Universidad Mayor de San Andres
Facultad de Ciencias Puras y Naturales
Carrera de Informática
lab-121
Lic. Cecilia E. Tarquino P.

PROYECTO: Seguimiento de prestamos de material

Marco Antonio Vino Chipana
CI 9111299 L.P.
29 de noviembre de 2018

1 Introducción

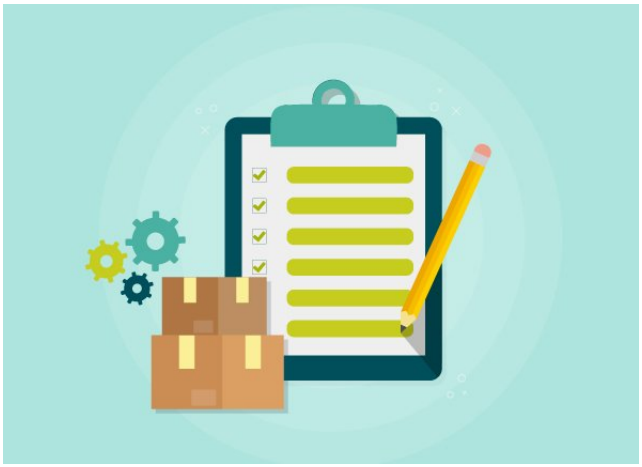
Cuando se tiene que llevar el registro de los prestamos que se realizan en alguna institución, nos encontramos con el problema de que este dbe ser oportuno, rápido y eficiente. En este caso de estudio lo que vamos a realizar es el modelado y programación de un sistema de control de prestamos. Este sistema se desarollará utilizando en entorno gráfico que nos facilitan las aplicaciones de Windows Forms. Se utilizará el lenguaje de programación C#, dentro del entorno de desarrollo SharpDevelop en su versión 5.1.

De la misa forma se utilizan los conceptos de la programación orientada a objetos, tales como al herencia, la sobrecarga, las clases y objetos, la agregación , la composición y la persistencia de archivos.

2 Análisis del ambiente y el problema

2.1 Estado del arte

Como primer recurso se utilizará un inventario, el cual es una relación detallada, ordenada y valorada de los elementos que componen el patrimonio de una empresa o persona en un momento determinado.



El segundo elemento serán los usuarios, en informática un usuario es una persona que utiliza una aplicación o sistema a través del uso de una computadora, la cual para su utilización requiere cierto nivel de experticia.

Cada uno de los recursos que se podrán utilizar serán considerados ítems, Consideraremos como un ítem a cada uno de los recursos físicamente separables que puedan ser prestados individualmente, como ser, un tubo de ensayo, un mortero, etc.

2.2 Problemática

Debemos llevar un registro del historial de los préstamos a los usuarios para poder saber su confiabilidad y si es que tienen o no sanciones por incumplimiento de reglas
Alertar al administrador cuando los materiales no han sido devueltos en los plazos establecidos para poder hacer in seguimiento.
Saber en tiempo real la disponibilidad de recursos para poder solicitarlos en préstamo.
Tener un estimado de cuando debería estar disponible el recurso que necesitamos para poder hacer una planificación.
Averiguar la rotación de los elementos para ver la demanda de recursos y utilizar esta información para futuras adquisiciones, generando preferencias por aquellos elementos que más se hayan encontrado agotados en varias situaciones.

3 Objetivos

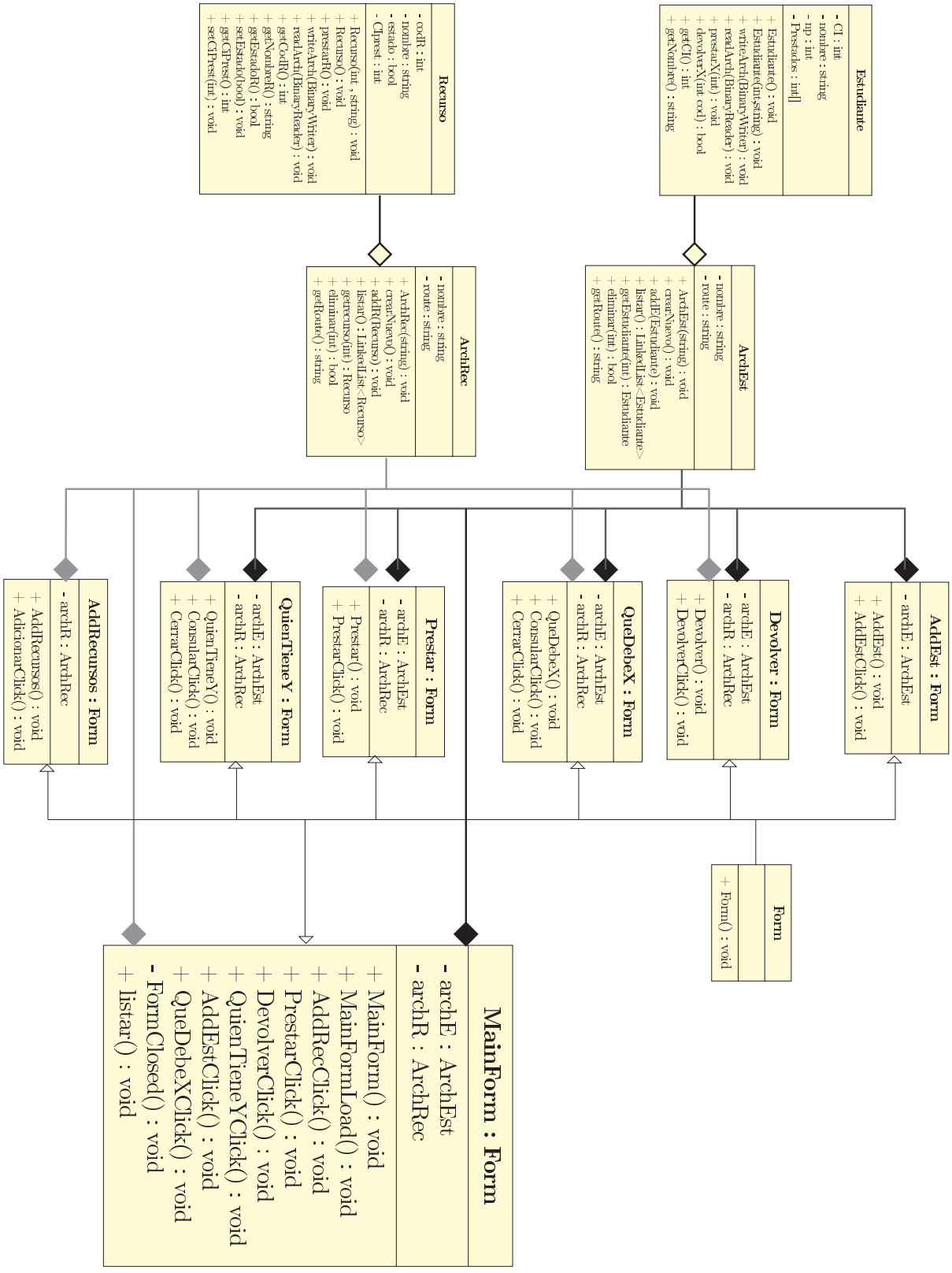
3.1 Objetivo general

Realizar seguimiento al movimiento de los recursos susceptibles a préstamo dentro de una institución.

3.2 Objetivos específicos

- Registrar todos los préstamos de material
- Identificar a todos los usuarios que pueden prestarse material
- Registrar las devoluciones del material
- Mostrar en tiempo real la disponibilidad de materiales
- Consultar quien tiene prestado un material con identificacion x.
- Consultar que recursos debe el estudiante X.

4 Diagrama de clases



5 Descripción de las clases

5.1 Clases más importantes del desarrollo

En esta sección se describirán las clases propias del desarrollo más allá del entorno gráfico que fueron utilizadas durante el desarrollo para cumplir nuestros objetivos.

