1. Crear una Class Library (Linq.Completo.Entidades) donde vamos a guardar las entidades que nos dan (Author, Books).
2. Crear la clase Author y pegamos la información que nos da el enunciado y lo mismo con Books.
3. Crear una Class Library (Linq.Completo.Servicios) donde vamos a guardar las interfaces y los métodos.
    3.1. La Class Library se utiliza porque genera un fichero .DLL que contiene toda la lógica y se puede copiar y llevar a otros proyectos y reutilizarlos. Se suben a un repositorio para que sean descargadas (como lo que conocemos en Node como npm).
4. Crear AutorExtendido para mezclar los datos del autor con libros publicados, número de ventas…
    4.1. Nos piden NumeroLibrosPublicados y TituloLibro (los creamos como variables con el get y el set) y todos los datos que tiene el autor(por herencia :Author).
    4.2. El constructor de AuthorExtendido tendrá además del id y el name, numeroLibrosPublicado en sus parámetros y debemos de darle un valor (NumeroLibrosPublicado = librosNumeroPublicado porque en el ejercicio 3 nos pide el autor con más libros publicados así que lo tendremos que pintar (aún no sabemos si necesitaremos pintar el título de los libros o no). Más tarde se añadirá otro constructor para inicializar TituloLibro = tituloLibro, pasando además del id y el name, el tituloLibro.
    4.3. Se queja porque necesita un constructor que la misma IDE te lo monta (pasándolo al constructor base).
5. Crear la interfaz con los métodos de Autores que se van a necesitar.
6. Crear la interfaz con los métodos de Libros que se van a necesitar.
7. Crear la clase MetodosAutores y que implemente IMetodosAutores.
    7.1. Como vamos a hacer consultas de mezclar autores con libros necesitamos un par de orígenes de datos. Nos montamos una lista con los autores y una lista con los libros.
8. Crear la clase MetodosLibros y que implemente IMetodosLibros (cuando se queje le damos Ctrl+. o Alt+Enter e implementa todos los métodos que están en la interfaz)
    8.1. Vamos a necesitar un origen de datos con los datos de los libros así que creamos una lista de libros para conseguir la información.
9. En Program.cs vamos a montar todas las llamadas que nos pide el ejercicio:
    9.1. Escribimos IMetodosAutores, le damos Crtl+. para añadir la referencia a Linq.Completo.Servicios. Lo mismo haremos con IMetodosLibros. Con esto podremos acceder desde Program a los métodos implementados en la interfaz.
    9.2. Libros con más ventas:
        9.2.1. Crear una variable con el top número de vantas que nos indican.

9.2.2. Crear un boolean para indicar/cambiar el sentido de si será ascendente o descendente. En este caso esSentidoDescendente = false porque buscamos los 3 con más ventas.

9.2.3. Crear variable con un ternario para que guarde si el texto es ascendente o descendente y poder pintarlo luego en la consola.

9.2.4. Crear la variable que llame al método correspondiente y pasándole el las dos variables como parámetro (boolean y la variable con el top)

9.3. Como se va a necesitar mostrar la lista de libros durante todo el ejercicio se crea un método para poder reutilizarlo. Como parámetro se le pasa List<Book> y como no lo encuentra con Ctrl+. y marcar que use Linq.Completo.Entidades.

9.4. Libros con menos ventas:

9.4.1. Es igual que el anterior pero cambiarndo esSentidoDescendente a true para que nos dé los 3 con menos ventas.

9.5. Autor con más libros publicados:

9.5.1. Se crea la variable que ejecute de metodosAutores el método que corresponda.

9.5.2.Como se va a necesitar mostrar la lista de autores durante todo el ejercicio se crea un método para poder reutilizarlo. Se crea estableciendo que el parámetro es una lista de AuthorExtendido (si tiene título quiero que me lo pinte y sino que no me lo pinte. Esto siempre va a devolver una lista.

9.5.3. Como en el ejercicio solo queremos mostrar un único autor que sea el que más libros tiene publicados y el método MostrarDatosAutores me devuelve una lista, así que cuando se llama a MostrarMetodosAutores hay que montar un new List<AuthorExtendido> y pasándole valor de la variable que se crea para extraer la información, en este caso {autorMasLibrosPublicados).

9.6. Autores con cantidad libros publicados:

9.7. Libros publicados en los últimos 50 años:

9.7.1. Crear variable con los libros publicados en los últimos 50 años.

9.7.2. Crear variable llamando al método correspondiente pasándole la variable como parámetro.

9.8. Libro más viejo:

9.8.1. Crear variable llamando al método correspondiente.

9.8.2. A MostrarDatosLibros le pasamos un new List<Book> {libroMasViejo} para que nos pinte un solo libro.

9.9. Libros que comiencen por “El”:

9.9.1. Crear la variable indicando por qué texto empieza “El”.

9.9.2. Crear variable llamando al método correspondiente y pasándole el parámetro la variable por la que queremos buscar.

10. En MetodosAutores:

10.1. GetAutoresConCantidadLibrosPublicados: join entre autores y libros, después un group by por id autor y con ese listado de agrupamiento hay que meterlo en un listado de AuthorExtendido metiendo el id, nombre y número de libros. Para los agrupamientos se tienen que hacer dos bucles foreach, en el primero te da la clave/key (en este caso el id) y luego en el segundo hay un listado asociado que es al siguiente al que hacemos un foreach y hacer un Count.

10.2. GetAutoresConLibroQueComiencePor(string comiencePor): join entre autores y libros y filtrar con un StartsWith (pasando el parámetro) y usar ToLower para comprar los textos.

10.3. GetAutorMayorNumeroLibrosPublicados: join entre autores y libros, después un group by por id autor, guardar en una variable el número de libros con un count y después ordenar descendentemente y quedarnos con el primero First().

11. En MetodosLibros:

11.1. GetLibroMasViejo: hacer una query, ordenar por fecha de publicación y quedarnos con uno First()

11.2. GetLibrosPublicadosUltimosAnios(int anio): hacer una query, filtrar por años que nos indique por (parámetro) y devolver los resultados ordenados.

11.3. GetTopLibrosVentas(bool esMayorNumeroVentas, int topLibros): preparar una query que obtenga libros, preguntar por el parámetro esMayorNumeroVentas para filtrar ascendente o descendente por número de ventas y quedarte con el topLibros que nos pasan por parámetro (esto se hace con Take)