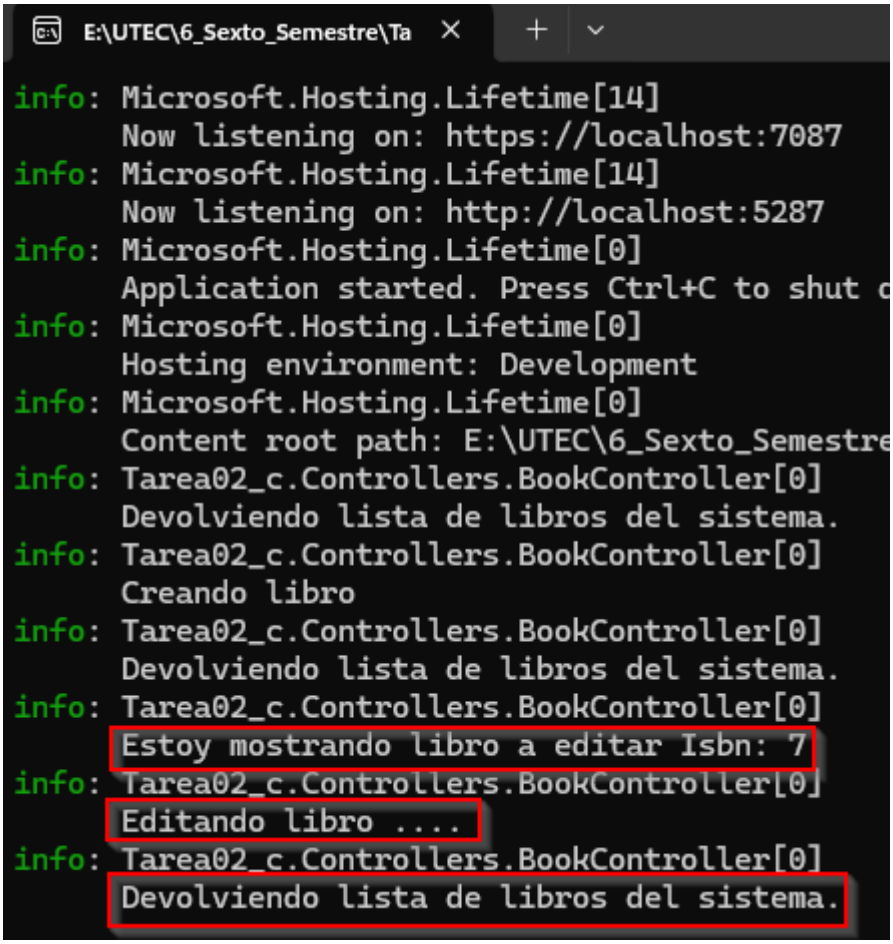
Al igual que en el caso anterior se debe presionar el botón de editar para ejecutar el endpoint.

List of registred books

Create new book

Isbn	Title	Author	CreationDate	
0	Voces anonimas - Inframundo	Guillermo Lockhart	21/7/2023	Edit | Details | Delete
1	Misterio en el Cabo Polonio	Helen Velando	17/5/2021	Edit | Details | Delete
2	Felipe	Susana Olaondo	23/1/2019	Edit | Details | Delete
3	El poder de seis	Pitacus Lore	12/6/2016	Edit | Details | Delete
4	Soy el numero cuatro	Pitacus Lore	1/5/2014	Edit | Details | Delete
5	Voces anonima - Juegos Prohibidos	Guillermo Lockhart	30/11/2015	Edit | Details | Delete
7	Libro de Prueba, probando endpoint de edición	Fidel Lucas Techera Delgado	2/9/2024	Edit | Details | Delete

Se comprueba en el navegador que se realizó la edición, se realiza una última comprobación en la consola.



En la imagen se puede apreciar el orden en que se ejecutaron los endpoints, para poder editar un libro.

A continuación, se adjunta link hacia el repositorio de GitHub en el cual se encuentra el proyecto, el cual incluye el código del resto de las vistas y comentarios de curiosidades acerca de la plataforma .NET durante el desarrollo del ejercicio.

GitHub: https://github.com/ElPiche/.NET_Excercices/tree/main/Tarea02_c