

Manual técnico

Índice

1-Información sobre la aplicación.....	3
Descripción del proyecto.....	3
Justificación de la aplicación a desarrollar.....	3
2-Requisitos.....	4
Requisitos del proyecto.....	4
Recursos del equipo.....	4
3-Tecnologías.....	5
Frontend.....	5
Backend.....	5
Explicación.....	5
XAML.....	5
Material Design.....	5
MahApps.Metro.....	5
C#.....	6
Entity Framework Core.....	6
4-Organización y desarrollo de la aplicación.....	7
Diagrama de la aplicación.....	7
Orden de las carpetas.....	8
Interfaces.....	9
Diseño de interfaces.....	9
Fuentes.....	15
Colores.....	16
Acceso a datos y programación.....	17
MODEL.....	17
DAL.....	23
Otras clases de interés.....	30
5-Conclusiones.....	34
Propuestas de mejora.....	34

1-Información sobre la aplicación

Descripción del proyecto

En la cultura consumerista del día de hoy es normal que mucha gente quiera tener una lista de todo aquello que desea leer, jugar, probar, ver, etc. Y con tantos productos y posibilidades es difícil llevar la cuenta de lo que se tiene y se quiere hacer. Muchos harían una lista, pero lo que busca el OmniBacklog es el hacer fácil el anotar y ordenar los libros que uno quiere leer.

Justificación de la aplicación a desarrollar

La mayor parte de páginas web del estilo suelen ser un tanto molestas de usar, en la mayor parte de los casos los objetos que añades se añaden de una manera muy desordenada y poco atractiva, y a simple vista es difícil ver qué es secuela de qué, o qué vino primero, sobretodo en el caso de los libros y eso es lo que esta aplicación intenta arreglar, con ella se puede ver a qué saga pertenecen los libros, y si esas sagas tienen distintas sagas dentro, que rara vez es el caso, se verá todo en una estructura de árbol que deja claro qué pertenece a qué.

2-Requisitos

Requisitos del proyecto

Lo que el programa necesita realizar es lo siguiente:

- Añadir, editar, eliminar y buscar libros, sagas, autores y géneros.
- Poder ver los libros según los autores y los géneros.
- Crear usuarios, como es local y de uso no empresarial no es necesario diferenciar roles.
- Cada usuario sólo se podrá borrar a sí mismo.
- Los usuarios podrán añadir libros a su biblioteca.
- Los usuarios podrán asignar los libros en su biblioteca como favoritos y si los están leyendo, y también pueden quitarlos.
- Crear un informe de la base de datos.

Recursos del equipo

- SQLExpress. Es necesario tener activada la autenticación de windows, que debería venir por defecto. En caso contrario, se activaría desde Microsoft SQL Management Studio.
- .NET Framework 4.7.2 o superiores

3-Tecnologías

Frontend

- XAML
- Material Design
- Metro MahApps

Backend

- C#
- Entity Framework Core
- iTextSharp

Explicación

XAML

eXtensible Application Markup Language es el lenguaje de formato para la interfaz de usuario para la Base de Presentación de Windows (WPF) y Silverlight (WPF/e), el cual es uno de los pilares de la interfaz de programación de aplicaciones .NET en su versión 3.0

Es un lenguaje declarativo basado en XML, optimizado para describir gráficamente interfaces de usuario.



Material Design

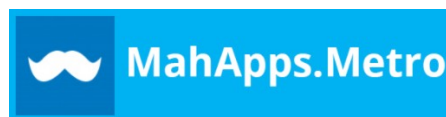
Herramientas para el desarrollo de interfaces con XAML. Proporcionan un vistoso y moderno diseño en las ventanas utilizadas.

Contienen diferentes estilos y nuevos controles para una configuración mejor y más sofisticada de la interfaz de usuario.



MahApps.Metro

Es un framework que permite a los desarrolladores crear una UI más moderna para sus aplicaciones de WPF con esfuerzo



mínimo. Su utilización es similar a Material Design.

C#

Lenguaje de programación multiparadigma desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C y C++ y utiliza el modelo de objetos de la plataforma .NET.



