LibraryXP en Detalle

Table of Contents

LibraryXP Referencias

Autor

Enrique Fu Orozco

Versión

0.1 (Demo)

Revisión

Revisión Inicial

Espacios de nombres

LibraryXP[6], LibraryXP.Controllers[17], LibraryXP.Data[36]

Introducción

Este proyecto tiene como finalidad demostrar el uso responsable de lectura y escritura basada en .json y también una capa de compatibilidad de Windows XP hacia adelante, hecho y fabricado basado en Terminal (CLI) para compatibilizar equipos antiguos. En él, se mostrará como ejemplo, una gestión de una Biblioteca, en la cual, se registrarán Libros, Autores, Usuarios y Préstamos, para tener un seguimiento de stock y prestados.

LibraryXP (Espacio de nombres)

Clases

Author[6], Book[7], DataBase[10], Loan[11], Program[14], User[14]

Author (Clase)

La clase autor es lo que tiene. Almacena el Nombre del autor solamente para referencias.

object
  LibraryXP.Author

C#

```
internal class Author
```

Requisitos

Espacio de nombres: LibraryXP[6]

Ensamblado: LibraryXP (in LibraryXP.exe)

Propiedades

IdAuthor[6], NameAuthor[6]

Métodos

Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**)

Author.IdAuthor (Propiedad)

C#

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

Código fuente

```
public int IdAuthor { get => idAuthor; set => idAuthor = value; }
```

Vea también

Se aplica a: Author[6]

Author.NameAuthor (Propiedad)

C#

```
public string NameAuthor {get; set;}
```

Código fuente

```
public string NameAuthor { get => nameAuthor; set => nameAuthor = value; }
```

Vea también

Se aplica a: Author[6]

Book (Clase)

El libro es una clase más específica en miembros. Referencia con código para su búsqueda manual, así como la asignación de un Autor, género, y año para una idea a lo que se refiere en caso de repetición o similitudes en títulos. El stock es un límite para realizar préstamos posteriores, evitando la molestia de prestar un libro que no está disponible.

object
LibraryXP.Book

C#

```
internal class Book
```

Requisitos

Espacio de nombres: LibraryXP[6]

Ensamblado: LibraryXP (in LibraryXP.exe)

Propiedades

CodeBook[7], Genre[8], IdAuthor[8], IdBook[8], NameBook[9], TotalStockBook[9], Year[9]

Métodos

Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**)

Book.CodeBook (Propiedad)

C#

```
public string CodeBook {get; set;}
```

Código fuente

```
public string CodeBook { get => codeBook; set => codeBook = value; }
```

Vea también

Se aplica a: Book7

Book.Genre (Propiedad)

C#

```
public string Genre {get; set;}
```

Código fuente

```
public string Genre { get => genre; set => genre = value; }
```

Vea también

Se aplica a: Book7

Book.IdAuthor (Propiedad)

C#

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

Código fuente

```
public int IdAuthor { get => idAuthor; set => idAuthor = value; }
```

Vea también

Se aplica a: Book7

Book.IdBook (Propiedad)

C#

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

Código fuente

```
public int IdBook { get => idBook; set => idBook = value;  }
```

Vea también

Se aplica a: Book7

Book.NameBook (Propiedad)

C#

```
public string NameBook {get; set;}
```

Código fuente

```
public string NameBook { get => nameBook; set => nameBook = value; }
```

Vea también

Se aplica a: Book7

Book.TotalStockBook (Propiedad)

C#

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

Código fuente

```
public int TotalStockBook { get => totalStockBook; set => totalStockBook =
value; }
```

Vea también

Se aplica a: Book7

Book.Year (Propiedad)

C#

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

Código fuente

```
public int Year { get => year; set => year = value; }
```

Vea también

Se aplica a: Book7

DataBase (Clase)

Esta es la "Base de Datos" (Si es que se le puede llamar de esa manera), para reutilizar las clases hechas en la carpeta Models. Como es reutilizable, entonces es fácil de acceder a través de los llamados de los controladores y con el JsonHelper.

