




Laboratorio II: TP Final



Federico Dacal
2° "A"



Biblioteca UTN-Fra

La aplicación simula la actividad de una biblioteca universitaria en la que sus socios pueden pedir prestado alguna de las publicaciones disponibles en la biblioteca.

El usuario de la aplicación podrá dar de alta, de baja o modificar los datos de los socios y, a su vez, podrá asignarles el préstamo de material y/o tramitar la devolución.

TicketException
Class
↳ Exception

Methods

- TicketException...

ITicket
Interface

Properties

- PathTicket

Methods

- ImprimirTicket

Prestamo
Class

Fields

- costo
- datosSocio
- fechaPrestamo
- publicacion
- socio

Properties

- Costo
- DatosSocio
- FechaPrestamo
- PathTicket
- Publicacion
- Socio

Methods

- AsignarCostoPrestamo
- ImprimirTicket
- Prestamo (+ 1 overload)
- ToString

Publicacion
Abstract Class

Fields

- anioPublicacion
- autor
- editorial
- estaDisponible
- id
- titulo

Properties

- AnioPublicacion
- Autor
- Editorial
- EsMaterialDeUTNFra
- EstaDisponible
- Id
- Tipo
- Titulo

Methods

- operator !=
- operator ==
- Publicacion (+ 1 overload)
- ToString

Biblioteca
Static Class

Fields

- denominacion
- dni
- prestamos
- publicaciones
- socios

Properties

- Denominacion
- Email
- Prestamos
- Publicaciones
- Socios

Methods

- AgregarSocio
- Biblioteca
- ExisteSocio
- PrestarPublicacion
- PublicacionEstaEnCatalogo
- QuitarSocio
- RecibirDevolucion

Socio
Class

Fields

- apellido
- dni
- email
- esEstudiante
- fechaNacimiento
- genero
- listaPrestamos
- nombre

Properties

- Apellido
- Dni
- Edad
- Email
- EsEstudiante
- FechaNacimiento
- Genero
- ListaPrestamos
- Nombre

Methods

- Equals
- ObtenerInformacionCo...
- operator != (+ 1 overloa...
- operator == (+ 1 overlo...
- Socio (+ 1 overload)
- ToString

Nested Types

ISerializableJso...
Generic Interface

Methods

- Deserialize
- Serialize

Serializadora<T>
Generic Class

Methods

- Deserialize
- ISerializableJso...
- ISerializableJso...
- Serialize

Novela
Class
↳ Publicacion

Fields

Paper
Class
↳ Publicacion

Fields

PublicacionUniversitaria
Class
↳ Publicacion

Fields

ECampo

ToString

- Equals
- ObtenerInformacionCo...
- operator != (+ 1 overloa...
- operator == (+ 1 overlo...
- Socio (+ 1 overload)
- ToString
- Nested Types

DateTimeExtens...
Static Class

Methods

- FindYears

Novela
Class
Publication

Fields

- generos

Properties

- EsMaterialDeUTNFra
- Generos

Methods

- Equals
- Novela (+ 1 overload)
- ToString

Nested Types

EGeneroLiterario
Enum

Paper
Class
Publication

Fields

- campos

Properties

- Campos
- EsMaterialDeUTNFra

Methods

- Equals
- Paper (+ 1 overload)
- ToString

PublicacionUniversitaria
Class
Publication

Fields

- campos
- catedra

Properties

- Campos
- Catedra
- EsMaterialDeUTNFra

Methods

- Equals
- PublicacionUniversitaria...
- ToString

- ECampo**
Enum
- Computacion
 - IT
 - IA
 - Matematica
 - CienciaDeDatos
 - Telecomunicaciones
 - Electronica
 - Mecanica
 - Quimica
 - Industrial
 - Sociologia
 - Antropologia
 - Energias
 - Negocios
 - Economia