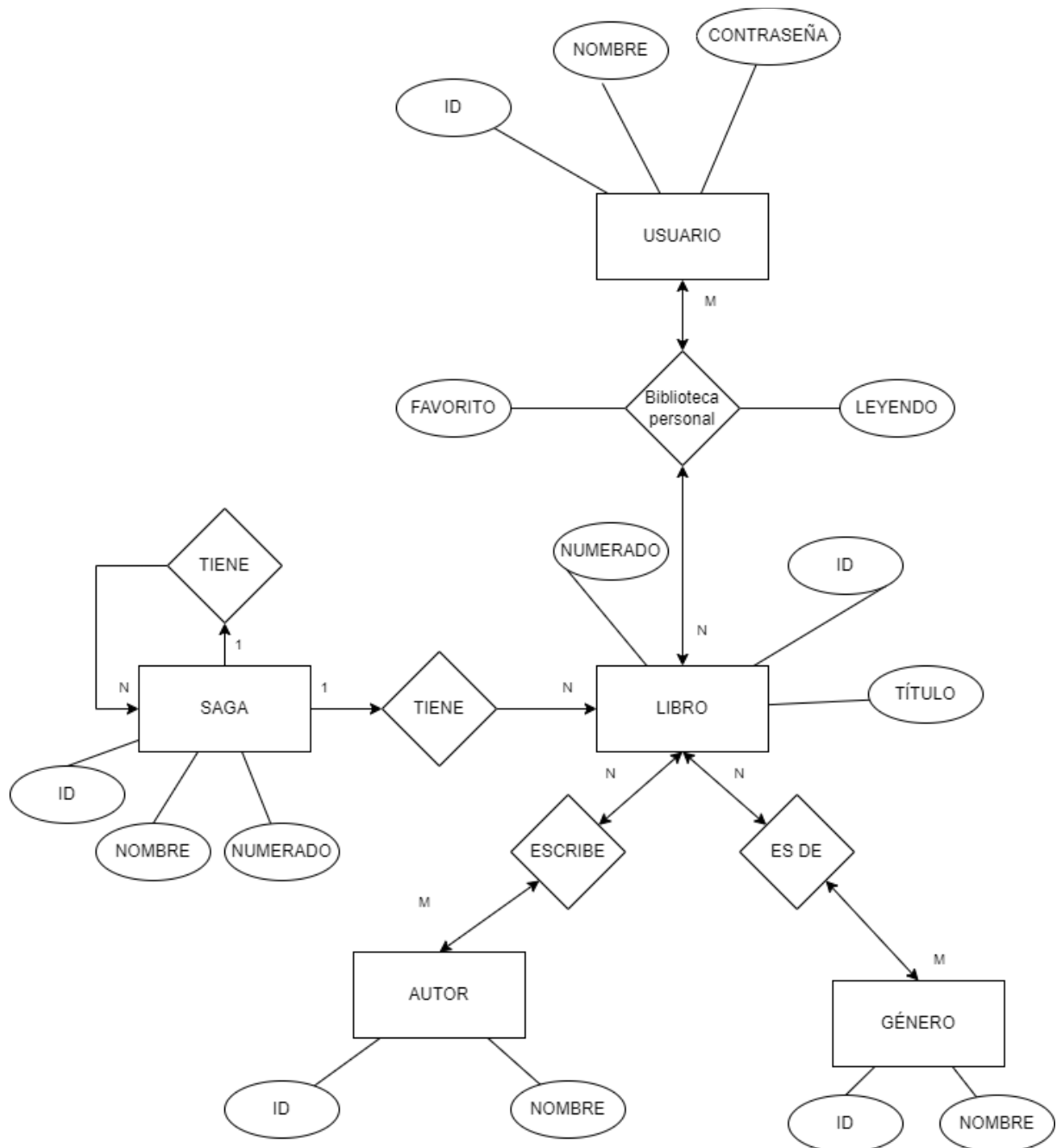
Entity Framework Core

Entity Framework Core es un asignador de base de datos de objeto moderno para .NET. Admite consultas LINQ, seguimiento de cambios, actualizaciones y migraciones de esquemas. EF Core funciona con una gran variedad de bases de datos, incluidas SQL Database, SQLite, MySQL, PostgreSQL, y Azure Cosmos DB.

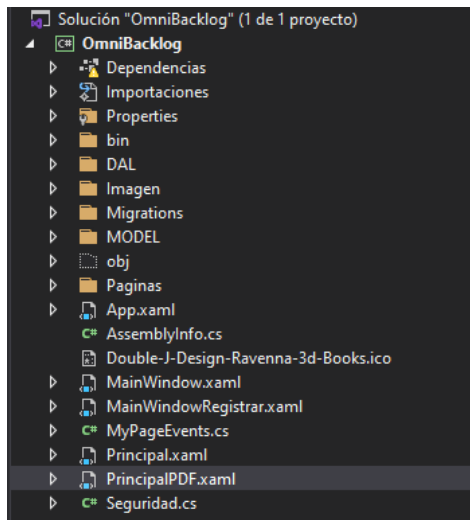


4-Organización y desarrollo de la aplicación

Diagrama de la aplicación



Orden de las carpetas



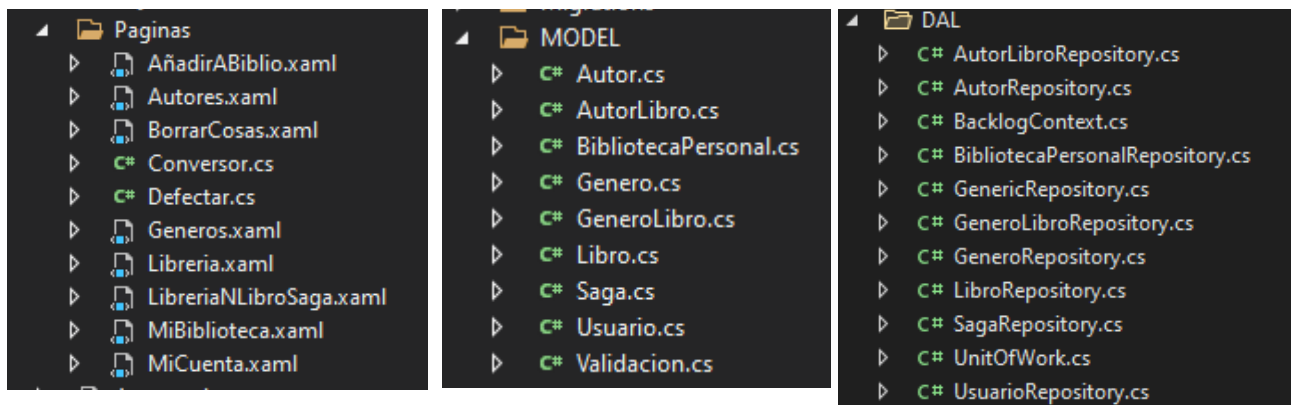
En la solución se pueden ver las carpetas, fuera de las cuales están las ventanas de registro (MainWindowRegistrar), inicio de sesión (MainWindow), la Principal y una ventana que solamente lee un PDF (PrincipalPDF) que será la guía de usuario o el informe. También están las clases MyPageEvents y Seguridad.

En la carpeta Paginas están las páginas que se verán dentro de la ventana Principal y que serán las que traten con toda la información pertinente a los libros.

En MODEL están las clases de los objetos que guarda la base de datos (Libro, Saga, Autor, etc.) y con las que se trabaja en todo el proyecto además de la clase Validacion.

En DAL están las clases de repositorio que se utilizan para la extracción de datos de la base de datos además del UnitOfWork, que permite el guardar, añadir, cambiar y eliminar los datos.

Finalmente, en Migrations están las migraciones creadas para la base de datos SQLExpress que se utiliza en esta aplicación



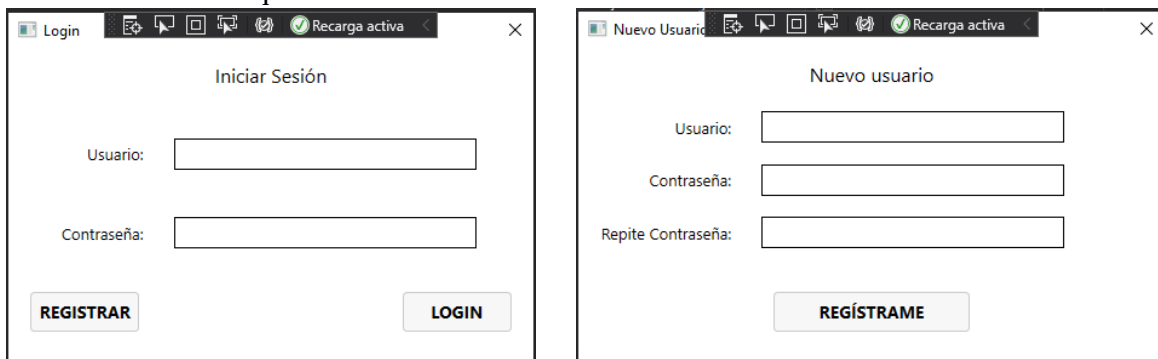
Las clases y sus usos se verán más adelante en Acceso a Datos y Programación.