object
LibraryXP.DataBase

C#

```
internal class DataBase
```

Requisitos

Espacio de nombres: LibraryXP[6]

Ensamblado: LibraryXP (in LibraryXP.exe)

Propiedades

Authors[10], Books[10], Loans[11], Users[11]

Métodos

Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**)

DataBase.Authors (Propiedad)

C#

```
public List<Author> Authors {get; set;}
```

Código fuente

```
public List<Author> Authors { get; set; } =  new List<Author>();
```

Vea también

Se aplica a: DataBase[10]

DataBase.Books (Propiedad)

C#

```
public List<Book> Books {get; set;}
```

Código fuente

```
public List<Book> Books { get; set; } = new List<Book>();
```

Vea también

Se aplica a: DataBase[10]

DataBase.Loans (Propiedad)

C#

```
public List<Loan> Loans {get; set;}
```

Código fuente

```
public List<Loan> Loans { get; set; } = new List<Loan>();
```

Vea también

Se aplica a: DataBase[10]

DataBase.Users (Propiedad)

C#

```
public List<User> Users {get; set;}
```

Código fuente

```
public List<User> Users { get; set; } = new List<User>();
```

Vea también

Se aplica a: DataBase[10]

Loan (Clase)

El préstamo es bastante condicional si se analiza bien el código. Utiliza clases como el Usuario y el Libro para poder generar el préstamo. Existen miembros en los que definen realmente como un préstamo, como lo son: isActive, dateLoan y returnLoan. Estos datos determinan si está activo el préstamo y estas fechas condicionan el puntaje (O historial) del Usuario, afectando su reputación positivamente o negativamente, dependiendo de su responsabilidad al regresar el libro.

object
  LibraryXP.Loan

C#

```
internal class Loan
```

Requisitos

Espacio de nombres: LibraryXP[6]

Ensamblado: LibraryXP (in LibraryXP.exe)

Propiedades

DateLoan[12], IdBook[12], IdLoan[13], IdUser[13], IsActive[13], ReturnLoan[13]

Métodos

Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**)

Loan.DateLoan (Propiedad)

C#

```
public DateTime DateLoan {get; set;}
```

Código fuente

```
public DateTime DateLoan { get => dateLoan; set => dateLoan = value; }
```

Vea también

Se aplica a: Loan[11]

Loan.IdBook (Propiedad)

C#

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

Código fuente

```
public int IdBook { get => idBook; set => idBook = value; }
```

Vea también

Se aplica a: Loan[11]

Loan.IdLoan (Propiedad)

C#

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

Código fuente

```
public int IdLoan { get => idLoan; set => idLoan = value; }
```

Vea también

Se aplica a: Loan11

Loan.IdUser (Propiedad)

C#

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

Código fuente

```
public int IdUser { get => idUser; set => idUser = value; }
```

Vea también

Se aplica a: Loan11

Loan.IsActive (Propiedad)

C#

```
public bool IsActive {get; set;}
```

Código fuente

```
public bool IsActive { get => isActive; set => isActive = value; }
```

Vea también

Se aplica a: Loan11

Loan.ReturnLoan (Propiedad)

C#

```
public Nullable<DateTime> ReturnLoan {get; set;}
```

Código fuente

```
public DateTime? ReturnLoan { get => returnLoan; set => returnLoan = value; }
```

Vea también

Se aplica a: Loan[11]

Program (Clase)

El objetivo de este programa es generar archivos .json para almacenar en solo aquel archivo los datos de una gestión de Libros, ya sea con las siguientes clases: Autores, Libros, Préstamos y Usuarios. Si bien, esto es básico, está optimizado para utilizar menos recursos posibles. También está hecho a través de .NET 3.5 para compatibilizar con Windows XP y otros equipos viejos que dependan del Terminal (CLI)

object
LibraryXP.Program

C#

```
internal class Program
```

Requisitos

Espacio de nombres: LibraryXP[6]

Ensamblado: LibraryXP (in LibraryXP.exe)

Métodos

Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**)