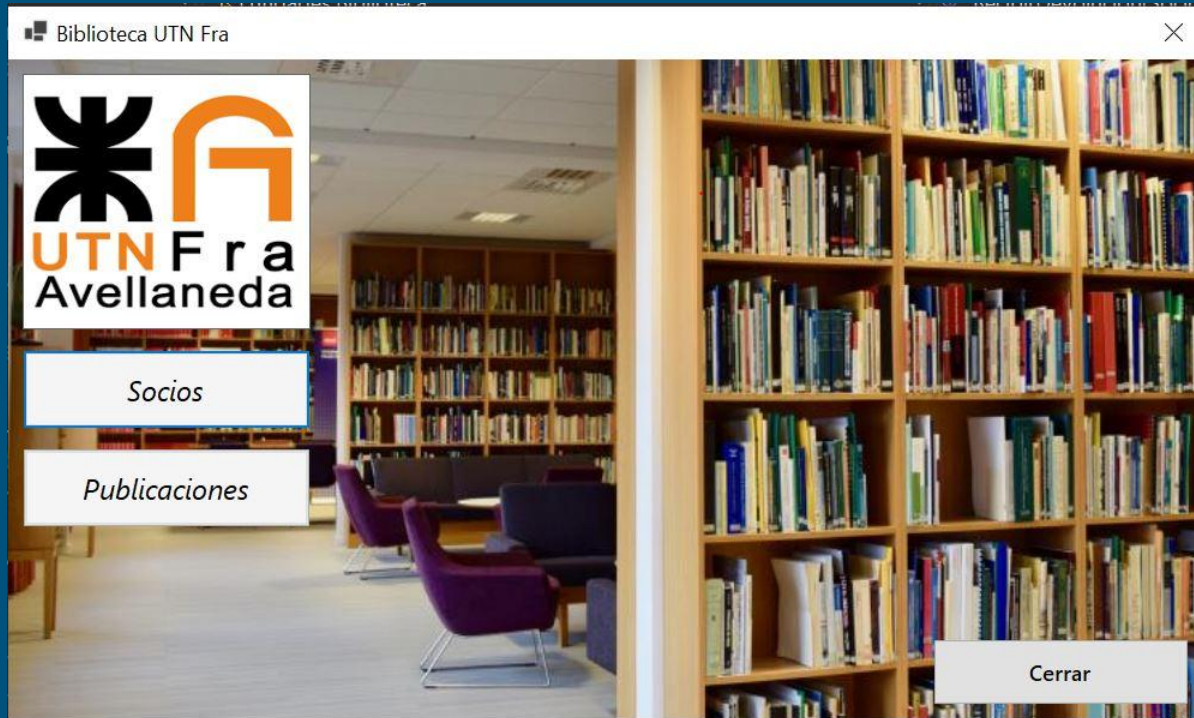
Temario: Clases 10 a 20

- Excepciones
- Unit Testing
- Interfaces
- Generics
- Archivos
- Serialización
- Métodos de Extensión
- SQL
- Conexión ADO.NET
- Delegados
- Hilos
- Eventos



Breve demostración de la aplicación

Menú:



- Socios:

Socios

	Apellido	Nombre	EsEstudiante	Email	Dni	Genero	Edad
▶	Perez	Pedro	<input checked="" type="checkbox"/>	pedroperez@g...	42323765	Masculino	21
	Lopez	Juana	<input checked="" type="checkbox"/>	juanalop@gmai...	4323367	Femenino	21
	Perez	Juan Carlos	<input type="checkbox"/>	juanca@yahoo....	31324445	Masculino	30
	Alonso	Alfonso	<input type="checkbox"/>	alal@gmail.com	32119768	Masculino	28
	Fernandez	Ana	<input checked="" type="checkbox"/>	anaf@gmail.com	38333223	Femenino	29
	Carrasco	Miguel	<input checked="" type="checkbox"/>	miguecarrasco...	40113800	Masculino	22

Nuevo Socio

Modificar

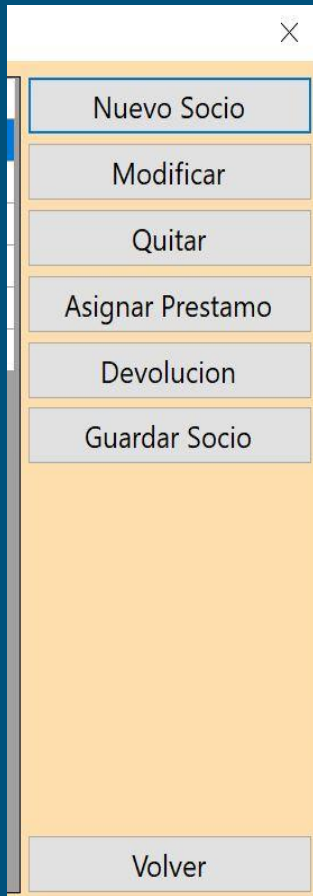
Quitar

Asignar Prestamo

Devolucion

Guardar Socio

Volver



Nuevo Socio: nos permite dar de alta un socio.