Interfaces

Diseño de interfaces

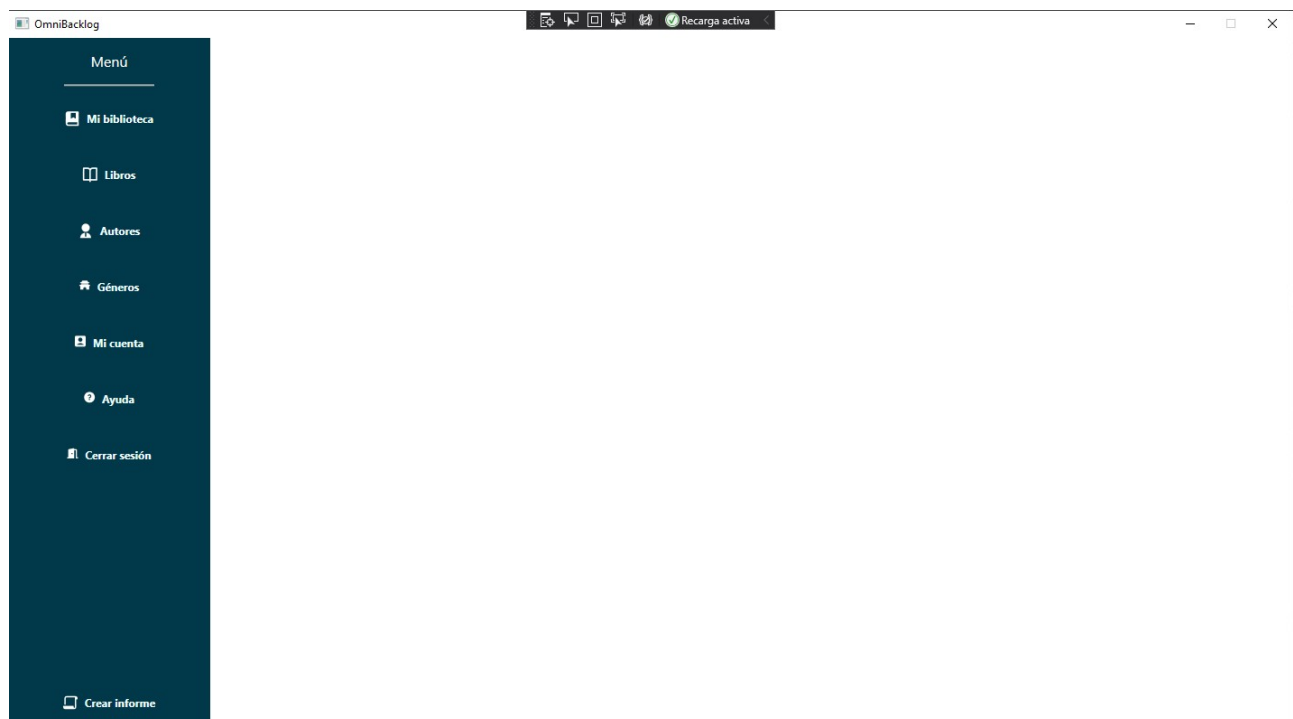
El principal objetivo en el diseño de las ventanas de la aplicación fue el desglosar la base de datos para fácil manipulación de datos.

- Las primeras ventanas que veamos serán las de inicio de sesión y la de registro, que hacen exactamente lo que su nombre indica.



The image shows two side-by-side screenshots of web application windows. The left window is titled 'Login' and 'Iniciar Sesión'. It contains two input fields: 'Usuario:' and 'Contraseña:'. Below the 'Contraseña:' field are two buttons: 'REGISTRAR' on the left and 'LOGIN' on the right. The right window is titled 'Nuevo Usuario' and 'Nuevo usuario'. It contains three input fields: 'Usuario:', 'Contraseña:', and 'Repite Contraseña:'. Below the 'Repite Contraseña:' field is a single button labeled 'REGÍSTRAME'. Both windows have a browser-style header with a 'Recarga activa' status indicator.

- Al iniciar sesión, se abrirá la ventana principal en la que se hará todo, tiene en su lado izquierdo varios botones, de los cuales la mayoría abren páginas en el lado derecho de la ventana.



The image shows a screenshot of the main application window, titled 'OmniBacklog'. On the left side, there is a dark blue sidebar menu with the title 'Menú'. The menu items are: 'Mi biblioteca' (with a book icon), 'Libros' (with a book icon), 'Autores' (with a person icon), 'Géneros' (with a book icon), 'Mi cuenta' (with a person icon), 'Ayuda' (with a question mark icon), and 'Cerrar sesión' (with a logout icon). At the bottom of the sidebar is a button labeled 'Crear informe' (with a document icon). The main content area of the window is currently empty.

- Mi biblioteca** muestra la página de la biblioteca personal, la cual tiene dos DataGrids que muestran los libros que has asignado a la biblioteca y los libros que están siendo

leídos actualmente.

El checkbox *Sólo favoritos* sirve para cambiar la información del que muestra el DataGrid *Biblioteca* a sólo los libros que están asignados como favoritos.

Los botones a la derecha de ambos DataGrids tienen funciones auto explicativas, *Quitar de leyendo* quita los libros del DataGrid *Leyendo*, *Leer* los asigna como leyendo, *Favorito* como favorito, y *Sacar de la biblioteca* borra los libros seleccionados de la biblioteca personal.

Finalmente, se pueden buscar los libros por saga o por título, mostrando los resultados en el DataGrid según las coincidencias que haya.

The screenshot shows the OmniBacklog application interface. On the left is a dark blue sidebar menu with options: 'Menú', 'Mi biblioteca', 'Libros', 'Autores', 'Géneros', 'Mi cuenta', 'Ayuda', 'Cerrar sesión', and 'Crear informe'. The main content area is divided into two sections: 'Leyendo' and 'Biblioteca'.

Leyendo Section: Contains a table with 4 columns: ID, TÍTULO, NUMERADO, and SAGA. It lists one book: ID 1, 'White Sand', NUMERADO 1, SAGA 'The Cosmere'. To the right of this table is a button labeled 'QUITAR DE LEYENDO'.