5.1.1 Estudiante

Estudiante
- CI : int - nombre : string - np : int - Prestados: int[]
+ Estudiante() : void + Estudiante(int,string) : void + writeArch(BinaryWriter) : void + readArch(BinaryReader) : void + prestarX(int) : void + devolverX(int cod) : bool + getCI() : int + getNombre() : string

Esta clase guarda la información de cada estudiante, con su *nombre* , su *C.I.* además de todos los recursos que se ha prestado.
El método **prestarX(int)** agrega un recurso a la lista de presados del estudiante. El método **devolverX(int)** elimina de la lista de recursos prestados al estudiante, aquel que tenga el código que se pasa como argumento, retorna verdadero si se pudo realizar la devolución y falso si hubo algún fallo.

5.1.2 Archivo Estudiante

ArchEst
- nombre : string - route : string
+ ArchEst(string) : void + crearNuevo() : void + addE(Estudiante) : void + listar() : LinkedList¡Estudiante¿ + getEstudiante(int): Estudiante + eliminar(int) : bool + getRoute() : string

Es la clase que realiza el manejo del archivo donde se guardan todos los estudiantes. Como atributos tiene el *nombre* del archivo, además de su *ruta*. El método **addE(Estudiante)**, agrega al estudiante que pasamos como argumento al final del archivo. La función **listar()**

nos permite extraer una lista que contiene a todos los estudiantes guardados en el archivo. la función **getEstudiente(int)** extrae al estudiante cuyo carnet se pasó como argumento. El método **eliminar(int)** borra del archivo al estudiante cuyo carnet se haya ingresado como argumento.

5.1.3 Recurso

Recurso
- codR : int - nombre : string - estado : bool - CIprest : int
+ Recurso(int , string) : void + Recurso() : void + prestarR() : void + writeArch(BinaryWriter) : void + readArch(BinaryReader) : void + getCodR() : int + getNombreR() : string + getEstadoR() : bool + setEstado(bool) : void + getCiPrest() : int + setCiPrest(int) : void

La clase recurso se utiliza para guardar la información de un elemento suceptible a prestamo dentro de nuestro sistema, tiene como atributos un identificador único denotado por *codR*, además de un *nombre*, un *estado* (que es verdadero si está en préstamo y falso si no), además de un *CiPrest*, para saber a quien se le prestó el recurso.
El método **prestar()** cambia el estado de un recurso a verdadero, el método **setCiPrest(int)** pone como persona a quien se prestó el recurso, aquella que tenga el carnet ingresado. De la misma forma **setEstado(bool)** nos sirve para cambiar el estado del recurso.

5.1.4 Archivo Recurso

ArchRec
- nombre : string - route : string
+ ArchRec(string) : void + crearNuevo() : void + addR(Recurso) : void + listar() : LinkedList¡Recurso¿ + getrecurso(int): Recurso + eliminar(int) : bool + getRoute() : string

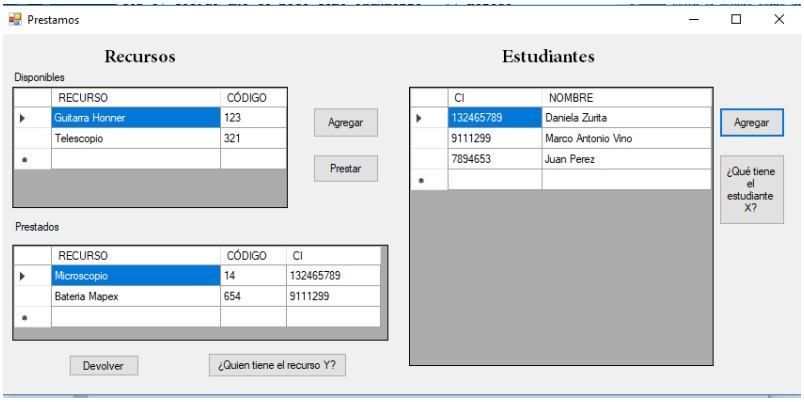
Es la clase que realiza el manejo del archivo donde se guardan todos los Recursos. Como atributos tiene el *nombre* del archivo, además de su *ruta*. El método **addR(Recurso)**, agrega un nuevo Recurso, el que pasamos como argumento al final del archivo, por defecto setea el estado como disponible. La función **listar()** nos permite extraer una lista que contiene a todos los recursos guardados en el archivo. la función **getrecurso(int)** extrae al recurso con el código que se pasó como argumento. El método **eliminar(int)** borra del archivo al recurso cuyo código se haya ingresado como argumento.

5.2 Clases de los formularios del entorno gráfico

Se describirán como funcionan los diferentes formularios desarrollados dentro del entorno gráfico del proyecto.

5.3 Formulario principal

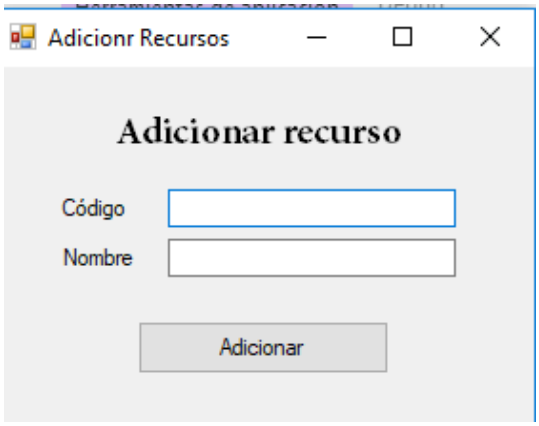
MainForm : Form
- archE : ArchEst - archR : ArchRec
+ MainForm() : void + MainFormLoad() : void + AddRecClick() : void + PrestarClick() : void + DevolverClick() : void + QuienTieneYClick() : void + AddEstClick() : void + QueDebeXClick() : void - FormClosed() : void + listar() : void



El formulario principal es el que se abre al inicio del sistema, el mismo tiene como atributos a los archivos tanto de *estudiantes* como de *recursos*. Su función principal es mostrar el estado de las diferentes listas, y esto se lleva a cabo del método **listar()**, cada uno de los botones tiene un método que abre otra ventana para la interacción, por ejemplor **AddRecClick()** es un método que abré el formulario que nos permite agregar un nuevo recurso. De la misma forma **PrestarClick()** abre la ventana para registrar un prestamo. El método **FormClosed()** nos sirve para actualizar la vista de elementos cada vez que se cierra una ventana de introducción de datos. Los métodos **QuienTieneY()** y **QueDebeX()** abren ventanas de consulta para cada una de sus tareas.

5.3.1 Agregar Recurso

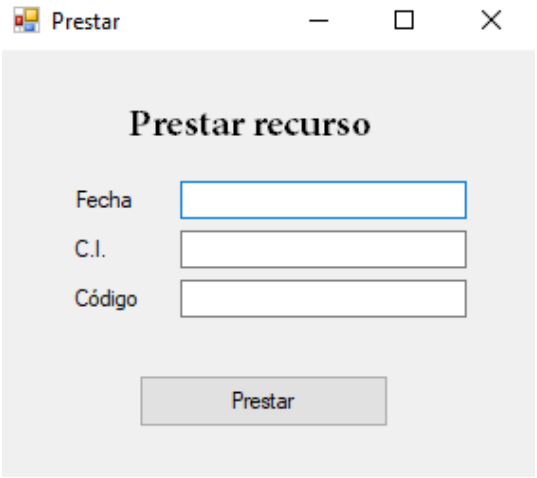
AddRecursos : Form
- archR : ArchRec
+ AddRecursos() : void + AdicionarClick() : void



Esta ventana nos permite agregar un recurso más a la lista de recursos guardaas en el archivo **ArchRec**. Esta funcionalidad es llevada a cabo por el método **AdicionarClick()**, que se ejecuta cuando le damos click al boton adicionar. Para poder ser llevado a cabo realiza las comprobaciones necesarias de que el código sea un número válido.

