**Proyecto Sprint III – CODIGO**

**ENTREGABLE 26 SEPTIEMBRE**

**TEAM LEADER ALEJANDRO DE LA ESPRIELLA MUÑOZ**

Razor es el tipo de plantilla para vistas que maneja BLAZOR y cómo funciona incrustando HTML con C#

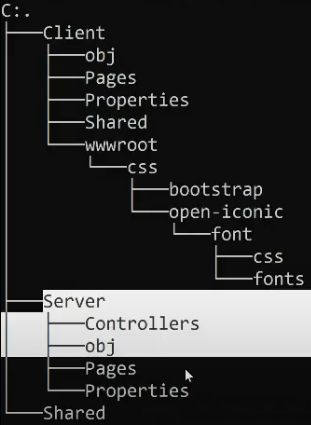
HTML es para generar una estructura de una pagina HTML y CSS no es un lenguaje de programación

Crear 3 carpetas del **proyecto SPRINT III** (Client Server Shared)

**Client** Programación que ve el usuario final

**Server** programación de la lógica del sistema

**Shared** Capa que relaciona el servidor con el cliente



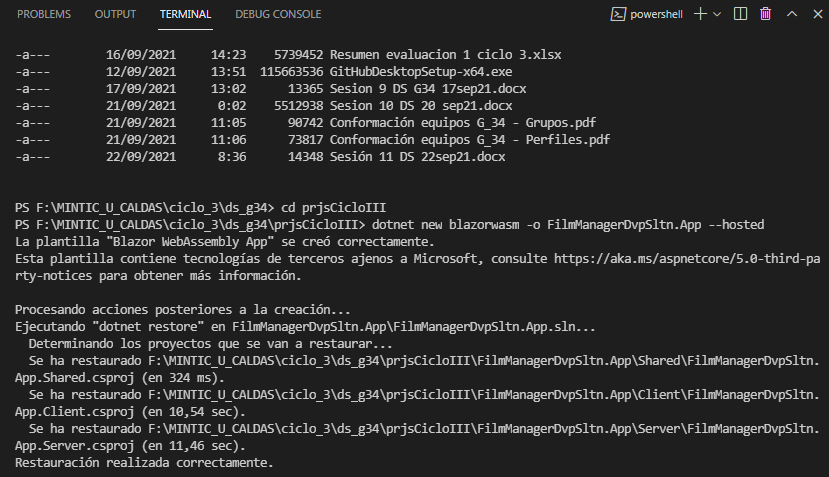
**NOMBRE DEL PROYECTO** FilmManagerDvpSltn

Elegir ubicación del proyecto

Ejecutar siguiente instrucción para crear la carpeta del proyecto

- [ ] 1. dotnet new blazorwasm -o FilmManagerDvpSltn.App --hosted

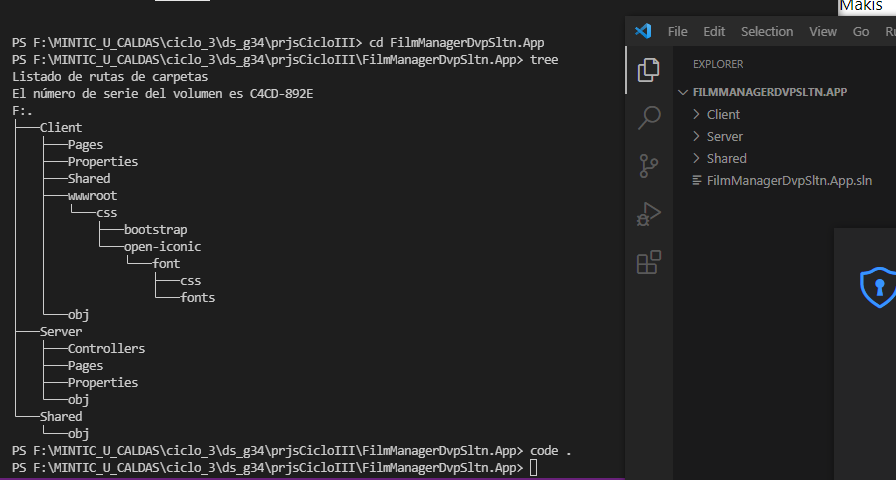
(VSC menú FILE)