Biblioteca Section: Includes a search area with two radio buttons: 'Buscar por libro' (selected) and 'Buscar por saga'. A text input field contains 'Introduce nombre de saga o título del libro' and a 'BUSCAR' button. Below the search area is a checkbox labeled 'Sólo favoritos'. The table below has the same 4 columns as the 'Leyendo' table and lists 10 books:

ID	TÍTULO	NUMERADO	SAGA
1	White Sand	1	The Cosmere
2	Elantris	2	The Cosmere
3	The Emperors Soul	3	The Cosmere
4	The Final Empire	1	Mistborn the Original Trilogy
5	The Well of Ascension	2	Mistborn the Original Trilogy
6	The Hero of Ages	3	Mistborn the Original Trilogy
7	Mistborn: Secret History	4	Mistborn the Original Trilogy
8	The Alloy of Law	1	Wax and Wayne
9	Shadows of Self	2	Wax and Wayne
10	The Bands of Mourning	3	Wax and Wayne

To the right of the 'Biblioteca' table are four buttons: 'LEER', 'FAVORITO', 'SACAR DE LA BIBLIOTECA', and 'AÑADIR LIBROS'.

- Hacer click en *Añadir Libros* abre una ventana a mayores que permite al usuario buscar y añadir libros a su biblioteca personal directamente.

Añadir a la biblioteca

Introduce el título del libro

BUSCAR

ID	TÍTULO	NUMERADO	SAGA
1	White Sand	1	The Cosmere
2	Elantris	2	The Cosmere
3	The Emperors Soul	3	The Cosmere
4	The Final Empire	1	Mistborn the Original Trilogy
5	The Well of Ascension	2	Mistborn the Original Trilogy
6	The Hero of Ages	3	Mistborn the Original Trilogy
7	Mistborn: Secret History	4	Mistborn the Original Trilogy
8	The Alloy of Law	1	Wax and Wayne
9	Shadows of Self	2	Wax and Wayne
10	The Bands of Mourning	3	Wax and Wayne
11	Harry Potter y la Piedra Filosofal	1	
12	Harry Potter y la Cámara Secreta	2	
13	Harry Potter y el Prisionero de Azkaban	3	
14	Harry Potter y el Cáliz de Fuego	4	

AÑADIR A BIBLIOTECA

- En **Libros**, la página que se abre muestra un TreeView con una vista en árbol de las sagas que hay y los libros que pertenecen a ellas. Siempre van a existir dos sagas: Huérfanos, para guardar los libros de sagas que se borran, y Únicos, para libros que no forman parte de ninguna saga. Al seleccionar un elemento del TreeView, sea Saga o Libro, los atributos de éste se verán en el formulario de la derecha, el cual permite la edición. Al escoger una Saga, sólo los elementos *Título o nombre* y *Numerado* podrán ser editados, pero si es un libro, se le podrá editar casi todo. Los atributos son auto explicativos a excepción de numerado, que se refiere al orden de lectura de los libros (si una saga tiene orden de lectura que la pone después de un libro de la saga padre, la saga se comenzaría después de dicho libro). Los botones *Añadir* y *Quitar* referentes a Autores y Géneros añaden o quitan los respectivos atributos del libro seleccionado. Si no tiene autor o género o se le quitan todos, ambos tendrán Sin Asignar como autor/género.

The screenshot shows the OmniBacklog application interface. On the left is a dark blue sidebar menu with options: Menú, Mi biblioteca, Libros (selected), Autores, Géneros, Mi cuenta, Ayuda, and Cerrar sesión. At the bottom of the sidebar is 'Crear informe'. The main area has a search bar with 'Buscar libro' and 'Buscar saga' radio buttons, a text input 'Introduce nombre de saga o título del libro', and a 'BUSCAR' button. Below the search bar is a list of categories: Huérfanos, Únicos, Harry Potter, and El Cosmere. At the bottom of the main area are two buttons: 'AÑADIR LIBRO/SAGA' and 'ELIMINAR Y MOVER'. On the right is a 'Selección' modal with fields for 'Título o nombre' (with a character limit), 'Saga padre', 'Orden de lectura' (with + and - buttons), and two lists for 'Géneros' and 'Autores' (each with 'AÑADIR->' and 'QUITAR' buttons). A 'GUARDAR CAMBIOS' button is at the bottom of the modal.

- El botón *Añadir libro/saga* de la página anterior abre otra ventana que nos permitirá añadir libros y sagas a la base de datos. Esta ventana contiene un checkbox *Principal* que al marcarlo hace que las sagas creadas sean todas sin padre y los botones *Borrar* borran el contenido de los formularios correspondientes. Por lo demás, funcionan de similar manera a la página Libros

The screenshot shows the 'Añadir libros y sagas' modal window. It has two columns: 'Nueva saga' and 'Nuevo libro'. The 'Nueva saga' column has a 'Saga padre' dropdown, a 'Saga (nueva):' text input with a character limit, a 'Principal' checkbox, and a 'Numerado Saga:' spinner. The 'Nuevo libro' column has a 'Saga padre' dropdown, a 'Título:' text input with a character limit, a 'Numerado Libro:' spinner, a 'Géneros:' list box with 'Fantasía medieval', 'Fantasía baja', and 'Ciencia ficción', and an 'Autores:' list box with 'George RR Martin', 'JK Rowling', and 'Brandon Sanderson'. At the bottom of each column are 'AÑADIR SAGA' and 'BORRAR' buttons. A 'Recarga activa' button is at the top right of the modal.

- Mientras que el botón *Eliminar y Mover* abre una ventana que permite cambiar las sagas padres de los libros y las sagas disponibles o borrarlas. Se pueden hacer una a una o en masa

Eliminar y mover en masa

Recarga activa

Sagas

ID	NOMBRE	NUMERA	SAGA PADRE
8	Canción de Hielo y Fuego	0	
7	Harry Potter	0	
5	Mistborn the Original Trilogy	1	The Cosmere
3	The Cosmere	0	
6	Wax and Wayne	5	The Cosmere

Libros

ID	TÍTULO	NUMERA	SAGA
1	White Sand	1	The Cosmere
2	Elantris	2	The Cosmere
3	The Emperors Soul	3	The Cosmere
4	The Final Empire	1	Mistborn the Original Trilogy
5	The Well of Ascension	2	Mistborn the Original Trilogy
6	The Hero of Ages	3	Mistborn the Original Trilogy
7	Mistborn: Secret History	4	Mistborn the Original Trilogy
8	The Alloy of Law	1	Wax and Wayne
9	Shadows of Self	2	Wax and Wayne
10	The Bands of Mourning	3	Wax and Wayne
11	Harry Potter y la Piedra Filosofal	1	Harry Potter
12	Harry Potter y la Cámara Secreta	2	Harry Potter
13	Harry Potter y el Prisionero de Azkaban	3	Harry Potter
14	Harry Potter y el Cáliz de Fuego	4	Harry Potter
15	Harry Potter y el Orden del Fénix	5	Harry Potter
16	Harry Potter y el Misterio del Príncipe	6	Harry Potter
17	Harry Potter y las Reliquias de la Muerte	7	Harry Potter
19	Choque de Reyes	2	Canción de Hielo y Fuego
20	Tormenta de Espadas	3	Canción de Hielo y Fuego
21	Festín de Cuervos	4	Canción de Hielo y Fuego

BORRAR

CAMBIAR PADRE

BORRAR

CAMBIAR PADRE

HACER PRINCIPAL

- Las páginas **Autores** y **Géneros** funcionan de la misma manera, con dos ListView que muestran los géneros/autores, y un DataGridView que muestra los libros que tienen esos géneros/autores. Ambas páginas tendrán un objeto por defecto llamado Sin Asignar, que sirve para aquellos libros cuyo autor o género aún no ha sido asignado. Se pueden editar los nombres de los objetos, además de borrarlos (excepto Sin Asignar), y también se pueden añadir. Al seleccionar libros se podrán asignar varios autores/géneros de una sola vez (esto sobrescribirá lo anteriormente asignado), aunque si se le da al checkbox *Sin Asignar* sólo será asignado a sin asignar.