5.3.2 Prestar recurso

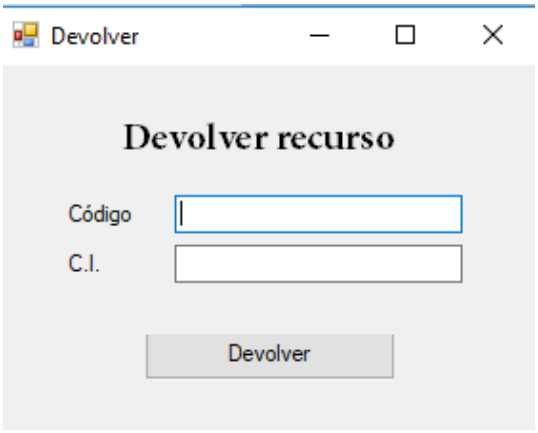
Prestar : Form
- archE : ArchEst - archR : ArchRec
+ Prestar() : void + PrestarClick() : void



Tiene como atributos tanto los archivos de estudiante como los de recursos para poder registrar los cambios. Esta ventana nos sirve para registrar los préstamos que se realizan en el sistema. Tiene 3 elementos a ser llenados, la fecha, el ci de la persona a la que se le prestará y el cod del recurso que se presta. Dentro del método **PrestarClick()** se realizan las comprobaciones para que los datos ingresados sean válidos. Se cambia el estado del recurso ingresado, además que se le asigna al atributo CiPrest el carnet de la persona que se presto el recurso. Al mismo tiempo, Al estudiante con el ci ingresado se le agrega el recurso a articulos prestados siempre y cuando este disponible.

5.3.3 Devolver recurso

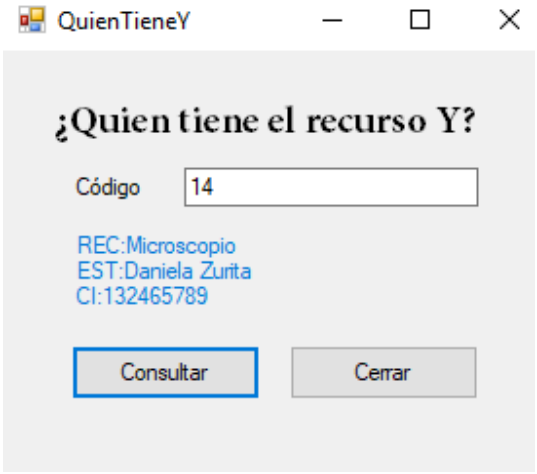
Devolver : Form
- archE : ArchEst - archR : ArchRec
+ Devolver() : void + DevolverClick() : void



Tiene como atributos tanto los archivos de estudiante como los de recursos para poder registrar los cambios. El método **DevolverClick()**, verifica que los datos ingresados sean válidos y cambia el estado del recurso con el cod ingresado, además que eliminar de las lista de prestados del estudiante con el CI ingresado, el recurso que esta devolviendo.

5.3.4 ¿Quién tiene el recurso Y?

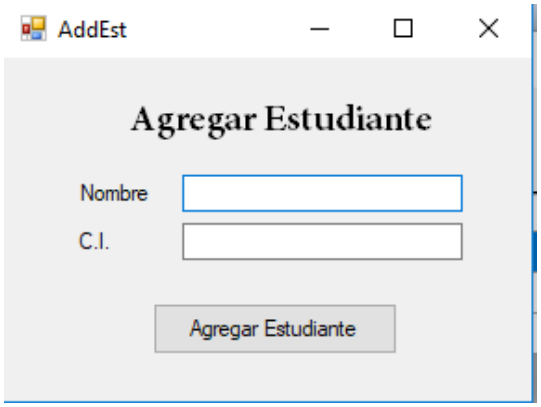
QuienTieneY : Form
- archE : ArchEst - archR : ArchRec
+ QuienTieneY() : void + ConsularClick() : void + CerrarClick() : void



Tiene como atributos tanto los archivos de estudiante como los de recursos para consultar los valores. El método **ConsultarClick()** Busca si el recurso con el código ingresado ha sido prestado, y si ha sido prestado, y si está prestado extrae el CIPrest, y busca en el *ArchEst* los datos del estudiante que tiene el recurso consultado.
El método **CerrarClick()**, cierra el formulario actual y envía la señal de actualizar al formulario principal.

5.3.5 Agregar Estudiante

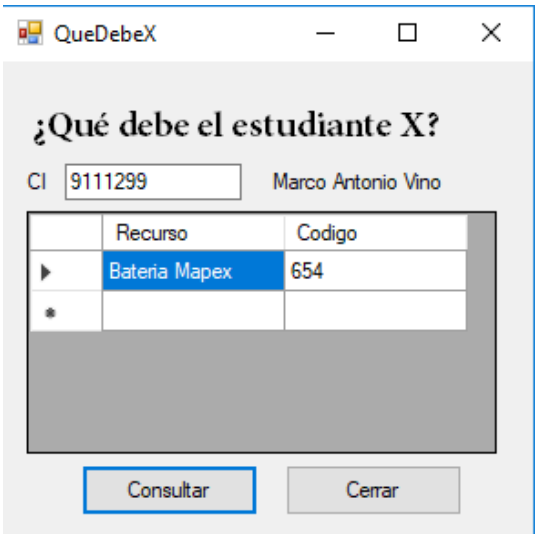
AddEst : Form
- archE : ArchEst
+ AddEst() : void + AddEstClick() : void



Esta ventana nos permite agregar un estudiante más a la lista de estudiantes guardos en el archivo **ArchEst**. Esta funcionalidad es llevada a cabo por el método **AdicionarClick()**, que se ejecuta cuando le damos click al boton adicionar. Para poder ser llevado a cabo realiza las comprobaciones necesarias de que el CI sea un número válido.

5.3.6 ¿Qué debe el estudiante X?

QueDebeX : Form
- archE : ArchEst - archR : ArchRec
+ QueDebeX() : void + ConsularClick() : void + CerrarClick() : void



Tiene como atributos tanto los archivos de estudiante como los de recursos para consultar los valores. En método **ConsultarClick()**, busca el valor de CI en el *archE*, y si lo encuentra pasa a buscar dentro del *archR* todos aquellos recursos que se hayan prestado con ese CI y los va mostrando en el data grid view.
El método **CerrarClick()**, cierra el formulario actual y envía la señal de actualizar al formulario principal.

6 Herramientas y librerias utilizadas

6.1 C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.
Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.
El nombre C Sharp fue inspirado por el signo '#' que se compone de cuatro signos '+' pegados.
las Librerias utilizadas en el desarrollo del proyecto se detallan a continuación.

- **System** contiene clases fundamentales y clases básicas que definen tipos de datos de valor y referencia comúnmente utilizados, eventos y controladores de eventos, interfaces, atributos y excepciones de procesamiento.
- **System.Collections.Generic** contiene interfaces y clases que definen colecciones genéricas, que permiten a los usuarios crear colecciones tipificadas con fuerza que proporcionan una seguridad y rendimiento de tipo mejor que las colecciones tipográficas no genéricas.
- **System.Windows.Forms** contiene clases para crear aplicaciones basadas en Windows que aprovechan al máximo las características de la interfaz de usuario ricas disponibles en el sistema operativo Microsoft Windows.

6.2 SharpDevelop 5.1

SharpDevelop es un entorno de desarrollo integrado libre desarrollado por *ICSharpCode Team*, con su última version 5.1 lanzada el 14 de abril de 2016.
Para el completado automático de código, la aplicación incorpora sus propios analizadores sintácticos. Además que integra la funcionalidad de trabajar en el entorno de Windows Forms.

6.3 LaTeX

LaTeXes un sistema de composición de textos de código abierto, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, es usado de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas.
Específicamente las librrerias que se utilizaron en este proyecto son:

- **listingsutf8** nos permite mostrar el código fuente en un documento, directamente desde el archivo que contiene el código fuente.
- **tikz-uml** que nos permite desarrollar los diagramas UML de manera sencilla y directa.