Revisar la creación de las subcarpetas **CLIENT SERVER SHARED**

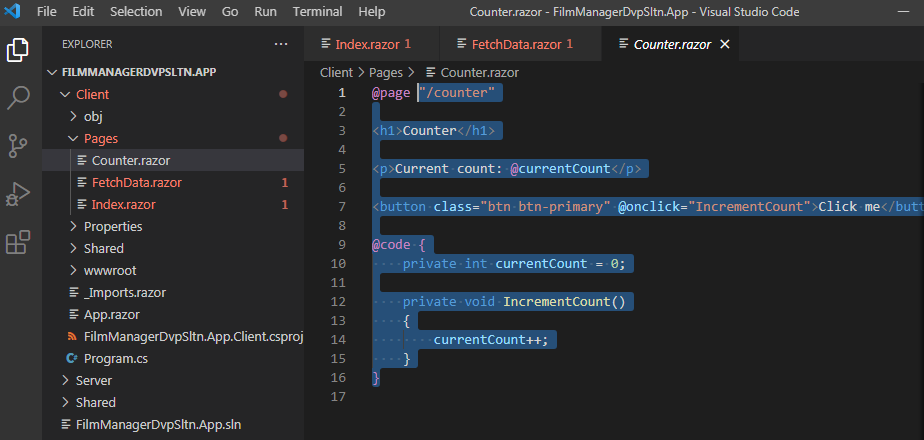
-[ ] 2. Ejecutar comando **TREE** para ver la estructura del proyecto e Iniciar el proyecto en VSC

-[ ] 3. Ejecutar en terminal VSC el comando CODE .



-[ ] 4. Borrar el contenido de PAGES, de los archivos Index.razor

FetchData.razor y Counter.razor que se creó automáticamente con blazorwasm



Documentacion .NET

<https://docs.microsoft.com/en-us/dotnet/>

Estructura del proyecto Razor

<https://docs.microsoft.com/es-es/aspnet/core/blazor/project-structure?view=aspnetcore-5.0>

Servicio – consume la información de una API

API - application programing interfaces es un conjunto de subrutinas, funciones y procedimientos que permiten realizar alguna acción o acceder a alguna característica o contenido que de un sitio

Json JavaScript Object Notation tipo de estructura de texto plano que se usa para el intercambio de datos

En una página web la información se tiene que estar actualizando periódicamente. Para ir guardando la información que los usuarios del sitio web ingresan en el tiempo, durante sus interacciones con este.

**Servicio** activo permanentemente, API,(interfaz) que toma la información que los usuarios ingresan y posteriormente la haga disponible en el sito web.

**@Page** Directiva que va aindicar cual va a ser el cambio en la URI, y ese cambio se refleja en el componente SHARED y el archivo NAVMENU.RAZOR

<https://newsapi.org/>

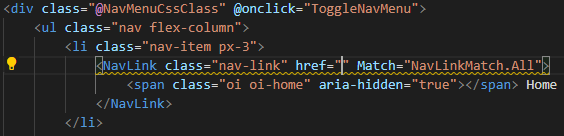
<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/uri-identificador-de-recursos-uniformes/>

URI Identificador de recurso uniforme, representación compacta de un recurso disponible para su aplicación en una intranet o en internet. La clase URI define las propiedades y métodos para manejar URIs sirve para acceder a un **recurso físico o abstracto por Internet.** Las aplicaciones utilizan este identificador único para interactuar con el recurso o consultar información sobre el mismo.

URL se utiliza para indicar dónde se encuentra un recurso. dirección para acceder a las paginas por internet

 URN [***uniform resource name***](t3://page?uid=2376) es independiente de la ubicación y designa un recurso de forma permanente. Por lo tanto, si el URL se conoce principalmente como una forma de identificar un dominio web, el URN también puede tratarse de un ISBN que identifique un libro de manera indefinida.