Modificar: nos permite modificar los datos de un socio existente.

Quitar: nos permite dar de baja un socio de la lista.

Asignar Préstamo: nos permite cargar en el sistema el préstamo de una publicación al socio seleccionado.

Devolución: nos permite cargar en el sistema la devolución de los distintos préstamos del socio.

Guardar Socio: nos permite guardar en Desktop el estado del socio seleccionado en un archivo Json.

Agregar nuevo socio

Datos Socio

Nombre

Apellido

Ingrese apellido

Documento

Ingrese documento o DNI

Email

Ingrese email

Genero

Masculino

Fecha Nacimiento

Monday , June 6, 2022

¿Estudia en UTN-Fra?

☐ Si

☐ No

Confirmar

Cancelar

Modificar socio Perez

Datos Socio

Nombre

Juan Carlos

Apellido

Perez

Documento

31324445

Email

juanca@yahoo.com

Genero

Masculino

Fecha Nacimiento

Wednesday, February 5, 1992

¿Estudia en UTN-Fra?

☐ Si

☒ No

Guardar Cambios

Cancelar

◀
▶

▼

10

1000

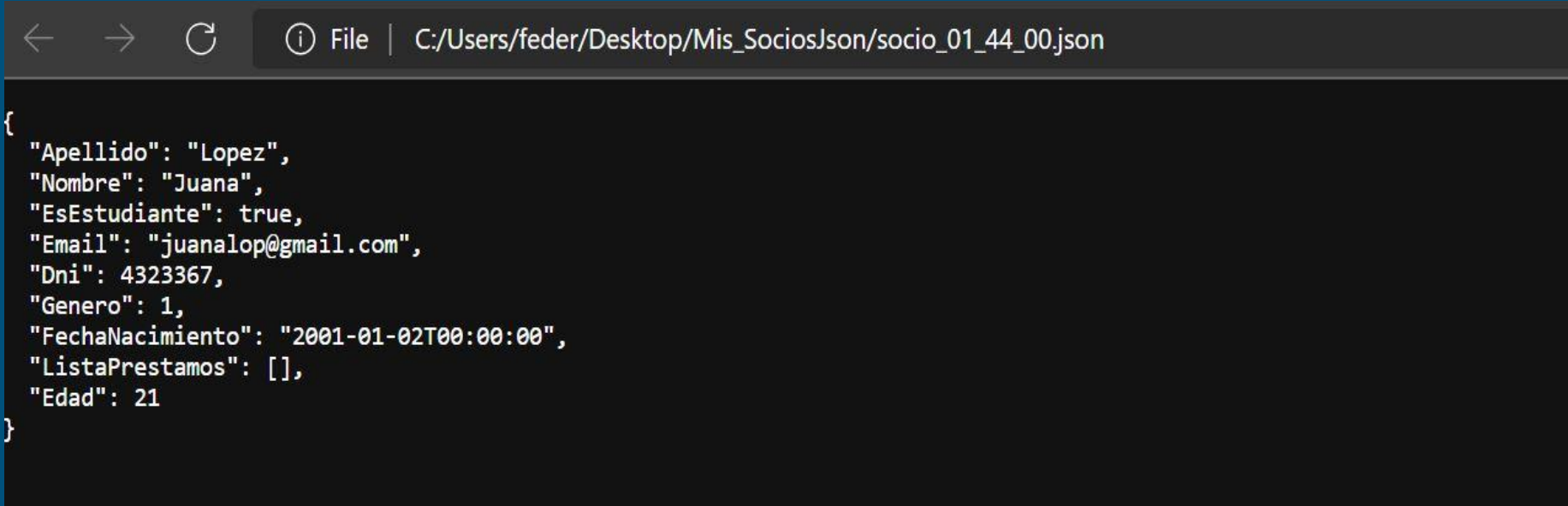
Una vez
asignado, de
manera
automática, se
'imprime' un
ticket (.txt).

```
Ticket_03_17_57 - Notepad
File Edit Format View Help
Biblioteca UTN-Fra
utn-fra@sistemas.com.ar
*****
Socio: Perez, Pedro
DNI: 42323765

Titulo: '1984'
Autor: George Orwell
Editorial: Universal
Año: 1949
Codigo: N101
Genero: CienciaFiccion, Distopia.
6/6/2022 3:17:57 AM
$30

*****
```