7 Código fuente

7.1 Clases más importantes

Clase Estudiante:

```
1  using System;
2  using System.IO;
3
4  namespace Prestamos
5  {
6    public class Estudiante
7    {
8      private int CI;
9      private string nombre;
10     private int np =0;
11     private int[] Prestados = new int[100];
12
13     public Estudiante()
14     { np =0;
15     }
16
17     public Estudiante( int ci, string n)
18     {
19       CI= ci;
20       nombre = n;
21     }
22
23     public void writeArch(BinaryWriter escritor)
24     {
25       escritor.Write(CI);
26       escritor.Write(nombre);
27       escritor.Write(np);
28       for (int i =0 ; i<np; i++)
29       {
30         escritor.Write(Prestados[i]);
31       }
32     }
33
34
35     public void readArch(BinaryReader lector)
36     {
37       CI = lector.ReadInt32();
38       nombre = lector.ReadString();
39       np = lector.ReadInt32();
40       for (int i =0 ; i<np; i++)
41       {
42         Prestados[i] = lector.ReadInt32();
43       }
44     }
45
46     public void prestarX(int cod)
47     {
48       Prestados[np++] = cod;
49     }
50
```

```
51      public bool devolverX(int cod)
52      {
53        for (int i=0 ; i < np; i++)
54        {
55          if (Prestados[i] == cod)
56          {
57            Prestados[i] = Prestados[np--];
58            return true;
59          }
60        }
61        return false;
62      }
63
64      public int getCI(){return CI;}
65      public string getNombre(){return nombre;}
66      ~Estudiante(){}
67    }
68  }
```

Clase Archivo Estudiante:

```
1  using System;
2  using System.IO;
3  using System.Collections.Generic;
4
5  namespace Prestamos
6  {
7    public class ArchEst
8    {
9      private string nombre;
10       private string route;
11     public ArchEst( string na ) {
12       nombre = na;
13           route = "..\\";
14           crearNuevo();
15       }
16
17       /// Metodo que crea un nuevo archivo borrando si existia uno
18     /// anterior.
19     public void crearNuevo() {
20       if( !System.IO.File.Exists(route+nombre) ){
21             File.Create(route + nombre).Close();
22           }
23       else
24         Console.WriteLine("El archivo no existe.");
25     }
26
27
28     /// Adiciona un producto al final del archivo.
29     public void addE(Estudiante Ei) {
30       // Abrimos el archivo o se crea un nuevo archivo si no existe
31       Stream archE = File.Open(route + nombre, FileMode.Append);
32       BinaryWriter escritor = new BinaryWriter(archE);
33       try {
34           Ei.writeArch(escritor);
35       }
36       catch( Exception ) {
```

```
37          Console.WriteLine("Fallo en adicionar el objeto !!!");
38        }
39        finally {
40          archE.Close();
41        }
42      }
43
44      public LinkedList<Estudiante> listar() {
45        // Abrimos el archivo o se crea un nuevo archivo si no existe
46        Stream archE = File.Open(route+nombre, FileMode.Open);
47        BinaryReader lector = new BinaryReader(archE);
48            var EstList= new LinkedList<Estudiante> {};
49            try
50            {
51
52          while( true ) {
53                    var Ei= new Estudiante();
54                    Ei.readArch(lector);
55                    EstList.AddLast(Ei);
56          }
57        }
58        catch( Exception ) {
59          Console.WriteLine("Fin de archivo ...");
60        }
61        finally {
62          archE.Close();
63        }
64            return EstList;
65      }
66
67
68        public Estudiante getEstudante(int ci)
69        {
70            Stream archE = File.Open(route + nombre, FileMode.Open);
71            BinaryReader lector = new BinaryReader(archE);
72            try
73            {
74                while (true)
75                {
76                    var Ei = new Estudiante();
77                    Ei.readArch(lector);
78                    if (Ei.getCI() == ci) {
79                        archE.Close();
80                        return Ei;
81                    }
82                }
83            }
84            catch (Exception)
85            {
86                Console.WriteLine("Fin de archivo ...");
87            }
88            finally
89            {
90                archE.Close();
91            }
92            return null;
```

```
93        }
94
95
96        public bool eliminar( int codigo ) {
97        bool sw = false;
98        try {
99          // Abrimos el archivo o se crea un nuevo archivo si no existe
100         Stream archp = File.Open(route + nombre, FileMode.Open);
101         Stream archTemporal = File.Open(route+"temp.dat", FileMode.
      OpenOrCreate);
102         BinaryReader lector = new BinaryReader(archp);
103         BinaryWriter escribeTemp = new BinaryWriter(archTemporal);
104         Estudiante Ei = new Estudiante();
105
106         try {
107           while( true ) {
108             Ei.readArch(lector);
109             if( Ei.getCI() != codigo ) {
110               Ei.writeArch(escribeTemp);
111             }
112             else {
113               sw = true;
114             }
115           }
116         }
117         catch( Exception ) {
118           // No hace nada.
119         }
120         finally {
121           archp.Close();
122           archTemporal.Close();
123           File.Replace(route+"temp.dat",route+nombre,nombre + ".bak");
124         }
125       }
126       catch( Exception ) {
127         Console.WriteLine("El archivo no se puede acceder !!!");
128       }
129       return sw;
130     }
131
132     public string getRoute(){return route + nombre;}
133   }
134 }
```

Clase Recurso:

```
1  using System;
2  using System.IO;
3
4  namespace Prestamos
5  {
6    public class Recurso
7    {
8      private int codR;
9      private string nombre;
10     private bool estado;
11     private int CIprest;
```

```
12      // true para prestado, false para disponible
13      public Recurso(int c, string n )
14      {
15        codR = c;
16        nombre = n;
17        estado = false;
18        CIprest = 0;
19      }
20
21      public Recurso(){}
22
23      public void prestarR()
24      {
25        estado = false;
26      }
27
28      public void writeArch(BinaryWriter escritor)
29      {
30        escritor.Write(codR);
31        escritor.Write(nombre);
32        escritor.Write(estado);
33        escritor.Write(CIprest);
34      }
35
36      public void readArch(BinaryReader lector)
37      {
38        codR = lector.ReadInt32();
39        nombre = lector.ReadString();
40        estado = lector.ReadBoolean();
41        CIprest = lector.ReadInt32();
42      }
43
44      public int getCodR(){return codR;}
45      public string getNombreR(){return nombre;}
46      public bool getEstadoR(){return estado;}
47      public void setEstado(bool x){estado = x;}
48      public int getCiPrest(){return CIprest;}
49      public void setCiPrest(int x){CIprest = x;}
50      ~Recurso(){}
51    }
52  }
```

Clase Archivo Recurso:

```
1  using System;
2  using System.IO;
3  using System.Collections.Generic;
4
5  namespace Prestamos
6  {
7    public class ArchRec
8    {
9      private string nombre;
10       private string route;
11     public ArchRec( string na ) {
12       nombre = na;
13           route = "..\\";
```

```
14            crearNuevo();
15        }
16
17        /// Metodo que crea un nuevo archivo borrando si existia uno
18      /// anterior.
19      public void crearNuevo() {
20        if( !System.IO.File.Exists(route+nombre) ){
21                File.Create(route + nombre).Close();
22            }
23        else
24          Console.WriteLine("El archivo no existe.");
25      }
26
27      /// Adiciona un producto al final del archivo.
28      public void addR(Recurso Ri) {
29        Stream archR = File.Open(route + nombre, FileMode.Append);
30        BinaryWriter escritor = new BinaryWriter(archR);
31        try {
32            Ri.writeArch(escritor);
33        }
34        catch( Exception ) {
35          Console.WriteLine("Fallo en adicionar el objeto !!!");
36        }
37        finally {
38          archR.Close();
39        }
40      }
41
42
43
44      public LinkedList<Recurso> listar() {
45        // Abrimos el archivo o se crea un nuevo archivo si no existe
46        Stream archR = File.Open(route+nombre, FileMode.Open);
47        BinaryReader lector = new BinaryReader(archR);
48            var RecList= new LinkedList<Recurso> {};
49            try
50            {
51
52          while( true ) {
53                    var Ri= new Recurso();
54                    Ri.readArch(lector);
55                    RecList.AddLast(Ri);
56          }
57        }
58        catch( Exception ) {
59          Console.WriteLine("Fin de archivo ...");
60        }
61        finally {
62          archR.Close();
63        }
64            return RecList;
65      }
66
67
68        public Recurso getrecurso(int cod)
69        {
```

```
70          Stream archR = File.Open(route + nombre, FileMode.Open);
71          BinaryReader lector = new BinaryReader(archR);
72          try
73          {
74              while (true)
75              {
76                  var Ri = new Recurso();
77                  Ri.readArch(lector);
78                  if (Ri.getCodR() == cod) {
79                      archR.Close();
80                      return Ri;
81                  }
82              }
83          }
84          catch (Exception)
85          {
86              Console.WriteLine("Fin de archivo ...");
87          }
88          finally
89          {
90              archR.Close();
91          }
92          return null;
93      }
94
95
96      public bool eliminar( int codigo ) {
97      bool sw = false;
98      try {
99        // Abrimos el archivo o se crea un nuevo archivo si no existe
100       Stream archp = File.Open(route+nombre, FileMode.Open);
101       Stream archTemporal = File.Open(route+"temp.dat", FileMode.
      OpenOrCreate);
102       BinaryReader lector = new BinaryReader(archp);
103       BinaryWriter escribeTemp = new BinaryWriter(archTemporal);
104       Recurso Ei = new Recurso();
105
106       try {
107         while( true ) {
108           Ei.readArch(lector);
109           if( Ei.getCodR() != codigo ) {
110             Ei.writeArch(escribeTemp);
111           }
112           else {
113             sw = true;
114           }
115         }
116       }
117       catch( Exception ) {
118         Console.WriteLine("NoEncontroEl Archivo");
119         // No hace nada.
120       }
121       finally {
122         archp.Close();
123         archTemporal.Close();
124         File.Replace(route+"temp.dat",route+nombre, nombre + ".bak");
```

```
125         }
126       }
127       catch( Exception ) {
128         Console.WriteLine("El archivo no se puede acceder !!!");
129       }
130       return sw;
131     }
132
133     public string getRoute(){return route + nombre;}
134   }
135 }
```