User (Clase)

El usuario es la clase clave para determinar si el Usuario realmente es responsable con devolver libros y registrar aquellos usuarios frecuentes o nuevos quienes solicitan libros para ser prestados. Tienen un sistema de puntaje para determinar y priorizar aquellos que sean responsables y determinar si es apto para el préstamo.

object
LibraryXP.User

C#

```
internal class User
```

Requisitos

Espacio de nombres: LibraryXP[6]

Ensamblado: LibraryXP (in LibraryXP.exe)

Propiedades

CreditScoreUser[15], IdUser[15], LastNameUser[15], NameUser[16]

Métodos

Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**)

User.CreditScoreUser (Propiedad)

C#

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

Código fuente

```
 public int CreditScoreUser { get => creditScoreUser; set => creditScoreUser =
value; }
```

Vea también

Se aplica a: User[14]

User.IdUser (Propiedad)

C#

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

Código fuente

```
 public int IdUser { get => idUser; set => idUser = value; }
```

Vea también

Se aplica a: User[14]

User.LastNameUser (Propiedad)

C#

```
public string LastNameUser {get; set;}
```

Código fuente

```
 public string LastNameUser { get => lastNameUser; set => lastNameUser = value;
}
```

Vea también

Se aplica a: User[14]

User.NameUser (Propiedad)

C#

```
public string NameUser {get; set;}
```

Código fuente

```
 public string NameUser { get => nameUser; set => nameUser = value; }
```

Vea también

Se aplica a: User[14]

LibraryXP.Controllers (Espacio de nombres)

Clases

AuthorController[17], BookController[20], LoanController[25], UserController[30]

AuthorController (Clase)

object

LibraryXP.Controllers.AuthorController

C#

```
internal class AuthorController
```

Requisitos

Espacio de nombres: LibraryXP.Controllers[17]

Ensamblado: LibraryXP (in LibraryXP.exe)

Métodos

CreateAuthor[17], DeleteAuthor[18], Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetAuthorByID[19], GetAuthors[19], GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**), UpdateAuthor[20]

AuthorController.CreateAuthor (Método)

C#

```
public static void CreateAuthor(
   Author newAuthor
)
```

Parámetros

newAuthor

Código fuente

```
public static void CreateAuthor(Author newAuthor)
{
    var db = JsonHelper.ReadDB();

    newAuthor.IdAuthor = JsonHelper.GetNextID(db.Authors);

    db.Authors.Add(newAuthor);
    JsonHelper.SaveDB(db);
}
```

Vea también

Se aplica a: AuthorController[17]

AuthorController.DeleteAuthor (Método)

Eliminación del Autor. El metodo puede revisar si tiene libros con préstamos activos para evitar borrar su existencia y crear sinsentidos. En caso de que ningún libro tenga préstamo activo, este se elimina, y también eliminando sus libros en el proceso.

C#

```
public static bool DeleteAuthor(
   int id
)
```

Parámetros

id

El ID a borrar

Valor devuelto

Un bool para indicar si todo ha ido bien o no.

Código fuente

```
public static bool DeleteAuthor(int id)
{
    var db = JsonHelper.ReadDB();

    var author = db.Authors.FirstOrDefault(u => u.IdAuthor == id);

    if (author == null)
    {
        return false;
    }
    //Revisa si tiene libros con préstamos activos
    bool hasActiveLoans = db.Books
    .Where(book => book.IdAuthor == id)
    .Any(book =>
        db.Loans.Any(loan =>
            loan.IdBook == book.IdBook &&
            loan.IsActive == true // préstamo activo
        )
    );


    if (hasActiveLoans)
    {
        Console.WriteLine("Uno de sus libros tiene préstamos.");
        Console.ReadLine();
        return false;
    }
```

```
    // Eliminar libros del autor
    db.Books.RemoveAll(b => b.IdAuthor == id);

    //Eliminar autor
    db.Authors.Remove(author);
    JsonHelper.SaveDB(db);

    return true;
}
```

Vea también

Se aplica a: AuthorController[17]