NavMenu.razor Es un Menu que tiene enlaces <LI> que redireccionan a otra parte de la pagina, u bicados en una <UL> columna de navegación

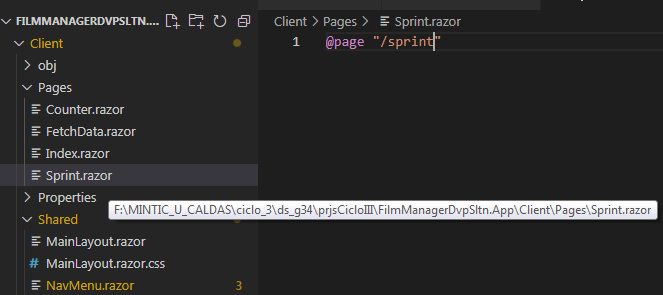


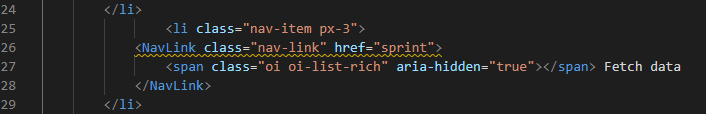
href ubicación del proyecto

Agregar vista page SPRINT a NavMenu

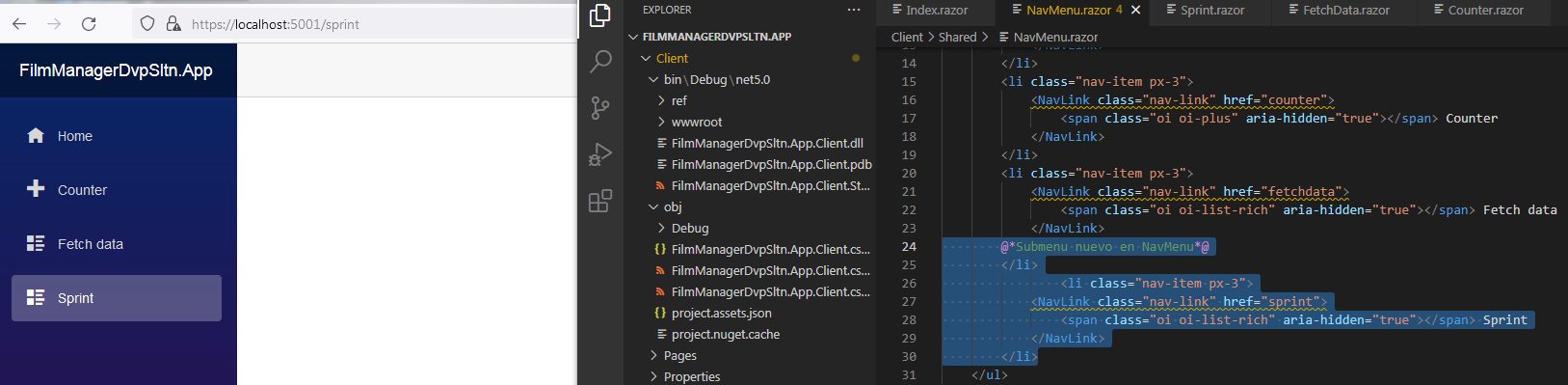
-[ ] 4. Crear Archivo Sprint.razor en Client\Pages

Crear acceso al archivo Sprint.razor en NavMenu





**Comprobar vista**



Crear componentes(estructura html) – reutilizando código

<https://getbootstrap.com/docs/5.1/components/card/>

Content y Forms

**A PARTIR DE AQUÍ**

**DESARROLLO DE REQUERIMIENTOS SPRINT III**

Crear una carpeta con nombre Services en el proyecto de Client

Carpeta: Services

Interfaz: IServiceFilm.cs

Clase: ServiceFilm.cs

Ir a la carpeta Shared del Proyecto (FilmManagerDvpSltn.App\Shared\)