7.2 Clases del entorno gráfico

Clase del formulario principa:l

```
1  using System;
2  using System.Collections.Generic;
3  using System.Windows.Forms;
4
5  namespace Prestamos
6  {
7    public partial class MainForm : Form
8    {
9      private ArchEst archE;
10     private ArchRec archR;
11
12     public MainForm()
13     {
14       InitializeComponent();
15       archE = new ArchEst("HistEstudiantes.dat");
16       archR = new ArchRec("Inventario.dat");
17       listar();
18     }
19
20     void MainFormLoad(object sender, EventArgs e){}
21     void Label1Click(object sender, EventArgs e){}
22     void Label2Click(object sender, EventArgs e){}
23
24     void Button1Click(object sender, EventArgs e)
25     {
26       Form AggInst = new AddRecursos();
27       AggInst.FormClosed += new System.Windows.Forms.FormClosedEventHandler(
       FormClosed );
28       AggInst.Show();
29
30     }
31     void Button2Click(object sender, EventArgs e)
32     {
33       Form Prestar_i = new Prestar();
34       Prestar_i.FormClosed += new System.Windows.Forms.FormClosedEventHandler(
       FormClosed );
35       Prestar_i.Show();
36
37     }
```

```
38      void Button3Click(object sender, EventArgs e)
39      {
40        Form Devolveri = new Devolver();
41        Devolveri.FormClosed += new System.Windows.Forms.FormClosedEventHandler(
        FormClosed );
42        Devolveri.Show();
43      }
44
45      void Button4Click(object sender, EventArgs e)
46      {
47        Form consulta = new QuienTieneY();
48        consulta.Show();
49      }
50
51      void Button5Click(object sender, EventArgs e)
52      {
53        Form AddEst = new AddEst();
54        AddEst.FormClosed += new System.Windows.Forms.FormClosedEventHandler(
       FormClosed );
55        AddEst.Show();
56      }
57      void Button6Click(object sender, EventArgs e)
58      {
59        Form consulta = new QueDebeX();
60        consulta.FormClosed += new System.Windows.Forms.FormClosedEventHandler(
       FormClosed );
61        consulta.Show();
62
63      }
64
65      void DataGridView2CellContentClick(object sender,
       DataGridViewCellEventArgs e){}
66      void GridEstudiantesCellContentClick(object sender,
       DataGridViewCellEventArgs e){}
67
68      private void FormClosed( object sender, FormClosedEventArgs e ) {
69          listar();
70      }
71
72
73      public void listar() {
74              var estList = archE.listar();
75              GridEstudiantes.Rows.Clear();
76              foreach (Estudiante Ei in estList) {
77                  int n= GridEstudiantes.Rows.Add();
78                  GridEstudiantes.Rows[n].Cells[1].Value = Ei.getNombre();
79                  GridEstudiantes.Rows[n].Cells[0].Value = Ei.getCI();
80              }
81              var recList = archR.listar();
82              GridDisponibles.Rows.Clear();
83              GridPrestados.Rows.Clear();
84              foreach (Recurso Ri in recList) {
85                if (Ri.getEstadoR())
86                {int n= GridPrestados.Rows.Add();
87                  GridPrestados.Rows[n].Cells[1].Value = Ri.getCodR();
88                  GridPrestados.Rows[n].Cells[0].Value = Ri.getNombreR();
```

```
89                  GridPrestados.Rows[n].Cells[2].Value = Ri.getCiPrest();
90                }
91                else
92                {int n= GridDisponibles.Rows.Add();
93                GridDisponibles.Rows[n].Cells[1].Value = Ri.getCodR();
94                    GridDisponibles.Rows[n].Cells[0].Value = Ri.getNombreR();
95                }
96              }
97
98          }
99    }
100 }
```

Clase del formulario agregar estudiante:

```
1  using System;
2  using System.Windows.Forms;
3
4  namespace Prestamos
5  {
6    public partial class AddEst : Form
7    {
8
9      private ArchEst archE = new ArchEst("HistEstudiantes.dat");
10     public AddEst()
11     {
12       InitializeComponent();
13     }
14     void Label2Click(object sender, EventArgs e){}
15     void Button1Click(object sender, EventArgs e)
16     {
17       try
18       {
19         if (CI.Text != "" && Nombre.Text != "")
20         {
21           var Ei = new Estudiante(Int32.Parse(CI.Text), Nombre.Text);
22           archE.addE(Ei);
23         }
24         Close();
25       }
26       catch (Exception)
27       {
28         Alerta.Text = "C.I no valido, revise nuevamente";
29       }
30     }
31   }
32 }
```

Clase del formulario agregar recurso:

```
1  using System;
2  using System.Windows.Forms;
3
4  namespace Prestamos
5  {
6
7    public partial class AddRecursos : Form
8    {
```

```
9     private ArchRec archR = new ArchRec("Inventario.dat");
10
11    public AddRecursos()
12    {
13      InitializeComponent();
14    }
15
16    void Label2Click(object sender, EventArgs e){}
17    void Label1Click(object sender, EventArgs e){}
18    void Label3Click(object sender, EventArgs e){}
19    void Button1Click(object sender, EventArgs e)
20    {
21      try
22      {
23        if(CodR.Text != "" && Nombre.Text !="")
24        {
25        var Ri = new Recurso(Int32.Parse(CodR.Text), Nombre.Text);
26        archR.addR(Ri);
27        Close();
28        }
29      }
30      catch (Exception)
31      {
32        Alerta.Text = "Introduzca un codigo Valido";
33        Console.WriteLine("Error al agregar el recurso");
34      }
35    }
36  }
37 }
```

Clase formulario registro devolución recurso

```
1  using System;
2  using System.Drawing;
3  using System.Windows.Forms;
4
5  namespace Prestamos
6  {
7    public partial class Devolver : Form
8    {
9      private ArchEst archE;
10     private ArchRec archR;
11     public Devolver()
12     {
13       InitializeComponent();
14       archE = new ArchEst("HistEstudiantes.dat");
15       archR = new ArchRec("Inventario.dat");
16     }
17     void Label1Click(object sender, EventArgs e){}
18     void Button1Click(object sender, EventArgs e)
19     {
20       if ( CI.Text != "" && Cod.Text !="")
21       { try
22         {
23           Recurso Ri =  archR.getrecurso(Int32.Parse(Cod.Text));
24           if (!Ri.getEstadoR())
25           {
```

```
26              Mensaje.Text = "El recurso NO ha sido prestado";
27            }
28            else {
29              Estudiante Ei = archE.getEstudante(Int32.Parse(CI.Text));
30              if (Ei.devolverX(Ri.getCodR()))
31              { Mensaje.Text = "Se devolvio el Articulo";
32                Ri.setEstado(false);
33                archR.eliminar(Int32.Parse(Cod.Text));
34                archR.addR(Ri);
35                archE.eliminar(Int32.Parse(CI.Text));
36                archE.addE(Ei);
37                Close();
38              }
39              else{ Mensaje.Text = "El Estudiante no tiene el articulo";}
40            }
41          }
42          catch (Exception)
43          {
44            Mensaje.Text = "Error, revise sus parametros";
45          }
46        }
47      }
48    }
49  }
```