AuthorController.GetAuthorByID (Método)

C#

```
public static Author GetAuthorByID(
   int id
)
```

Parámetros

id

Código fuente

```
public static Author GetAuthorByID(int id)
{
    var db = JsonHelper.ReadDB();
    return db.Authors.FirstOrDefault(u => u.IdAuthor == id);
}
```

Vea también

Se aplica a: AuthorController[17]

AuthorController.GetAuthors (Método)

C#

```
public static List<Author> GetAuthors()
```

Código fuente

```
public static List<Author> GetAuthors()
```

```
{
    var db = JsonHelper.ReadDB();
    return db.Authors;
}
```

Vea también

Se aplica a: AuthorController[17]

AuthorController.UpdateAuthor (Método)

C#

```
public static bool UpdateAuthor(
   int id,
   string nameAuthor
)
```

Parámetros

id

nameAuthor

Código fuente

```
public static bool UpdateAuthor(int id, string nameAuthor)
{
    var db = JsonHelper.ReadDB();
    var author = db.Authors.FirstOrDefault(u => u.IdAuthor == id);

    if (author == null)
    {
        return false;
    }

    author.NameAuthor = nameAuthor;

    JsonHelper.SaveDB(db);

    return true;
}
```

Vea también

Se aplica a: AuthorController[17]

BookController (Clase)

object

LibraryXP.Controllers.BookController

C#

```
internal class BookController
```

Requisitos

Espacio de nombres: LibraryXP.Controllers[17]

Ensamblado: LibraryXP (in LibraryXP.exe)

Métodos

CreateBook[21], DeleteBook[21], Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetBookByID[22], GetBooks[23], GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**), UpdateBook[23]

BookController.CreateBook (Método)

C#

```
public static void CreateBook(
   Book newBook
)
```

Parámetros

newBook

Código fuente

```
public static void CreateBook(Book newBook)
{
    var db = JsonHelper.ReadDB();

    newBook.IdBook = JsonHelper.GetNextID(db.Books);

    db.Books.Add(newBook);
    JsonHelper.SaveDB(db);
}
```

Vea también

Se aplica a: BookController[20]

BookController.DeleteBook (Método)

Solo una eliminación del Libro. En él, la diferencia que se observa, es que primero, verifica si quedan préstamos activos, para evitar crear sinsentido de datos en los préstamos. Si no queda ningún

préstamo activo (Todos han sido cerrados), se borran primero los préstamos (Sin afectar el puntaje de responsabilidad al usuario) y posteriormente elimina el Libro.

C#

```
public static bool DeleteBook(
   int id
)
```

Parámetros

id

ID para eliminar el Libro

Valor devuelto

Un bool para indicar si el método ha sido ejecutado exitosamente.

Código fuente

```
public static bool DeleteBook(int id)
{
    var db = JsonHelper.ReadDB();
    var book = db.Books.FirstOrDefault(u => u.IdBook == id);

    if (book == null)
    {
        return false;
    }

    int activeLoans = LoanController.CountActiveLoans(id);

    if (activeLoans > 0)
    {
        Console.WriteLine("Quedan préstamos pendientes");
        Console.ReadLine();
        return false;
    }
    //Eliminar todos los préstamos relacionados.
    db.Loans.RemoveAll(l => l.IdBook == id);
    //Eliminar el libro en específico
    db.Books.Remove(book);

    JsonHelper.SaveDB(db);

    return true;
}
```

Vea también

Se aplica a: BookController[20]

BookController.GetBookByID (Método)

C#

```
public static Book GetBookByID(
   int id
)
```

Parámetros

id

Código fuente

```
public static Book GetBookByID(int id)
{
    var db = JsonHelper.ReadDB();
    return db.Books.FirstOrDefault(u => u.IdBook == id);
}
```

Vea también

Se aplica a: BookController[20]

BookController.GetBooks (Método)

C#

```
public static List<Book> GetBooks()
```

Código fuente

```
public static List<Book> GetBooks()
{
    var db = JsonHelper.ReadDB();
    return db.Books;
}
```

Vea también

Se aplica a: BookController[20]

BookController.UpdateBook (Método)

Solo una actualización de Libro. Si bien, este método no está condicionado, es recomendable condicionarlo externamente para utilizar y verificar bien los préstamos activos.