Menú

- Mi biblioteca
- Libros
- Autores
- Géneros
- Mi cuenta
- Ayuda
- Cerrar sesión
- Crear informe

Sin asignar

George RR Martin
JK Rowling
Brandon Sanderson
Stephen King

Editar nombre: Sin asignar

Libros

ID	TÍTULO	NUMERADO	SAGA
11	Harry Potter y la Piedra Filosofal	1	Harry Potter
12	Harry Potter y la Cámara Secreta	2	Harry Potter
13	Harry Potter y el Prisionero de Azkaban	3	Harry Potter
14	Harry Potter y el Cáliz de Fuego	4	Harry Potter
15	Harry Potter y la Orden del Fénix	5	Harry Potter
16	Harry Potter y el Misterio del Príncipe	6	Harry Potter
17	Harry Potter y las Reliquias de la Muerte	7	Harry Potter
19	Choque de Reyes	2	Canción de Hielo y Fuego
20	Tormenta de Espadas	3	Canción de Hielo y Fuego
21	Festín de Cuervos	4	Canción de Hielo y Fuego
22	Danza de Dragones	5	Canción de Hielo y Fuego
23	Vientos de Invierno	6	Canción de Hielo y Fuego
24	Sueños de Primavera	7	Canción de Hielo y Fuego

Añadir autor

Nombre:

Cambiar autores

☐ Sin asignar

George RR Martin
JK Rowling
Brandon Sanderson
Stephen King

Menú

- Mi biblioteca
- Libros
- Autores
- Géneros
- Mi cuenta
- Ayuda
- Cerrar sesión
- Crear informe

Sin asignar

Fantasía medieval
Fantasía baja
Ciencia ficción
Magia
Terror
Suspense
Cosa

Editar nombre: Nombre: máximo 30 caracteres

Libros

ID	TÍTULO	NUMERADO	SAGA
----	--------	----------	------

Añadir género

Nombre:

Cambiar géneros

☐ Sin asignar

Fantasía medieval
Fantasía baja
Ciencia ficción
Magia
Terror
Suspense
Cosa

- En **Mi cuenta** se puede editar el nombre de usuario y la contraseña, además de borrar la cuenta si así se desea.

OmniBacklog

Menú

- Mi biblioteca
- Libros
- Autores
- Géneros
- Mi cuenta
- Ayuda
- Cerrar sesión
- Crear informe

Usuario

Número de libros en mi biblioteca personal: 10

DATOS DEL USUARIO

Nombre de usuario: CAMBIAR NOMBRE DE USUARIO

Nueva contraseña: CAMBIAR CONTRASEÑA

Repite contraseña: CAMBIAR CONTRASEÑA

BORRAR CUENTA

- El botón **Ayuda** abre una ventana con un WebView el cual abre un PDF de guía de usuario.
- **Cerrar sesión** cierra la sesión del usuario y la ventana principal y vuelve a abrirse la ventana de inicio de sesión.
- Finalmente, **Crear informe** genera un PDF con toda la información disponible en las tablas de la base de datos.

Fuentes

Uso la fuente de Material Designs, llamada Roboto. Todas las ventanas y páginas tendrán en la XAML, en donde van los espacios de nombre `FontFamily="{DynamicResource MaterialDesignFont}"`.

- Los títulos tienen 16px de tamaño.
- El texto de los formularios y las etiquetas tienen 12px.
- El texto de los botones de la ventana de inicio de sesión y de registrar tienen 14px, los del menú principal tienen 12px, los botones grandes de la página **Libros** y los botones de **Mi cuenta** tienen 16px, los de la ventana de añadir libros y sagas tienen 10px y el resto tiene 13px.
- El texto de **Mi cuenta** tiene 20px, a excepción del título, que tiene 24px.

Colores

No utilicé una gama de colores muy amplia, prefiriendo algo simple que no cargase demasiado la vista y que parezca algo formal.

- Color del menú de Principal y sus botones: #02394A
- Color de los fondos: Blanco
- Color de todos los botones: #FFF7F7F7
- Color del formulario para edición de los objetos libros y sagas: #FFE7E9EC

Acceso a datos y programación

MODEL

En MODEL están las clases de los objetos que se usarán para gestionar la base de datos además de la clase de validación. Las clases son las siguientes:

- Autor

```
namespace OmniBacklog.MODEL
{
    24 referencias
    public class Autor
    {
        15 referencias
        public int AutorId { get; set; }

        [Required(ErrorMessage = "Nombre obligatorio")]
        [StringLength(50, ErrorMessage = "Nombre Max 50 caracteres")]
        [MinLength(1, ErrorMessage = "Nombre min 1 caracter")]
        8 referencias
        public string Nombre { get; set; }

        4 referencias
        public virtual ICollection<AutorLibro> AutorLibros { get; set; }

        3 referencias
        public Autor()
        {
            AutorLibros = new HashSet<AutorLibro>();
        }
    }
}
```

- Genero

```
namespace OmniBacklog.MODEL
{
    25 referencias
    public class Genero
    {
        16 referencias
        public int GeneroId { get; set; }

        [Required(ErrorMessage = "Nombre obligatorio")]
        [StringLength(30, ErrorMessage = "Nombre Max 30 caracteres")]
        [MinLength(1, ErrorMessage = "Nombre min 1 caracter")]
        8 referencias
        public string Nombre { get; set; }

        4 referencias
        public virtual ICollection<GeneroLibro> GeneroLibros { get; set; }

        3 referencias
        public Genero()
        {
            GeneroLibros = new HashSet<GeneroLibro>();
        }
    }
}
```