Clase formulario registro préstamo de recurso

```
1  using System;
2  using System.Windows.Forms;
3  namespace Prestamos
4  {
5    public partial class Prestar : Form
6    {
7      private ArchEst archE;
8      private ArchRec archR;
9      public Prestar()
10     {
11       InitializeComponent();
12       archE = new ArchEst("HistEstudiantes.dat");
13       archR = new ArchRec("Inventario.dat");
14     }
15     void Label2Click(object sender, EventArgs e){}
16     void Button1Click(object sender, EventArgs e)
17     {
18       if (fecha.Text != "" && CI.Text != "" && Cod.Text !="")
19       try
20       {
21         Recurso Ri =  archR.getrecurso(Int32.Parse(Cod.Text));
22         if (Ri.getEstadoR())
23         {
24           Mensaje.Text = "El recurso ya ha sido prestado";
25         }
26         else
27         {
28           Ri.setEstado(true);
29           Estudiante Ei = archE.getEstudante(Int32.Parse(CI.Text));
30           Ei.prestarX(Ri.getCodR());
```

```
31            Ri.setCiPrest(Ei.getCI());
32            archR.eliminar(Int32.Parse(Cod.Text));
33            archR.addR(Ri);
34            archE.eliminar(Int32.Parse(CI.Text));
35            archE.addE(Ei);
36            Close();
37          }
38        }
39        catch (Exception)
40        {
41          Mensaje.Text = "Error, revise sus parÃ¡metros";
42        }
43      }
44      void Label1Click(object sender, EventArgs e){}
45      void TextBox1TextChanged(object sender, EventArgs e){}
46    }
47  }
```

Clase formulario consulta que debe el estudiante X

```
1  using System;
2  using System.Windows.Forms;
3
4  namespace Prestamos
5  {
6    public partial class QueDebeX : Form
7    {
8      private ArchRec archR = new ArchRec("Inventario.dat");
9      private ArchEst archE = new ArchEst("HistEstudiantes.dat");
10     public QueDebeX()
11     {
12       InitializeComponent();
13     }
14     void Label2Click(object sender, EventArgs e){ }
15     void ConsultarClick(object sender, EventArgs e)
16     {
17       if (Cod.Text != "")
18       {
19         try
20         {
21           Estudiante Ei = archE.getEstudante(Int32.Parse(Cod.Text));
22           Nombre.Text = Ei.getNombre();
23           var recList = archR.listar();
24                 Grid.Rows.Clear();
25                 foreach (Recurso Ri in recList) {
26                   if (Ri.getEstadoR() && Ri.getCiPrest() == Ei.getCI())
27                   {int n= Grid.Rows.Add();
28                     Grid.Rows[n].Cells[1].Value = Ri.getCodR();
29                     Grid.Rows[n].Cells[0].Value = Ri.getNombreR();
30                   }
31                 }
32         }
33         catch (Exception)
34         {
35           Nombre.Text = "NO EXISTE";
36         }
37       }
```

```
38      }
39
40      void CerrarClick(object sender, EventArgs e)
41      {
42        Close();
43      }
44    }
45  }
```

Clase formulario consulta quien tiene el recurso Y

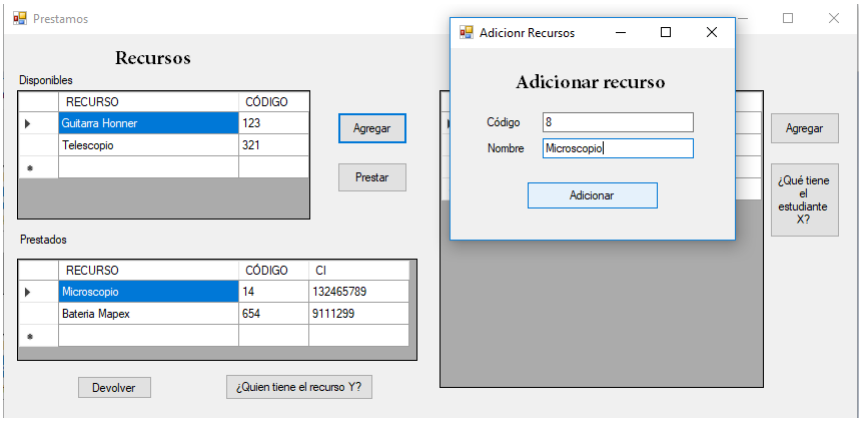
```
1  using System;
2  using System.Windows.Forms;
3
4  namespace Prestamos
5  {
6    public partial class QuienTieneY : Form
7    {
8      private ArchRec archR = new ArchRec("Inventario.dat");
9      private ArchEst archE = new ArchEst("HistEstudiantes.dat");
10     public QuienTieneY()
11     {
12       InitializeComponent();
13     }
14     void Label2Click(object sender, EventArgs e){}
15     void CerrarClick(object sender, EventArgs e)
16     {
17       Close();
18     }
19     void ConsultarClick(object sender, EventArgs e)
20     {
21       if (Cod.Text != "")
22       {
23         try
24         {
25           Recurso Ri = archR.getrecurso(Int32.Parse(Cod.Text));
26           if (Ri.getEstadoR())
27           {
28             Estudiante Ei = archE.getEstudante(Ri.getCiPrest());
29             Mensaje.Text = "REC:\t" + Ri.getNombreR() + "\n" + "EST:\t" +  Ei.
    getNombre() + "\n" +"CI:\t"+   Ei.getCI();
30           }
31           else
32           {
33             Mensaje.Text = "ACTUALMENTE DISPONIBLE";
34           }
35
36         }
37         catch (Exception)
38         {
39           Mensaje.Text = "NO EXISTE";
40         }
41
42       }
43     }
44   }
45 }
```

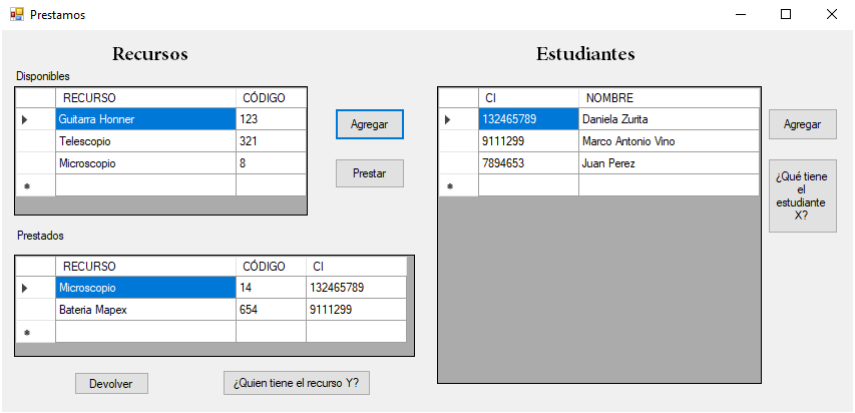
Clase que inicializa el formulario principal

```
using System;
using System.Windows.Forms;

namespace Prestamos
{
  internal sealed class Program
  {
    [STAThread]
    private static void Main(string[] args)
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new MainForm());
    }

  }
}
```

8 Ejecución del programa

1) Agregar un recurso a la central de préstamos.