C#

```
public static bool UpdateBook(
   int id,
   string codeBook,
```

```
    string nameBook,
    int idAuthor,
    string genre,
    int year,
    int totalStockBook
)
```

Parámetros

id

ID del Libro a editar.

codeBook

Nuevo Código para el Libro

nameBook

Nuevo Nombre para el Libro

idAuthor

Asignación de otro ID de Autor para el Libro

genre

Nuevo Género para el Libro

year

Nuevo Año para el Libro

totalStockBook

Nuevo Stock para el Libro. Condicionado por la cantidad de préstamos vigentes, y debe ser igual o mayor que los préstamos existentes.

Valor devuelto

Un bool para indicar si ha sido exitoso la ejecución del método.

Código fuente

```
 public static bool UpdateBook(int id, string codeBook, string nameBook, int
idAuthor, string genre, int year, int totalStockBook)
 {
     var db = JsonHelper.ReadDB();
     var book = db.Books.FirstOrDefault(u => u.IdBook == id);

     if (book == null)
     {
         return false;
     }

     book.CodeBook = codeBook;
     book.NameBook = nameBook;
     book.IdAuthor = idAuthor;
     book.Genre = genre;
     book.Year = year;
     book.TotalStockBook = totalStockBook;

     JsonHelper.SaveDB(db);

     return true;
```

```
}
```

Vea también

Se aplica a: BookController[20]

LoanController (Clase)

object
LibraryXP.Controllers.LoanController

C#

```
internal class LoanController
```

Requisitos

Espacio de nombres: LibraryXP.Controllers[17]

Ensamblado: LibraryXP (in LibraryXP.exe)

Métodos

CountActiveLoans[25], CreateLoan[26], DeleteLoan[26], Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetLoanByID[27], GetLoans[28], GetType (Se hereda de **object**), HasActiveLoanByUser[28], MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**), UpdateLoan[29]

LoanController.CountActiveLoans (Método)

Método para contar cuantos préstamos tiene el libro. Esto evita sobrepasarse en el stock.

C#

```
public static int CountActiveLoans(
   int idBook
)
```

Parámetros

idBook

El ID del libro.

Valor devuelto

Un Int en el que muestra cuántos préstamos tiene activos actualmente.

Código fuente

```
public static int CountActiveLoans(int idBook)
{
    var db = JsonHelper.ReadDB();
```

```
    return db.Loans.Count(l =>
        l.IdBook == idBook &&
        l.IsActive == true // activo
    );
}
```

Vea también

Se aplica a: LoanController[25]

LoanController.CreateLoan (Método)

C#

```
public static void CreateLoan(
   Loan newLoan
)
```

Parámetros

newLoan

Código fuente

```
public static void CreateLoan(Loan newLoan)
{
    var db = JsonHelper.ReadDB();

    newLoan.IdLoan = JsonHelper.GetNextID(db.Loans);

    db.Loans.Add(newLoan);
    JsonHelper.SaveDB(db);
}
```

Vea también

Se aplica a: LoanController[25]

LoanController.DeleteLoan (Método)

C#

```
public static bool DeleteLoan(
   int id
)
```

Parámetros

id

Código fuente

```
public static bool DeleteLoan(int id)
{
    var db = JsonHelper.ReadDB();
    var loan = db.Loans.FirstOrDefault(u => u.IdLoan == id);

    if (loan == null)
    {
        return false;
    }

    db.Loans.Remove(loan);

    JsonHelper.SaveDB(db);

    return true;
}
```

Vea también

Se aplica a: LoanController[25]

LoanController.GetLoanByID (Método)

C#

```
public static Loan GetLoanByID(
   int id
)
```

Parámetros

id

Código fuente

```
public static Loan GetLoanByID(int id)
{
    var db = JsonHelper.ReadDB();
    return db.Loans.FirstOrDefault(u => u.IdLoan == id);
}
```

Vea también

Se aplica a: LoanController[25]

LoanController.GetLoans (Método)

C#

```
public static List<Loan> GetLoans()
```

Código fuente

```
public static List<Loan> GetLoans()
{
    var db = JsonHelper.ReadDB();
    return db.Loans;
}
```

Vea también

Se aplica a: LoanController[25]

LoanController.HasActiveLoanByUser (Método)

Revisa si el usuario en cuestión tiene Préstamos activos.