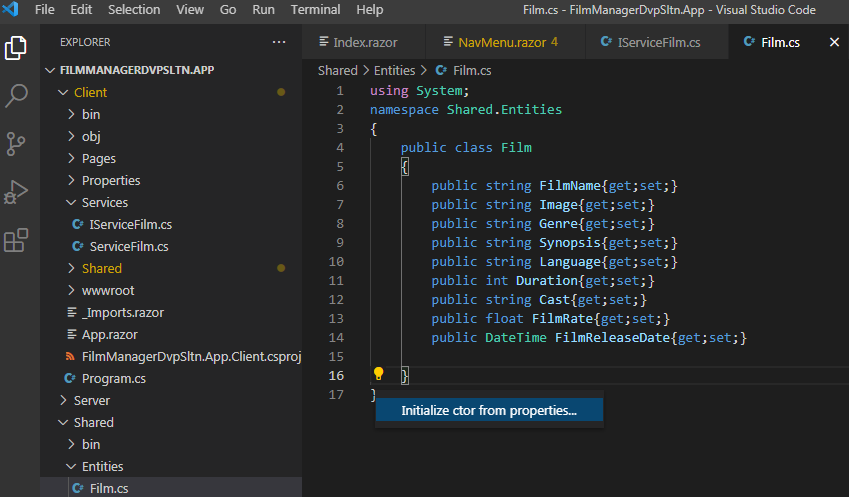
Crear carpeta Entitities

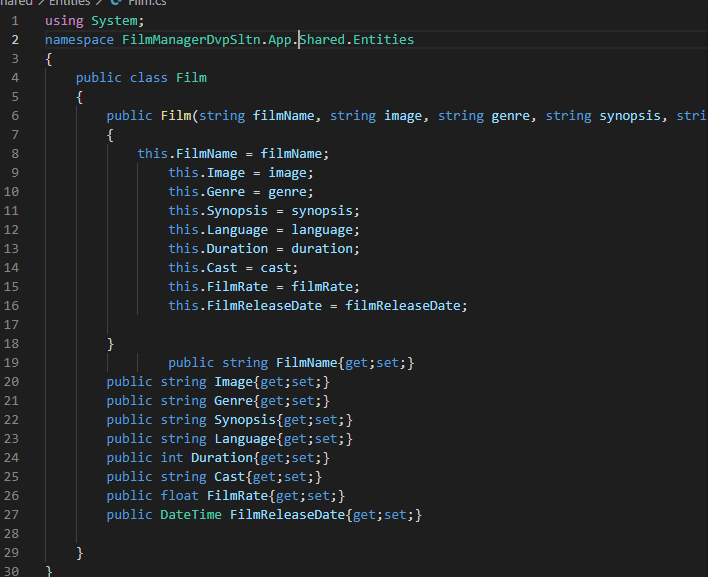
Shared es la capa intermedia que nos conecta con el server y el server con la base de datos

Crear Clase Film en Entities

Crear atributos de las películas FilmName Image Genre Synopsis Language Duration FilmReleaseDate

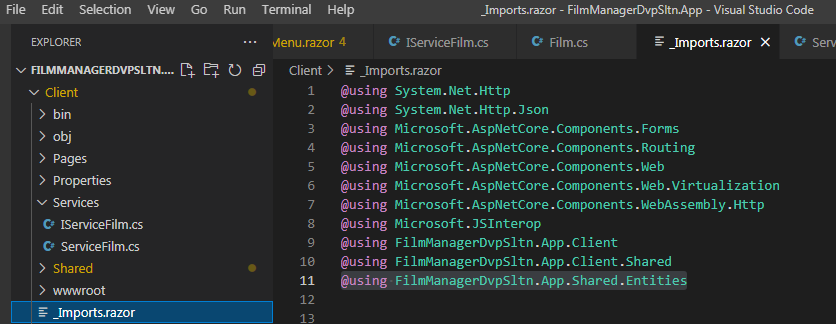
Un atributo **Image** cuando se le agrega **{get;set;}** se convierte en una propiedad, porque va a tener método para obtener un valor y devolver un valor.

Método constructor Initialize ctor from properties, C# lo realiza implícitamente, por lo cual no es necesario hacerlo, por practica de aprendizaje y para ver la importancia de la nemotecnia, la forma en que se declaran las variables o atributos en Mayúscula, se hace la practica



Para llamar el paquete Film.cs en Entities en Client\Shared lo debo importar (crtl+p) , en el archivo \_imports.razor

Dentro de \_imports.razor, se debe hacer público con @using FilmManagerDvpSltn.App.Shared.Entities



Así la carpeta Entities se hace visible para el Client (ubicación \FilmManagerDvpSltn.App\Client).