- Libro

```
namespace OmniBacklog.MODEL
{
    43 referencias
    public class Libro
    {
        24 referencias
        public int LibroId { get; set; }

        [Required(ErrorMessage = "Título obligatorio")]
        [StringLength(50, ErrorMessage = "Título máximo 50 caracteres")]
        [MinLength(1, ErrorMessage = "Título mínimo 1 caracter")]
        17 referencias
        public string Titulo { get; set; }

        7 referencias
        public int? Numerado { get; set; }

        4 referencias
        public virtual ICollection<AutorLibro> AutorLibros { get; set; }
        5 referencias
        public virtual ICollection<GeneroLibro> GeneroLibros { get; set; }
        1 referencia
        public virtual ICollection<BibliotecaPersonal> BibliotecasPersonales { get; set; }

        [ForeignKey("Saga")]
        10 referencias
        public int? SagaId { get; set; }
        4 referencias
        public virtual Saga Saga { get; set; }

        3 referencias
        public Libro()
        {
            AutorLibros = new HashSet<AutorLibro>();
            GeneroLibros = new HashSet<GeneroLibro>();
            BibliotecasPersonales = new HashSet<BibliotecaPersonal>();
        }
    }
}
```

- Saga

```
namespace OmniBacklog.MODEL
{
    56 referencias
    public class Saga
    {
        21 referencias
        public int SagaId { get; set; } //Identificador de las sagas

        [Required(ErrorMessage = "Nombre obligatorio")]
        [StringLength(50, ErrorMessage = "Nombre de la saga máximo 50 caracteres")]
        [MinLength(1, ErrorMessage = "Nombre de la saga mínimo 1 caracter")]
        13 referencias
        public string Nombre { get; set; } //Nombre de las sagas

        9 referencias
        public int? Numerado { get; set; } //Si hay sagas dentro de sagas, hace falta ordenarlas

        10 referencias
        public virtual ICollection<Saga> Sagas { get; set; }

        5 referencias
        public virtual ICollection<Libro> Libros { get; set; }

        [ForeignKey("Saga1")]
        8 referencias
        public int? Saga1Id { get; set; }

        3 referencias
        public virtual Saga Saga1 { get; set; }

        6 referencias
        public Saga()
        {
            Sagas = new HashSet<Saga>();
            Libros = new HashSet<Libro>();
        }
    }
}
```

- Usuario

```

namespace OmniBacklog.MODEL
{
    22 referencias
    public class Usuario
    {
        16 referencias
        public int UsuarioId { get; set; }

        [Required(ErrorMessage = "Nombre obligatorio")]
        [StringLength(30, ErrorMessage = "Nombre Max 30 caracteres")]
        [MinLength(6, ErrorMessage = "Nombre min 6 caracteres")]
        15 referencias
        public string Nombre { get; set; }

        [Required(ErrorMessage = "Contraseña obligatoria")]
        [StringLength(50, ErrorMessage = "Contraseña Max 50 caracteres")]
        [MinLength(6, ErrorMessage = "Contraseña min 6 caracteres")]
        9 referencias
        public string Contraseña { get; set; }

        6 referencias
        public virtual ICollection<BibliotecaPersonal> BibliotecasPersonales { get; set; }

        3 referencias
        public Usuario()
        {
            BibliotecasPersonales = new HashSet<BibliotecaPersonal>();
        }
    }
}

```

- AutorLibro

```

namespace OmniBacklog.MODEL
{
    32 referencias
    public class AutorLibro
    {
        12 referencias
        public int AutorId { get; set; }

        11 referencias
        public int LibroId { get; set; }

        2 referencias
        public virtual Autor Autor { get; set; }

        1 referencia
        public virtual Libro Libro { get; set; }
    }
}

```

- GeneroLibro

```
namespace OmniBacklog.MODEL
{
    32 referencias
    public class GeneroLibro
    {
        12 referencias
        public int GeneroId { get; set; }

        11 referencias
        public int LibroId { get; set; }

        1 referencia
        public virtual Libro Libro { get; set; }

        2 referencias
        public virtual Genero Genero { get; set; }
    }
}
```

- BibliotecaPersonal

```
namespace OmniBacklog.MODEL
{
    29 referencias
    public class BibliotecaPersonal
    {
        13 referencias
        public int LibroId { get; set; }

        12 referencias
        public int UsuarioId { get; set; }

        5 referencias
        public bool Leyendo { get; set; }

        5 referencias
        public bool Favorito { get; set; }

        3 referencias
        public virtual Libro Libro { get; set; }

        1 referencia
        public virtual Usuario Usuario { get; set; }
    }
}
```

- Validacion

```
namespace OmniBacklog.MODEL
{
    7 referencias
    public class Validacion
    {
        7 referencias
        public static string errores(Object obj)
        {
            string mensError = "";

            ValidationContext validationContext = new ValidationContext(obj, null, null);
            List<ValidationResult> errores = new List<ValidationResult>();
            Validator.TryValidateObject(obj, validationContext, errores, true);

            if (errores.Count() > 0)
            {
                string mensajeErrores = string.Empty;
                foreach (var error in errores)
                {
                    error.MemberNames.First();

                    mensError += error.ErrorMessage + "\n";
                    // mensError += error.ErrorMessage + Environment.NewLine;
                }
                return mensError;
            }
            else
            {
                return mensError;
            }
        }
    }
}
```


DAL

Los repositorios en DAL más el UnitOfWork nos permitirán tratar la información de los objetos en la base de datos. Hay un repositorio genérico, un repositorio por cada objeto y el UnitOfWork mencionado anteriormente que es el que tiene la función de, utilizando como plantilla el repositorio genérico, dar uso a las diferentes funciones de añadir, borrar, modificar, etc. También está el BacklogContext que se encarga de conectar la conexión a la base de datos, darle valores al crearse y otras funciones.

- GenericRepository tiene las funciones básicas que todos los repositorios deben de tener.

```
namespace OmniBacklog.DAL
{
    17 referencias
    public class GenericRepository<TEntity> where TEntity : class
    {
        internal BacklogContext context;
        internal DbSet<TEntity> dbSet;

        8 referencias
        public GenericRepository(BacklogContext context)
        {
            this.context = context;
            dbSet = context.Set<TEntity>();
        }

        22 referencias
        public void Update(TEntity entity)
        {
            context.Entry(entity).State = EntityState.Modified;
        }

        11 referencias
        public void Delete(TEntity entityToDelete)
        {
            if (context.Entry(entityToDelete).State == EntityState.Detached)
            {
                dbSet.Attach(entityToDelete);
            }
            dbSet.Remove(entityToDelete);

            context.Entry(entityToDelete).State = EntityState.Deleted;
        }

        2 referencias
        public void Delete(Expression<Func<TEntity, bool>> predicate)
        {