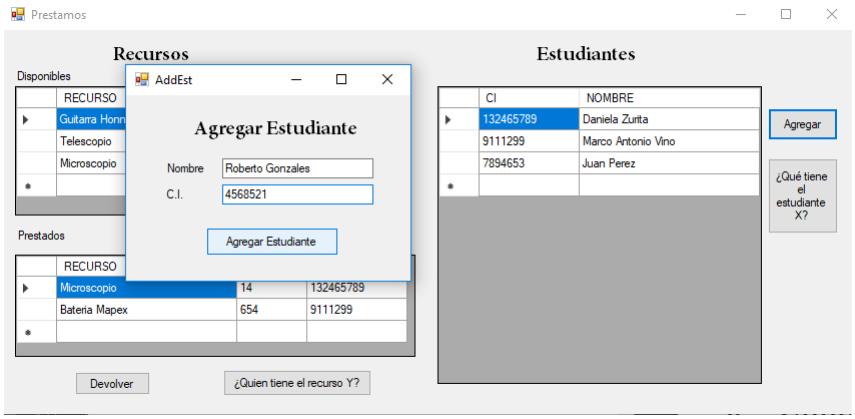


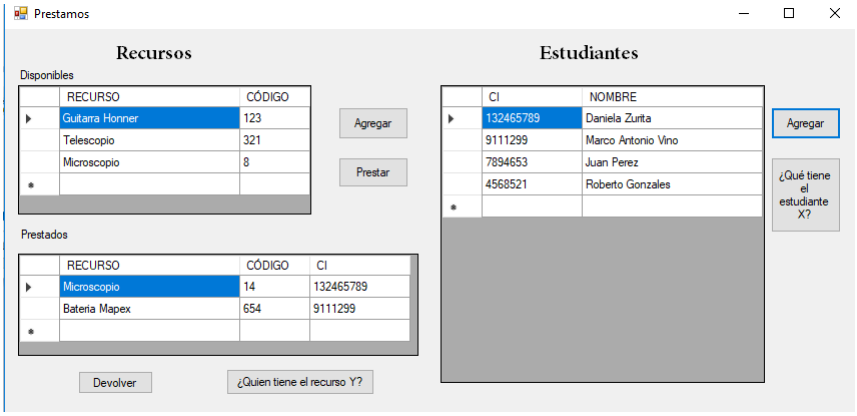


Inicialmente no tenia el recurso **microscopio** con el **código 8**, pero despues e la ejecución se lo tiene registrado. La función que permite esta adición es **Button1Click()** del Formulario *addRecurso*.

```
1     void Button1Click(object sender, EventArgs e)
2     {
3       try
4       {
5         if(CodR.Text != "" && Nombre.Text !="")
6         {
7         var Ri = new Recurso(Int32.Parse(CodR.Text), Nombre.Text);
8         archR.addR(Ri);
9         Close();
10        }
11      }
12      catch (Exception)
13      {
14        Alerta.Text = "Introduzca un codigo Valido";
15        Console.WriteLine("Error al agregar el recurso");
16      }
17    }
```

2) Agregar un estudiante a la central de préstamos.

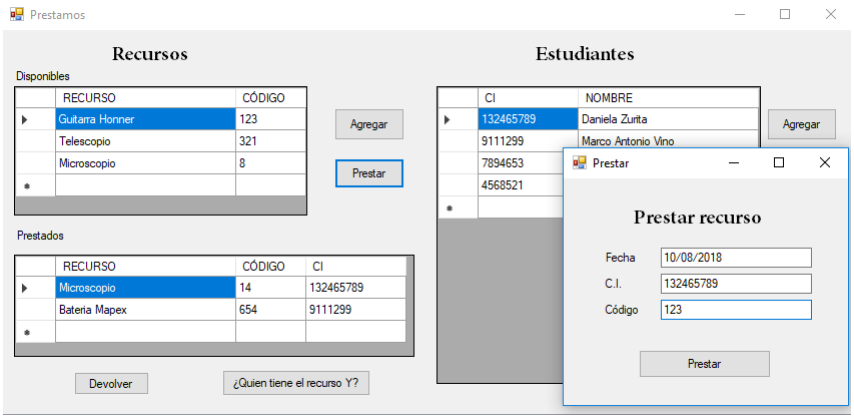


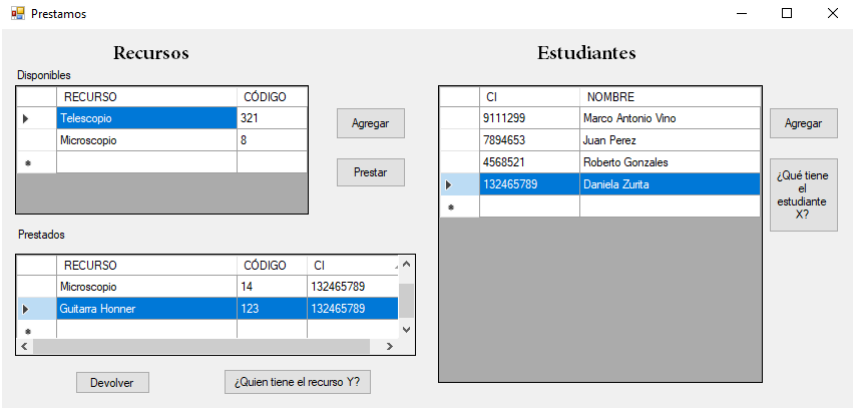


Inicialmente no tenia el estudiante **Roberto Gonzales** con el **CI 4568521**, pero despues e la ejecución se lo tiene registrado. La función que permite esta adición es **Button1Click()** del Formulario *addEstudiante*.

```
1     void Button1Click(object sender, EventArgs e)
2     {
3       try
4       {
5         if (CI.Text != "" && Nombre.Text != "")
6         {
7           var Ei = new Estudiante(Int32.Parse(CI.Text), Nombre.Text);
8           archE.addE(Ei);
9         }
10        Close();
11      }
12      catch (Exception)
13      {
14        Alerta.Text = "C.I no valido, revise nuevamente";
15      }
16    }
```

3) Prestar un recurso disponible a un estudiante.



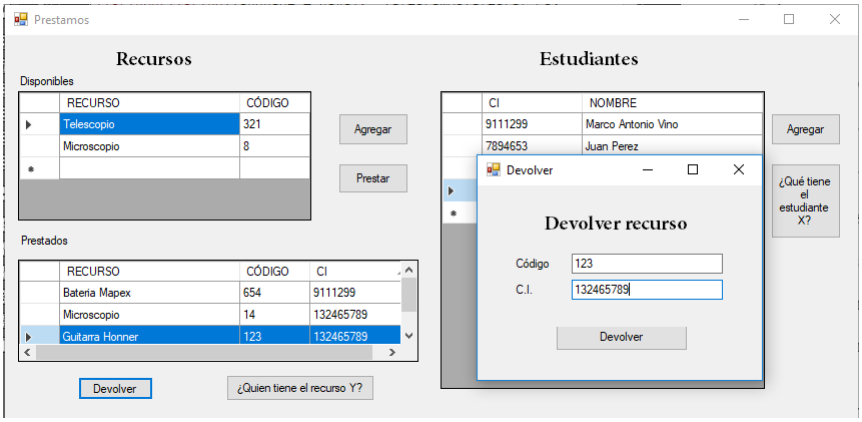


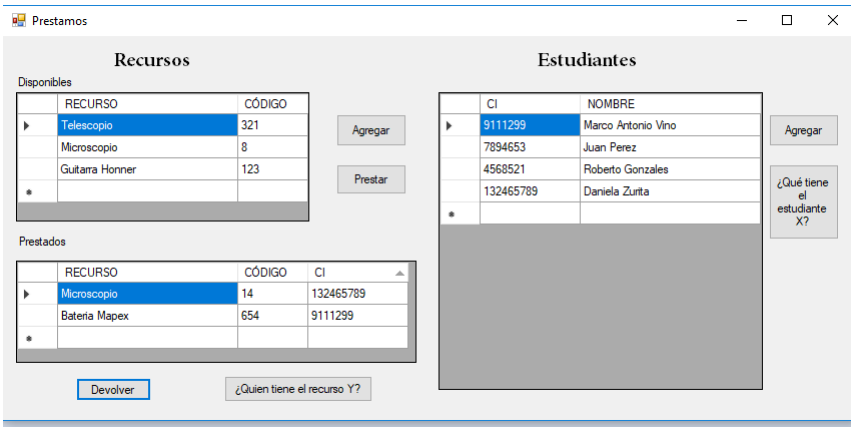
Inicialmente el recurso *guitarra honner* con *código 123*, estaba disponible. Pero se lo prestaron a la persona con CI 132465789. Con lo que en la nueva vista pasa a la lista de prestados y desaparece de los disponibles. El método que permite esto es el **Button1Click()** dentro del formulario *Prestar*.

```
1    void Button1Click(object sender, EventArgs e)
2    {
3      if (fecha.Text != "" && CI.Text != "" && Cod.Text !="")
4      try
5      {
6        Recurso Ri =  archR.getrecurso(Int32.Parse(Cod.Text));
7        if (Ri.getEstadoR())
8        {
9          Mensaje.Text = "El recurso ya ha sido prestado";
10       }
11       else
12       {
13         Ri.setEstado(true);
14         Estudiante Ei = archE.getEstudante(Int32.Parse(CI.Text));
15         Ei.prestarX(Ri.getCodR());
16         Ri.setCiPrest(Ei.getCI());
17         archR.eliminar(Int32.Parse(Cod.Text));
18         archR.addR(Ri);
19         archE.eliminar(Int32.Parse(CI.Text));
20         archE.addE(Ei);
```

```
21              Close();
22            }
23          }
24          catch (Exception)
25          {
26            Mensaje.Text = "Error, revise sus parÃ¡metros";
27          }
28        }
```

4) registrar la devolución de un recurso prestado a un estudiante.