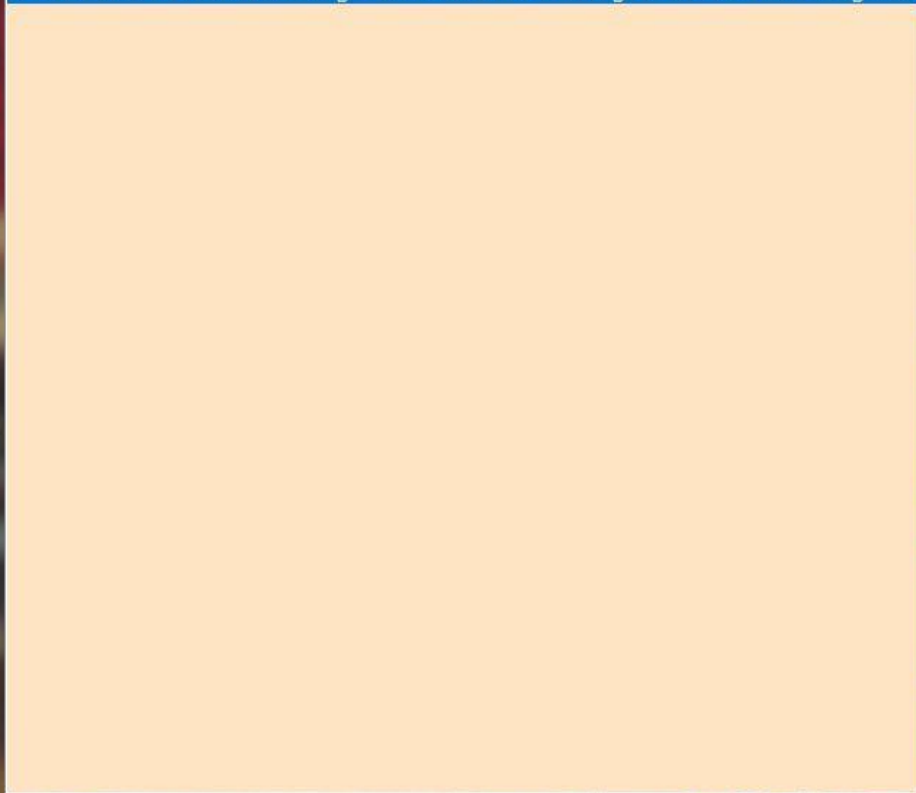
Los tickets se
'imprimen' en un
Directorio
'Ticket' ubicado
en el Escritorio
del usuario.

A screenshot of a file explorer window with a dark theme. The address bar at the top shows the file path: C:/Users/feder/Desktop/Mis_SociosJson/socio_01_44_00.json. The main area displays the content of the JSON file, which is a single object with several fields: Apellido, Nombre, EsEstudiante, Email, Dni, Genero, FechaNacimiento, ListaPrestamos, and Edad. The file is named socio_01_44_00.json and is located in the directory C:/Users/feder/Desktop/Mis_SociosJson/.

```
{
  "Apellido": "Lopez",
  "Nombre": "Juana",
  "EsEstudiante": true,
  "Email": "juanalop@gmail.com",
  "Dni": 4323367,
  "Genero": 1,
  "FechaNacimiento": "2001-01-02T00:00:00",
  "ListaPrestamos": [],
  "Edad": 21
}
```

Con el botón 'Guardar Socio' generamos un archivo .json que guarda el estado del objeto Socio seleccionado. Los archivos .json de objetos de tipo 'Socio' que guardemos se almacenan en el directorio 'Mis_Socios' en el Escritorio del usuario.

Titulo: 'The Waves' Autor: Virginia Woolf Editorial: Hogarth Año: 1931 Código: N



Recibir

Volver

Con el botón 'Devolución' accedemos a la lista de préstamos del socio seleccionado.

Al 'recibir' el préstamo lo marcamos como disponible para que pueda ser prestado de nuevo.