```

- AutorRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class AutorRepository : GenericRepository<Autor>
    {
        1 referencia
        public AutorRepository(BacklogContext context) : base(context) { }
    }
}
```

- AutorLibroRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class AutorLibroRepository : GenericRepository<AutorLibro>
    {
        1 referencia
        public AutorLibroRepository(BacklogContext context) : base(context) { }

        1 referencia
        public List<AutorLibro> getAutores(int id)
        {
            return Get(a => a.LibroId == id, includeProperties: "Autor");
        }

        1 referencia
        public List<AutorLibro> getLibros(int id)
        {
            return Get(a => a.AutorId == id, includeProperties: "Libro");
        }

        1 referencia
        public List<AutorLibro> getEverything()
        {
            return Get(includeProperties: "Libro,Autor");
        }
    }
}
```

- GeneroRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class GeneroRepository : GenericRepository<Genero>
    {
        1 referencia
        public GeneroRepository(BacklogContext context) : base(context) { }

        0 referencias
        public Genero getOneGenero(int id)
        {
            return Get(a => a.GeneroId == id).FirstOrDefault();
        }
    }
}
```


- GeneroLibroRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class GeneroLibroRepository : GenericRepository<GeneroLibro>
    {
        1 referencia
        public GeneroLibroRepository(BacklogContext context) : base(context) { }

        1 referencia
        public List<GeneroLibro> getGeneros(int id)
        {
            return Get(a => a.LibroId == id, includeProperties: "Genero");
        }

        1 referencia
        public List<GeneroLibro> getLibros(int id)
        {
            return Get(a => a.GeneroId == id, includeProperties: "Libro");
        }

        1 referencia
        public List<GeneroLibro> getEverything()
        {
            return Get(includeProperties: "Libro,Genero");
        }
    }
}
```

- LibroRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class LibroRepository : GenericRepository<Libro>
    {
        1 referencia
        public LibroRepository(BacklogContext context) : base(context) { }

        0 referencias
        public List<Libro> getSpecificBooks(string titulo)
        {
            return Get(a => a.Titulo.Contains(titulo));
        }

        2 referencias
        public Libro getSagaFromLibro(int id)
        {
            return Get(a => a.LibroId == id, includeProperties: "Saga").FirstOrDefault();
        }

        1 referencia
        public List<Libro> getSagaAndLibro()
        {
            return Get(includeProperties: "Saga");
        }

        1 referencia
        public List<Libro> getGenerosAutores()
        {
            return Get(includeProperties: "AutorLibros,GeneroLibros");
        }
    }
}
```

- SagaRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class SagaRepository : GenericRepository<Saga>
    {
        1 referencia
        public SagaRepository(BacklogContext context) : base(context) { }

        1 referencia
        public Saga getLibrosFromSagas(int id)
        {
            return Get(a => a.SagaId == id, includeProperties: "Libros").FirstOrDefault();
        }

        2 referencias
        public List<Saga> GetTree()
        {
            return Get(a => a.SagaId == null, includeProperties: "Sagas,Libros,Saga1");
        }
    }
}
```

- UsuarioRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class UsuarioRepository : GenericRepository<Usuario>
    {
        1 referencia
        public UsuarioRepository(BacklogContext context) : base(context) { }

        1 referencia
        public Usuario GetUsuario(string nombre)
        {
            return Get(a => a.Nombre == nombre, includeProperties: "BibliotecasPersonales").FirstOrDefault();
        }
    }
}
```

- BibliotecaPersonalRepository

```
namespace OmniBacklog.DAL
{
    4 referencias
    public class BibliotecaPersonalRepository : GenericRepository<BibliotecaPersonal>
    {
        1 referencia
        public BibliotecaPersonalRepository(BacklogContext context) : base(context) { }

        3 referencias
        public List<BibliotecaPersonal> getBibliotecaPersonal(int id)
        {
            return Get(a => a.UsuarioId == id);
        }

        5 referencias
        public List<BibliotecaPersonal> getBiblioteca(int id)
        {
            return Get(a => a.UsuarioId == id, includeProperties: "Libro,Libro.Saga");
        }

        1 referencia
        public List<BibliotecaPersonal> getLeyendo(int id)
        {
            return Get(a => a.UsuarioId == id && a.Leyendo == true, includeProperties: "Libro,Libro.Saga");
        }

        1 referencia
        public List<BibliotecaPersonal> getFavoritos(int id)
        {
            return Get(a => a.UsuarioId == id && a.Favorito == true, includeProperties: "Libro,Libro.Saga");
        }

        1 referencia
        public List<BibliotecaPersonal> getEverything()
        {
            return Get(includeProperties: "Libro,Usuario");
        }
    }
}
```

- UnitOfWork: se encarga de asociar los repositorios al genérico para poder usar sus funciones y de crear la base de datos al iniciar la aplicación

```
namespace OmniBacklog.DAL
{
    21 referencias
    public class UnitOfWork //: IDisposable
    {
        private BacklogContext context = new BacklogContext();
        private bool disposed = false;

        private AutorLibroRepository autorLibrosRepository;
        private AutorRepository autoresRepository;
        private BibliotecaPersonalRepository biblioRepository;
        private GeneroLibroRepository genLibRepository;
        private GeneroRepository generoRepository;
        private LibroRepository libroRepository;
        private SagaRepository sagaRepository;
        private UsuarioRepository usuarioRepository;

        13 referencias
        public AutorLibroRepository AutorLibroRepository
        {
            get
            {
                if (autorLibrosRepository == null)
                {
                    autorLibrosRepository = new AutorLibroRepository(context);
                }

                return autorLibrosRepository;
            }
        }

        12 referencias
        public AutorRepository AutorRepository
        {
            get
            {
                if (autoresRepository == null)
                {
                    autoresRepository = new AutorRepository(context);
                }

                return autoresRepository;
            }
        }
    }
}
```

- BacklogContext: tiene el contexto de la base de datos. Es la clase que se conecta y añade datos si estos han sido añadidos a la clase OnModelCreating(), clase que también sirve para dar PKs a las tablas intermedias.