C#

```
public static bool HasActiveLoanByUser(
   int id
)
```

Parámetros

id

El ID del usuario para revisar préstmos

Valor devuelto

Si tiene préstmos, mostrará true.

Código fuente

```
public static bool HasActiveLoanByUser(int id) {
    var db = JsonHelper.ReadDB();

    return db.Loans.Any(l =>
        l.IdUser == id &&
        l.IsActive == true // activo
    );
}
```

Vea también

Se aplica a: LoanController[25]

LoanController.UpdateLoan (Método)

Si el sistema de puntaje de créditos es funcional, entonces podríamos hacer que suba y baje puntaje para el historial de préstamos. Lógica sería así: Si aún está a la fecha, al devolver se le sube puntaje, pudiendo dar retroceso en caso de error. Si ya se pasó la fecha, solo se puede cerrar el caso y tener puntaje negativo.

C#

```
public static bool UpdateLoan(
   int id
)
```

Parámetros

id

Un ID para poder actualizar el Préstamo

Valor devuelto

Un bool para indicar si todo ha ido bien o no.

Código fuente

```
 public static bool UpdateLoan(int id)
 {
     var db = JsonHelper.ReadDB();
     var loan = db.Loans.FirstOrDefault(u => u.IdLoan == id);

     if (loan == null)
     {
         return false;
     }

     int scoreMeter = 15;
     int idUser = loan.IdUser;
     if (loan.IsActive == false && loan.ReturnLoan >= DateTime.Now)
     {
         //Si el préstamo no está activo y la fecha de regreso todavía no ha
sucedido hoy.
         if (HasActiveLoanByUser(idUser) == true)
         {
             Console.WriteLine("El usuario ya tiene un préstamo activo.");
             Console.ReadLine();
             return false;
         }
         UserController.UpdateScoreByUser(db, idUser, -(scoreMeter));
         Console.WriteLine("Se le ha bajado {0} puntos al Usuario", scoreMeter);
         Console.ReadLine();
         loan.IsActive = true;
     }
     else {
         //Caso de que esté activo
         if (loan.ReturnLoan >= DateTime.Now)
         {
             UserController.UpdateScoreByUser(db, idUser, scoreMeter);
             Console.WriteLine("Se le ha subido {0} puntos al Usuario",
```

```
scoreMeter);
                Console.ReadLine();
            }
            else {
                UserController.UpdateScoreByUser(db, idUser, -(scoreMeter));
                Console.WriteLine("Se le ha bajado {0} puntos al Usuario",
scoreMeter);
                Console.ReadLine();
            }
            loan.IsActive = false;
        }

        //Si el sistema de puntaje de créditos es funcional, entonces podríamos
hacer que suba y baje puntaje para
        //El historial de préstamos.
        //Lógica sería así: Si aún está a la fecha, al devolver se le sube puntaje,
pudiendo dar retroceso en caso de error.
        //Si ya se pasó la fecha, solo se puede cerrar el caso y tener puntaje
negativo.

        JsonHelper.SaveDB(db);

        return true;
    }
```

Vea también

Se aplica a: LoanController[25]

UserController (Clase)

object

LibraryXP.Controllers.UserController

C#

```
internal class UserController
```

Requisitos

Espacio de nombres: LibraryXP.Controllers[17]

Ensamblado: LibraryXP (in LibraryXP.exe)