Y cuando en la interface del servicio IServiceFilm se crea una List<Film>, esta Lista esta llamando a la clase Film que se encuentra en\FilmManagerDvpSltn.App\Shared\Entities, la cual hicimos visible.

Y en el enunciado de \Client\Services\IServiceFilm.cs se inserta la función

using FilmManagerDvpSltn.App.Shared.Entities

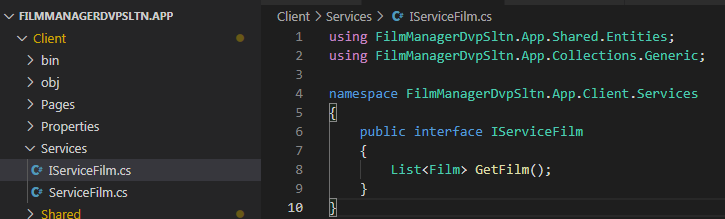
La Lista también debe importar una colección genérica con

using FilmManagerDvpSltn.App.Collections.Generic;

se llama el método GetFilm(){} pero no se implementa ({… }vacío) por ser una Interfaz que no lo permite.

Se implementa en la clase ServiceFilm.cs

Interfaz dice: genere todos los métodos que va a tener abstractos. Pero no los puede implementar



La interfaz nos permitirá crear un API (Interfaz de Programación de Aplicaciones) las clases, lo cual hará que las clases que implementen la interfaz incluyan los métodos que contiene.

Listas que puedo desplegar agregándolas a la Interface(todas las que quiera)

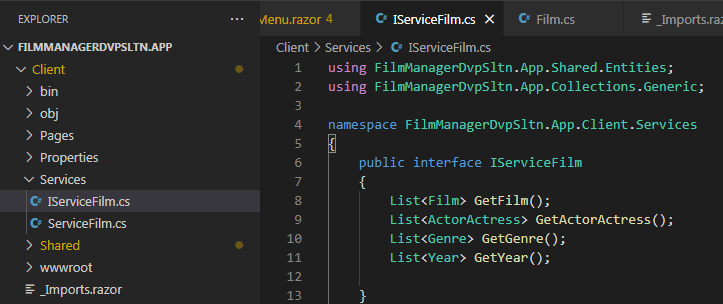
List<Film>

List<ActorActress>

List<Genre>

List<Year>

Estos componentes(Listas )luego se deben implementar en Shared\Entities\



Los servicios en \IServiceFilm.cs se pueden crear tanto en \Client como en \Server,

La carpeta de Entities también se puede crear en el \Server, como en \Shared donde se creó

La industria del software se recomienda en el Shared

lo importante es hacerlos visibles en el \_imports.razor en sus ubicaciones

El framework crea todas las carpetas Shared Service Client, son necesarias y crean un orden que le dan calidad al producto y son parte de las buenas practicas

En Client\Services\ServiceFilm.cs

using FilmManagerDvpSltn.App.Shared.Entities;

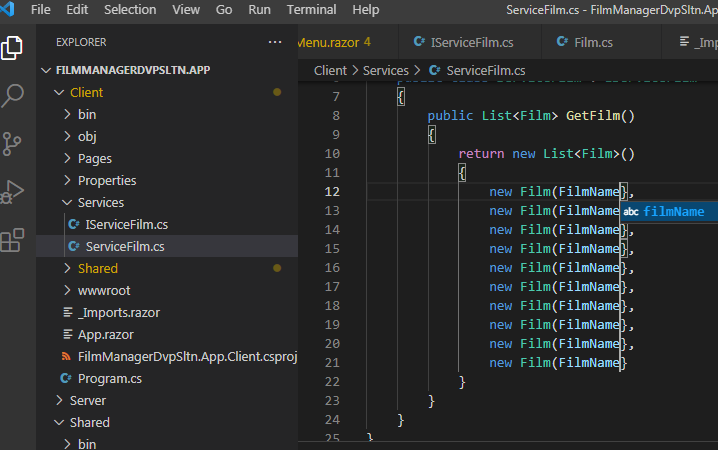
using FilmManagerDvpSltn.App.Collections.Generic;