Excepciones

3 references | 1/1 passing

```
public bool ImprimirTicket(string path)
{
    bool rta = false;
    try
    {
        if (!Directory.Exists(this.PathTicket))
        {
            Directory.CreateDirectory(this.PathTicket);
        }

        using (StreamWriter sw = new StreamWriter(Path.Combine(this.PathTicket, $"{path}_{this.FechaPrestamo.ToString("HH_mm_ss")}.txt")))
        {
            sw.WriteLine(Biblioteca.Denominacion);
            sw.WriteLine(Biblioteca.Email);
            sw.WriteLine("*****");
            sw.WriteLine(this.ToString());
            sw.WriteLine("*****");
        }
    }
    catch (Exception e)
    {
        throw new TicketException($"Ocurrió un problema al imprimir ticket: {e.Message}", e);
    }
    return rta;
}
```


Excepciones

El manejo de Excepciones se utiliza a través de todos los puntos pertinentes de la aplicación para gestionar los posibles errores en tiempo de ejecución y evitar que discontinúe el uso del programa.

Se implementa la Excepción propia 'TicketException' que es lanzada en caso que haya alguna dificultad en 'ImprimirTicket' al realizar un préstamo.

Unit Testing

Se realizaron Tests Unitarios para comprobar los métodos:

- Operador == de Clase 'Socio'
- Equals de Clase 'Socio'
- AgregarSocio de Clase 'Biblioteca'
- TicketException en ImprimirTicket

Interfaces

Se da implementación a las interfaces 'ITicket' (implementada en Clase 'Prestamo') y 'ISerializableJson' (implementada en 'Clase' Serializadora).

```
4 references
public interface ISerializableJson<T>
{
    where T : class

    2 references
    void Serializar(string path, T contenido);

    1 reference
    T Deserializar(string path);
}
```

```
1 reference
public interface ITicket
{
    4 references
    string PathTicket { get; }

    3 references | 🟢 1/1 passing
    bool ImprimirTicket(string path);
}
```

Generics

Clase 'Serializadora<T>' es parametrizada e implementa la interface `ISerializableJson<T>` que es una interfaz parametrizada.

`Serializadora<T>` posee los métodos `Serializar` y `Deserializar` para `.xml`. Implementa de manera explícita los métodos `Serializar` y `Deserializar` de la interface `ISerializableJson<T>`.

Archivos

Se utilizan archivos de texto (.txt) para 'imprimir' los tickets de los préstamos realizados.

Se utilizan archivos .xml para persistir los datos de los socios y publicaciones de la aplicación, los cuales se van actualizando a través de la interacción con el usuario y al iniciar la aplicación se los recupera (socios.xml y publicaciones.xml) desde el Directorio base de la aplicación (donde está ubicado el .exe)

Se utilizan archivos .json para guardar el estado de un Socio seleccionada desde la lista.

Serialización

Serialización XML: Recuperación y actualización de los archivos 'socios.xml' y 'publicaciones.xml' al iniciar la aplicación. Ubicados en (directorio base de la app):
C:\Users\User\Desktop\tps_laboratorio_ii\tps_laboratorio_ii\TP3\Dacal.Federico.2A.TPFinal\Vista\bin\Debug\net5.0-windows

Serialización JSON: Persistencia de los datos de un Socio seleccionado, los archivo .json se guardan en el Directorio 'Mis_Socios' del Escritorio.

Método de Extensión

Desarrollo de la clase 'DateTimeExtension' que extiende la clase DateTime. El método de esta clase permite calcular los años entre dos fecha (en la app se usa para calcular la edad de los socios a través de su fecha de nacimiento).

```
public static class DateTimeExtension
{
    1 reference
    public static int FindYears(this DateTime date)
    {
        int years = DateTime.Now.Year - date.Year;
        if (DateTime.Now.Month < date.Month || (DateTime.Now.Month == date.Month &&
            DateTime.Now.Day < date.Day))
        {
            years--;
        }
        return years;
    }
}
```

SQL

Se implementa el lenguaje de consultas SQL para la generación de la base de datos de la aplicación y la realización de consultas y comandos durante su ejecución.


El Script para iniciar la ejecución y conexión a la base de datos se encuentra en el directorio del TP ubicado a la altura del archivo .sln (solución).