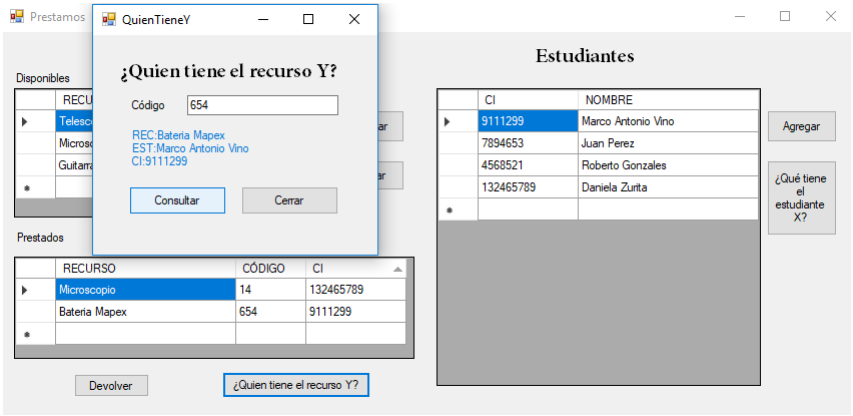


En las pantallas se muestra la devolución el recurso con *código 123* del *estudiante 132465789*, esto se lleva a cabo dentro del formulario *Devolver* con el método **Button1Click()**.

```
1     void Button1Click(object sender, EventArgs e)
2     {
3       if ( CI.Text != "" && Cod.Text !="")
4       { try
5         {
6           Recurso Ri =  archR.getrecurso(Int32.Parse(Cod.Text));
7           if (!Ri.getEstadoR())
8           {
9             Mensaje.Text = "El recurso NO ha sido prestado";
10          }
11          else {
```

```
12              Estudiante Ei = archE.getEstudante(Int32.Parse(CI.Text));
13              if (Ei.devolverX(Ri.getCodR()))
14              { Mensaje.Text = "Se devolvio el Articulo";
15                Ri.setEstado(false);
16                archR.eliminar(Int32.Parse(Cod.Text));
17                archR.addR(Ri);
18                archE.eliminar(Int32.Parse(CI.Text));
19                archE.addE(Ei);
20                Close();
21              }
22              else{ Mensaje.Text = "El Estudiante no tiene el articulo";}
23            }
24          }
25          catch (Exception)
26          {
27            Mensaje.Text = "Error, revise sus parametros";
28          }
29        }
30      }
```

5) Consultar ¿Qué estudiante tiene prestado el recurso Y?.

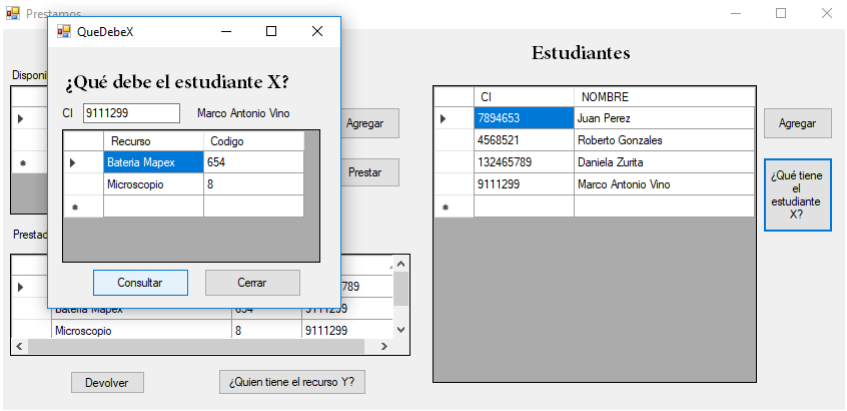


Al ingresar al boton ¿Quien tiene el recurso Y?, se despliega una nueva ventana que nos permite hacer la consulta en base al código del recurso. En este caso se pregunto por el recurso con el *código 654*, y nos muestra que lo tiene *Marco Antonio Vino*, esto se realiza a través del método **ConsultarClick()** del formulario *QuienTieneY*.

```
1     void ConsultarClick(object sender, EventArgs e)
2     {
3       if (Cod.Text != "")
4       {
5         try
6         {
7           Recurso Ri = archR.getrecurso(Int32.Parse(Cod.Text));
8           if (Ri.getEstadoR())
9           {
10            Estudiante Ei = archE.getEstudante(Ri.getCiPrest());
11            Mensaje.Text = "REC:\t" + Ri.getNombreR() + "\n" + "EST:\t" +
    Ei.getNombre() + "\n" +"CI:\t"+    Ei.getCI();
12          }
```

```
13              else
14              {
15                 Mensaje.Text = "ACTUALMENTE DISPONIBLE";
16              }
17
18           }
19           catch (Exception)
20           {
21              Mensaje.Text = "NO EXISTE";
22           }
23
24        }
25     }
```

6) Consultar los recursos que tiene prestado el estudiante X.



Al ingresar al boton ¿Qué tiene el estudiante X?, se despliega una nueva ventana que nos permite hacer la consulta en base al *CI* del estudiante. En este caso se pregunto por el Estudiante con el *CI 9111299*, y nos muestra Su nombre*Marco Antonio Vino* además de una lista de los recursos que se le pretaron, esto se realiza a través del método **ConsultarClick()** del formulario *QuéDebeX*.

```
1     void ConsultarClick(object sender, EventArgs e)
2     {
3        if (Cod.Text != "")
4        {
5           try
6           {
7              Estudiante Ei = archE.getEstudante(Int32.Parse(Cod.Text));
8              Nombre.Text = Ei.getNombre();
9              var recList = archR.listar();
10                Grid.Rows.Clear();
11                foreach (Recurso Ri in recList) {
12                   if (Ri.getEstadoR() && Ri.getCiPrest() == Ei.getCI())
13                   {int n= Grid.Rows.Add();
14                      Grid.Rows[n].Cells[1].Value = Ri.getCodR();
15                      Grid.Rows[n].Cells[0].Value = Ri.getNombreR();
16                   }
17                }
18          }
```

```
19          catch (Exception)
20          {
21              Nombre.Text = "NO EXISTE";
22          }
23      }
24  }
```

9 Conclusiones

Podemos observar que el desarrollo de uns sistema requiere de una planificaición, diseño y estructuración estrictos para poder ser realizados. Tambien nuestra aplicación puede cumplir con todos los requerimientos iniciales del enunciado del trabajo.
Las funcionalidades del mismo pueden ser ampliadas con desarrollos posteriores.

10 Bibliografía

References

[1] Gregorics, B., Gregorics, T., Kovács, G. F., Dobreff, A., and Dévai, G. Textual diagram layout language and visualization algorithm. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)* (2015), IEEE, pp. 196–205.

[2] Heinz, C., and Moses, B. The listings package, 1996.

[3] Joyanes Aguilar, L. *Programación orientada a objetos*. No. 004.652. 5. McGraw-Hill Interamericana,, 1996.

[4] Stevens, P., Whittle, J., and Booch, G. *UML 2003–The Unified Modeling Language, Modeling Languages and Applications: 6th International Conference San Francisco, CA, USA, October 20-24, 2003, Proceedings*, vol. 2863. Springer, 2003.

[5] Strauss, D. *C# Programming Cookbook*. Packt Publishing Ltd, 2016.

[6] Takeyas, B. L. Introducción a la programación en c# .net.