se llama Entities que es la clase que contiene la estructura del Film

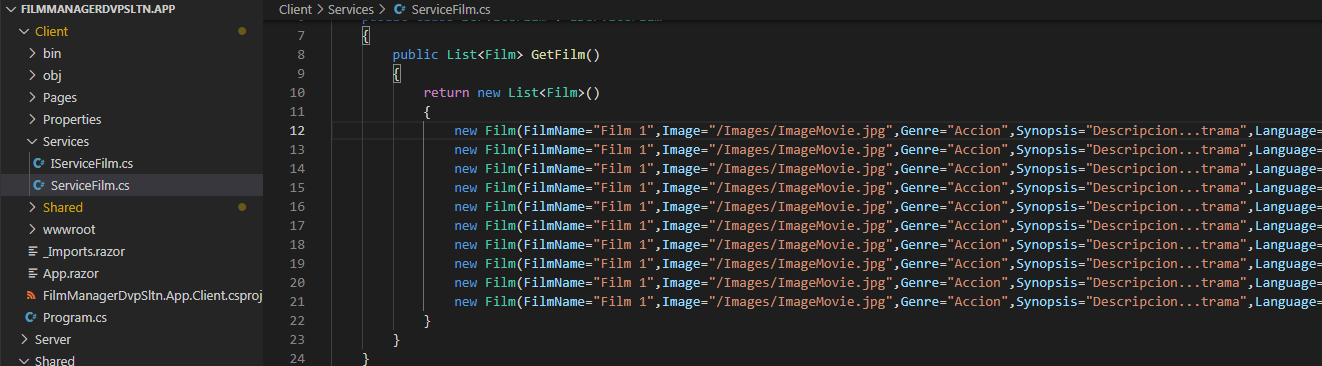
y se llama Collections que permite trabajar con la lista

crear una función publica que nos permite llamar la lista

Habilitar el copiado multiline seleccionando ){ + oprimiendo (ctrl+d) varias veces hasta elegir todas las líneas que contienen ){



Implementación del método List<Film> en ServiceFilm.cs



Se agregan todos los atributos creados en la clase Film.cs dentro de \Shared\Entities

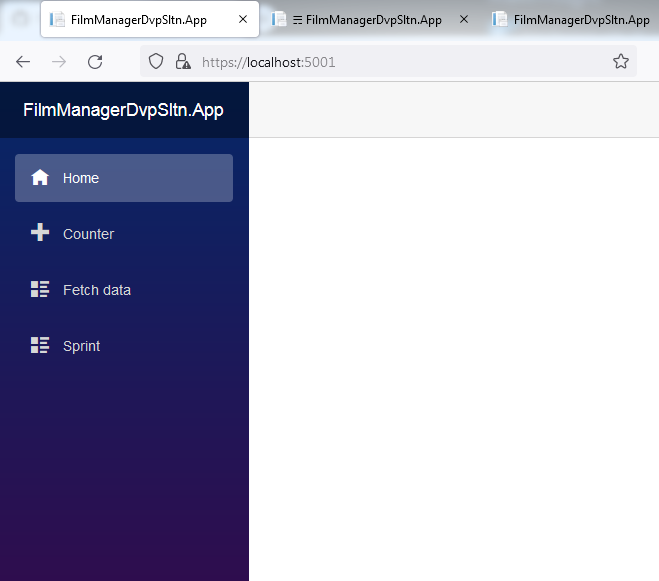
Crear carpeta Client\wwwroot\Films para almacenar las imágenes de los Films, descargados de internet