Archivo: Script_TP4.sql

SQL

```
CREATE DATABASE DB_BIBLIOTECA_UTN;  
GO
```

```
USE DB_BIBLIOTECA_UTN;  
GO
```

```
 CREATE TABLE Socios  
(dni bigint primary key,  
 nombre varchar(50) not null,  
 apellido varchar(60) not null,  
 email varchar(150) not null,  
 es_estudiante bit not null,  
 genero int not null,  
 fecha_nacimiento date not null);  
GO
```

Conexión ADO.NET

La Biblioteca de Clases 'Entidades' contiene dos clases encargadas de realizar la conexión a la base de datos y realizar las consultas o comandos necesarios para leer o actualizar los datos en la base.

Clase SocioDAO vinculada a la clase Socio.

Clase PrestamoDAO vinculada a la case Prestamo.

Conexión ADO.NET

SocioDAO

Static Class

Fields

-  command
-  connection
-  connectionString

Methods

-  Eliminar
-  Guardar
-  Leer
-  Modificar
-  SocioDAO


PrestamoDAO

Static Class

Fields

-  command
-  connection
-  connectionString

Methods

-  Borrar
-  Guardar
-  Leer
-  PrestamoDAO

Delegados

Se implementa el delegado 'propio' `LimitePrestamosDelegado` que tiene visibilidad a nivel namespace 'Entidades', vinculado al evento 'LimitePrestamosAlcanzado' de la clase estática `Biblioteca`.

A su vez, en el método 'EntregarComprobante' de `Biblioteca` se utiliza una expresión lambda pasada como delegado para la implementación de Hilos.

Delegados

```
namespace Entidades
{
    public delegate void LimitePrestamosDelegado(Socio socio, SocioEventArgs e);

    56 references
    public static class Biblioteca
```

```
await Task.Run(async() =>
```

Eventos

El evento 'LimitePrestamosAlcanzado' de la clase estática Biblioteca, se invoca dentro del método 'PrestarPublicacion' si se comprueba que el Socio ya posee 3 préstamos vigentes y se intenta asignarle uno más, en la aplicación sólo se prestará un máximo de 3 items.

La Clase SocioEventArgs es utilizada para pasar la información necesaria a la invocación del evento.

Eventos

```
public delegate void LimitePrestamosDelegado(Socio socio, SocioEventArgs e);
```

56 references

```
public static class Biblioteca
{
    private static string denominacion;
    private static string email;
    private static List<Publicacion> publicaciones;
    private static List<Prestamo> prestamos;
    private static List<Socio> socios;

    public static event LimitePrestamosDelegado LimitePrestamosAlcanzado;
```

Eventos

4 references

```
public class SocioEventArgs : EventArgs
```

```
{
```

2 references

```
public int PrestamosVigentes { get; set; }
```

```
if(socio.ListaPrestamos.Count < 3)
{
    Prestamo prestamo = new Prestamo(publicacion, socio);
    Biblioteca.Prestamos.Add(prestamo);
    socio.ListaPrestamos.Add(prestamo);
    publicacion.EstaDisponible = false;
    prestamo.ImprimirTicket("Ticket");
    rta = true;
}
else
{
    if(Biblioteca.LimitePrestamosAlcanzado is not null)
    {
        SocioEventArgs eventArgs = new SocioEventArgs();
        eventArgs.PrestamosVigentes = socio.ListaPrestamos.Count;
        Biblioteca.LimitePrestamosAlcanzado.Invoke(socio, eventArgs);
    }
}
```

Hilos

Cuando un socio devuelve un ejemplar, por defecto, se ejecuta de manera asíncrona la 'impresión' de un comprobante de devolución (archivo .txt que se ubicará en el Desktop del usuario en el directorio 'Comprobantes').

Es posible, si se quisiera, cancelar la 'impresión' del comprobante mediante el CancellationToken ubicado en el formulario FrmDevolucion. Dicha ejecución asíncrona posee pequeños tiempo de espera (Task.Delay) para simular un proceso de algunos segundos y además dar la posibilidad de cancelar la generación del archivo .txt antes que se termine de ejecutar esa acción.

Hilos

1 reference

```
public static async Task<bool> EntregarComprobante(Socio s, Prestamo p, CancellationToken ct)
{
    bool rta = false;
    rta = await Task.Run(async() =>
    {
        try
        {
            await Task.Delay(3000);

            if (ct.IsCancellationRequested)
            {
                throw new TaskCanceledException("La impresión del comprobante ha sido detenida");
            }
            else
            {

```




Muchas Gracias

Federico Dacal