Métodos

CreateUser[31], DeleteUser[31], Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetType (Se hereda de **object**), GetUserByID[32], GetUsers[32], MemberwiseClone (Se hereda de **object**), ReferenceEquals (Se hereda de **object**), ToString (Se hereda de **object**), UpdateScoreByUser[33], UpdateUser[34]

UserController.CreateUser (Método)

C#

```
public static void CreateUser(
   User newUser
)
```

Parámetros

newUser

Código fuente

```
public static void CreateUser(User newUser)
{
    var db = JsonHelper.ReadDB();

    newUser.IdUser = JsonHelper.GetNextID(db.Users);

    db.Users.Add(newUser);
    JsonHelper.SaveDB(db);
}
```

Vea también

Se aplica a: UserController[30]

UserController.DeleteUser (Método)

Elimina todos los préstamos relacionados y el usuario seleccionado

C#

```
public static bool DeleteUser(
   int id
)
```

Parámetros

id

Valor devuelto

Un bool para indicar si todo ha ido bien o no.

Código fuente

```
public static bool DeleteUser(int id)
{
    var db = JsonHelper.ReadDB();
    var user = db.Users.FirstOrDefault(u => u.IdUser == id);
```

```
    if (user == null)
    {
        return false;
    }

    //Eliminar todos los préstamos relacionados.
    db.Loans.RemoveAll(l => l.IdUser == id);
    //Eliminar solo el usuario
    db.Users.Remove(user);

    JsonHelper.SaveDB(db);

    return true;
}
```

Vea también

Se aplica a: UserController[30]

UserController.GetUserByID (Método)

C#

```
public static User GetUserByID(
   int id
)
```

Parámetros

id

Código fuente

```
public static User GetUserByID(int id)
{
    var db = JsonHelper.ReadDB();
    return db.Users.FirstOrDefault(u => u.IdUser == id);
}
```

Vea también

Se aplica a: UserController[30]

UserController.GetUsers (Método)

C#

```
public static List<User> GetUsers()
```

Código fuente

```
public static List<User> GetUsers()
{
    var db = JsonHelper.ReadDB();
    return db.Users;
}
```

Vea también

Se aplica a: UserController[30]

UserController.UpdateScoreByUser (Método)

Método para hacerlo en base a un DB ya existente. No hay necesidad de utilizar sin DB debido que solo exclusivamente lo hace dentro de los Préstamos.

C#

```
public static bool UpdateScoreByUser(
   DataBase db,
   int id,
   int newScore
)
```

Parámetros

db

La Base de datos abierta anteriormente para poder aplicar cambio

id

El ID del Usuario para cambiar su puntaje (O el historial crediticio)

newScore

El puntaje nuevo que se le suma o reste.

Valor devuelto

Un bool para indicar si todo ha ido bien o no.

Código fuente

```
public static bool UpdateScoreByUser(DataBase db, int id, int newScore)
{
    var user = db.Users.FirstOrDefault(u => u.IdUser == id);
    if (user == null)
    {
        Console.WriteLine("No se ha encontrado el usuario.");
        return false;
    }

    user.CreditScoreUser += newScore;

    return true;
```

```
}
```

Vea también

Se aplica a: UserController[30]

UserController.UpdateUser (Método)

El crédito de préstamos se mantiene intacto al cambiar elementos de la identidad del usuario. (Sigue siendo la misma persona)

C#

```
public static bool UpdateUser(
   int id,
   string nameUser,
   string lastNameUser
)
```

Parámetros

id

El ID para actualizar el Usuario

nameUser

Nuevo Nombre

lastNameUser

Nuevo Apellido

Valor devuelto

Un bool para indicar si todo ha ido bien o no.