En Pages crear el componente Cards

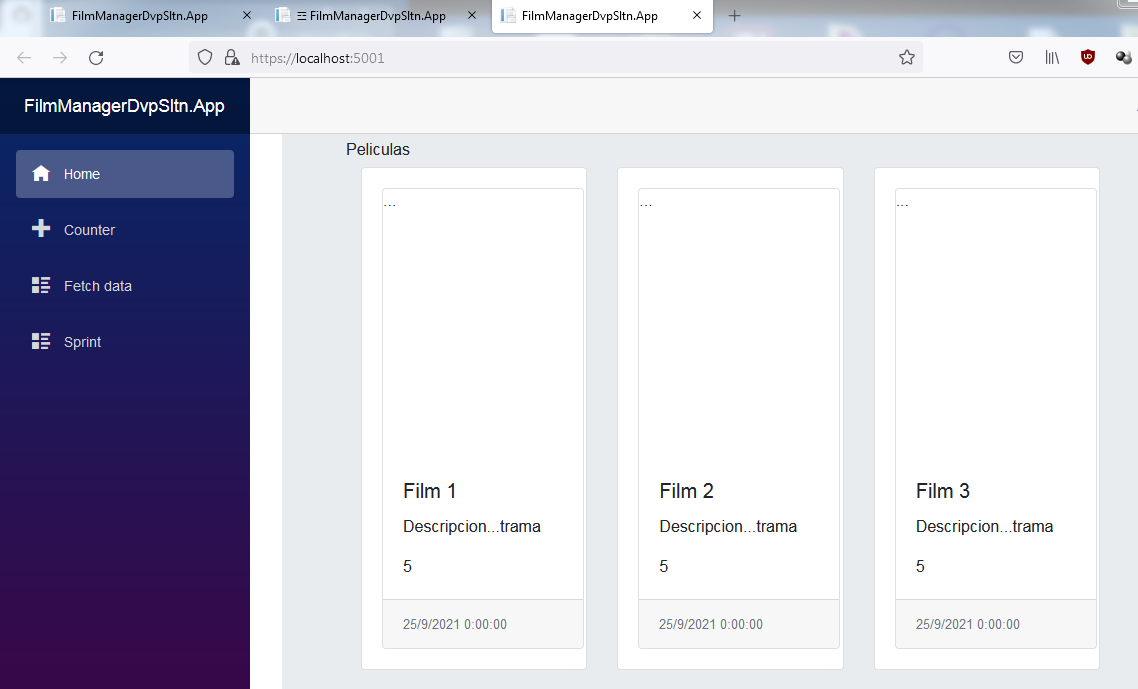
FilmCardComponent.razor

**Resultado**