```
namespace OmniBacklog.DAL
{
    14 referencias
    public class BacklogContext : DbContext
    {
        0 referencias
        public virtual DbSet<Autor> Autores { get; set; }
        0 referencias
        public virtual DbSet<AutorLibro> AutorLibros { get; set; }
        0 referencias
        public virtual DbSet<BibliotecaPersonal> BibliotecasPersonales { get; set; }
        0 referencias
        public virtual DbSet<Genero> Generos { get; set; }
        0 referencias
        public virtual DbSet<GeneroLibro> GenerosLibros { get; set; }
        0 referencias
        public virtual DbSet<Libro> Libros { get; set; }
        0 referencias
        public virtual DbSet<Saga> Sagas { get; set; }
        0 referencias
        public virtual DbSet<Usuario> Usuarios { get; set; }

        0 referencias
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Data Source=.\SQLEXPRESS;Initial Catalog=OmniBacklog ;Integrated Security=True"); //User Id = sa; Password = sqlserver
        }

        0 referencias
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<AutorLibro>().HasKey(c => new { c.AutorId, c.LibroId});
            modelBuilder.Entity<BibliotecaPersonal>().HasKey(c => new { c.UsuarioId, c.LibroId});
            modelBuilder.Entity<GeneroLibro>().HasKey(c => new { c.LibroId, c.GeneroId});

            modelBuilder.Entity<Usuario>().HasIndex(u => u.Nombre).IsUnique();
            modelBuilder.Entity<Saga>().HasIndex(u => u.Nombre).IsUnique();
            modelBuilder.Entity<Libro>().HasIndex(u => u.Titulo).IsUnique();
            //modelBuilder.Entity<Genero>().HasIndex(u => u.Nombre).IsUnique();
            //modelBuilder.Entity<Autor>().HasIndex(u => u.Nombre).IsUnique();
        }
    }
}
```


Otras clases de interés

Finalmente, hay otras tres clases que no son código de las interfaces, aunque dos de ellas las afectan directamente.

- MyPageEvents crea el header y el footer del informe de la base de datos

```
namespace OmniBacklog
{
    2 referencias
    class MyPageEvents : PdfPageEventHelper
    {
        // This is the contentbyte object of the writer
        PdfContentByte cb;

        // we will put the final number of pages in a template
        PdfTemplate headerTemplate, footerTemplate;

        // this is the BaseFont we are going to use for the header / footer
        BaseFont bf = null;

        // This keeps track of the creation time
        DateTime PrintTime = DateTime.Now;

        Fields

        Properties

        0 referencias
        public override void OnOpenDocument(PdfWriter writer, Document document)
        {
            try
            {
                PrintTime = DateTime.Now;
                bf = BaseFont.CreateFont(BaseFont.HELVETICA, BaseFont.CP1252, BaseFont.NOT_EMBEDDED);
                cb = writer.DirectContent;
                headerTemplate = cb.CreateTemplate(100, 100);
                footerTemplate = cb.CreateTemplate(50, 50);
            }
            catch (DocumentException de)
            {
            }
            catch (System.IO.IOException ioe)
            {
            }
        }

        0 referencias
        public override void OnEndPage(iTextSharp.text.pdf.PdfWriter writer, iTextSharp.text.Document document)
    }
}
```

- Defectar se usa al entrar en la ventana **Principal** y se encarga de recorrer todos los libros, y aquellos que no tengan padre serán asignados automáticamente a la saga Únicos.

```
namespace OmniBacklog.Paginas
{
    4 referencias
    public class Defectar
    {
        4 referencias
        public static void sinAsignar()
        {
            UnitOfWork bd = new UnitOfWork();
            List<Libro> libros = bd.LibroRepository.getGenerosAutores();
            GeneroLibro genLib;
            AutorLibro autLib;

            for (int i = 0; i < libros.Count; i++)
            {
                if (libros[i].SagaId == null)
                {
                    libros[i].SagaId = 2;
                }

                if (libros[i].GeneroLibros.Count == 0)
                {
                    genLib = new GeneroLibro();
                    genLib.GeneroId = 1;
                    genLib.LibroId = libros[i].LibroId;
                    bd.GeneroLibroRepository.Añadir(genLib);
                }
                if (libros[i].AutorLibros.Count == 0)
                {
                    autLib = new AutorLibro();
                    autLib.AutorId = 1;
                    autLib.LibroId = libros[i].LibroId;
                    bd.AutorLibroRepository.Añadir(autLib);
                }
            }

            bd.Save();
        }
    }
}
```

- Conversor se encarga de que funcione la estructura en árbol del TreeView de la página **Libros**, metiendo a los libros en las sagas como si estas últimas fuesen carpetas.

```
namespace OmniBacklog.Paginas
{
    0 referencias
    public class Conversor : IMultiValueConverter
    {
        /// <summary>
        ///
        /// </summary>
        /// <param name="values"></param>
        /// <param name="targetType"></param>
        /// <param name="parameter"></param>
        /// <param name="culture"></param>
        /// <returns></returns>
        0 referencias
        public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)
        {
            //Colección que toma los items superiores
            List<object> items = new List<object>();

            //Recorre los valores y los añade a la lista de hijos
            for (int i = 0; i < values.Length; i++)
            {
                IEnumerable childs = values[i] as IEnumerable ?? new List<object> { values[i] };

                //Recorre los hijos de childs y los añade a la lista de items
                foreach (var child in childs) { items.Add(child); }
            }

            return items;
        }

        0 referencias
        public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo culture)
        {
            throw new NotSupportedException("Cannot perform reverse-conversion");
        }
    }
}
```


- Seguridad se encarga de encriptar las contraseñas que ponen los usuarios, aunque también está la función desencriptar, ésta no es utilizada de momento.

```
namespace OmniBacklog
{
    public static class Seguridad
    {
        /// Encripta una cadena

        public static string Encriptar(this string _cadenaAencriptar)
        {
            string result = string.Empty;
            byte[] encryted = System.Text.Encoding.Unicode.GetBytes(_cadenaAencriptar);
            result = Convert.ToBase64String(encryted);
            return result;
        }

        /// Esta función desencripta la cadena que le enviamos en el parámetro de entrada.

        public static string DesEncriptar(this string _cadenaAdesencriptar)
        {
            string result = string.Empty;
            byte[] decryted = Convert.FromBase64String(_cadenaAdesencriptar);
            //result = System.Text.Encoding.Unicode.GetString(decryted, 0, decryted.ToArray().Length);
            result = System.Text.Encoding.Unicode.GetString(decryted);
            return result;
        }
    }
}
```

5-Conclusiones

Propuestas de mejora

- Arreglo de bugs.
- Optimización de código.
- Mejoras de interfaz.
 - Hacerla responsive.
 - Mejorar la búsqueda en la página **Libros**, mostrando sólo los resultados con coincidencias al buscar libros y no las sagas que contengan dichos libros.
 - Que aparezcan los libros y las sagas del TreeView de **Libros** en orden de lectura en vez de las sagas hijas primero y los libros después.
- Mayor personalización del usuario, como imagen de perfil y roles.
- Más información sobre las sagas y los libros (sinopsis, resúmenes, fechas de publicación, editoriales, etc.)
- Migrar la base de datos a una en la nube, como Azure.
- Extender uso a otros medios de entretenimiento como series, películas, videojuegos, etc.