Código fuente

```
public static bool UpdateUser(int id, string nameUser, string lastNameUser)
{
    var db = JsonHelper.ReadDB();
    var user = db.Users.FirstOrDefault(u => u.IdUser == id);

    if (user == null)
    {
        return false;
    }

    user.NameUser = nameUser;
    user.LastNameUser = lastNameUser;

    JsonHelper.SaveDB(db);

    return true;
}
```

Vea también

Se aplica a: UserController30

LibraryXP.Data (Espacio de nombres)

Clases

JsonHelper[36]

JsonHelper (Clase)

El pilar fundamental de la conexión de Base de Datos con la conexión de archivo .json para poder desarrollar el proceso. En él, se almacenan todo lo necesario como para abrir para su lectura y guardar para su escritura. Así como la detección de las clases necesarias ya hechas en la DataBase.cs

object
LibraryXP.Data.JsonHelper

C#

```
internal class JsonHelper
```

Requisitos

Espacio de nombres: LibraryXP.Data[36]

Ensamblado: LibraryXP (in LibraryXP.exe)

Métodos

Equals (Se hereda de **object**), Finalize (Se hereda de **object**), GetHashCode (Se hereda de **object**), GetNextID<T>[36], GetType (Se hereda de **object**), MemberwiseClone (Se hereda de **object**), ReadDB[37], ReferenceEquals (Se hereda de **object**), SaveDB[38], ToString (Se hereda de **object**)

JsonHelper.GetNextID<T> (Método)

Este método captura el siguiente ID dentro de la clase. Solo verifica lo que es el último número y le agrega un 1.

C#

```
public static int GetNextID<T>(
   List<T> list
)
   where T : class
```

Parámetros de tipo

T

La Clase a detectar.

Parámetros

list

La lista que incluye la clase verificando el último número ID existente

Valor devuelto

Un int que encuentra el último ID + 1

Excepciones

Tipo de excepción	Condición
Exception	Lanza si la clase no existe.

Código fuente

```
public static int GetNextID<T>(List<T> list) where T : class
{
    if (typeof(T) == typeof(User))
    {
        var users = list.Cast<User>().ToList();
        return users.Count == 0 ? 1 : users.Max(u => u.IdUser) + 1;
    }
    else if (typeof(T) == typeof(Loan))
    {
        var loans = list.Cast<Loan>().ToList();
        return loans.Count == 0 ? 1 : loans.Max(u => u.IdLoan) + 1;
    }
    else if (typeof(T) == typeof(Book))
    {
        var books = list.Cast<Book>().ToList();
        return books.Count == 0 ? 1 : books.Max(u => u.IdBook) + 1;
    }
    else if (typeof(T) == typeof(Author))
    {
        var authors = list.Cast<Author>().ToList();
        return authors.Count == 0 ? 1 : authors.Max(u => u.IdAuthor) + 1;
    }

    throw new Exception("Tipo no soportado para auto-increment de ID.");
}
```

Vea también

Se aplica a: JsonHelper[36]

JsonHelper.ReadDB (Método)

Abre el archivo .json y lo lee.

C#

```
public static DataBase ReadDB()
```

Valor devuelto

Devuelve toda la base de datos. (Solo para proyectos pequeños es válido)

Código fuente

```
public static DataBase ReadDB()
{
    if (!File.Exists(filePath))
        return new DataBase();

    string json = File.ReadAllText(filePath);
    return JsonConvert.DeserializeObject<DataBase>(json) ?? new DataBase();
}
```

Vea también

Se aplica a: JsonHelper[36]

JsonHelper.SaveDB (Método)

Guarda todo lo modificado en la base de datos. Es recomendable abrir una sola vez para que haga sentido el guardado de la Base de datos.

C#

```
public static void SaveDB(
   DataBase dataBase
)
```

Parámetros

dataBase

La base de datos abierta anteriormente con ReadDB()

Código fuente

```
public static void SaveDB(DataBase dataBase)
{
    string json = JsonConvert.SerializeObject(dataBase, Formatting.Indented);
    File.WriteAllText(filePath, json);
}
```

Vea también

Se aplica a: JsonHelper[36]

Ver más proyectos

Para ver más proyectos, puede visitar al GitHub de HF1NaCl en el que se muestran más desarrollos hechos en distintos programas, lenguajes de programación y Frameworks. No dude en consultar si desea un desarrollador o programador.

Autor

Enrique Fu Orozco

Versión

0.1

Revisión

Demo

Índice