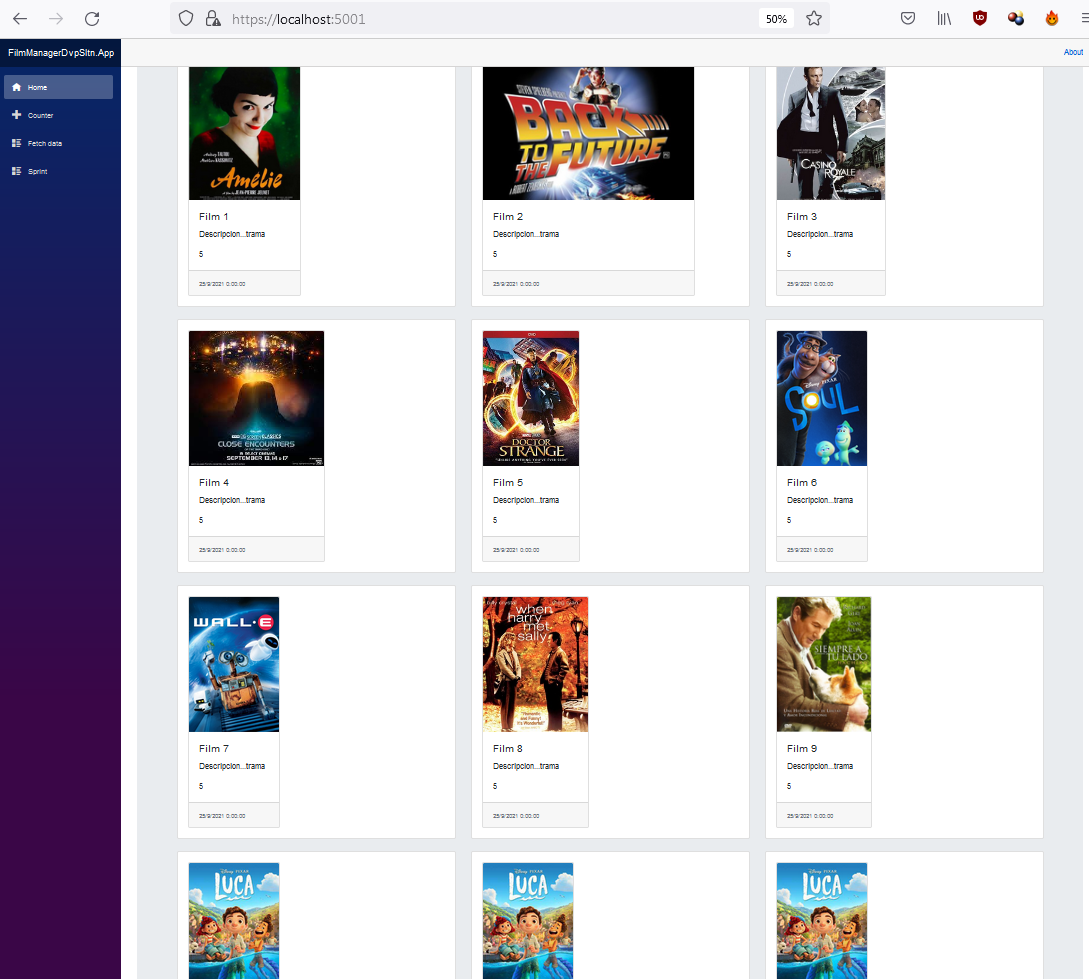
Version 1



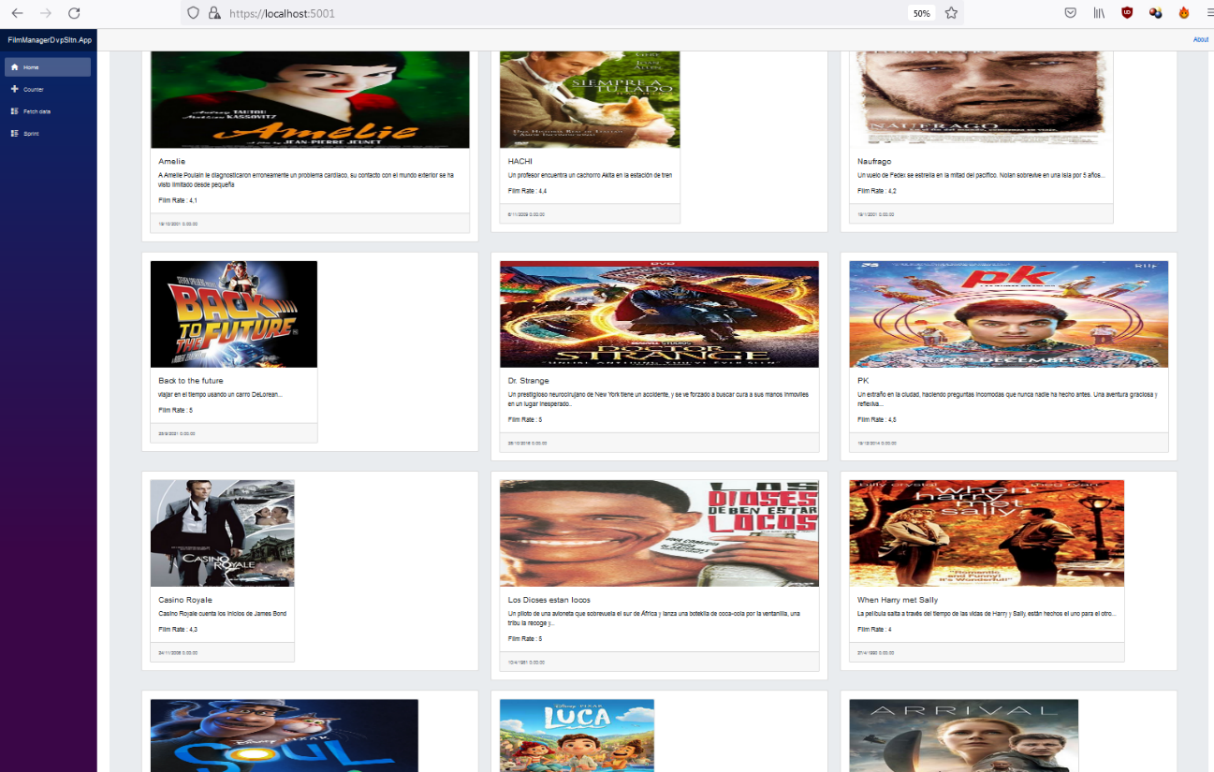
Version 2



Version 3



Version 4



Version 5

