



Manual Técnico

Curso: Diseño de Aplicaciones de Software

Jefry López Meneses

Harold André Rodríguez Cortes

Karl Aase Blanco

Mauricio Paniagua Morales

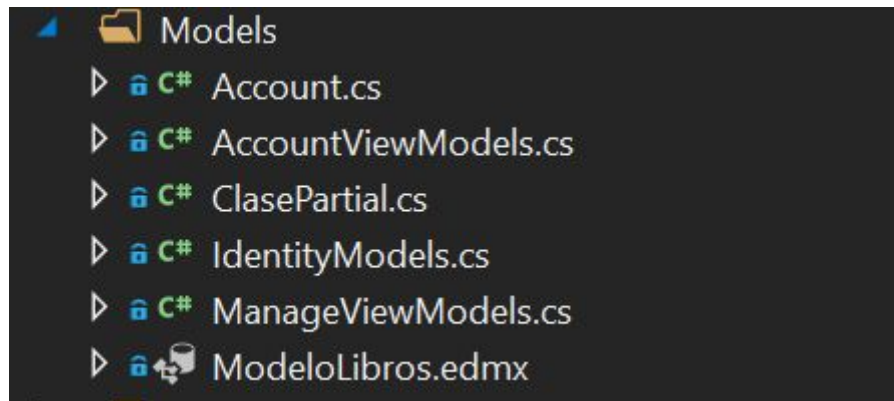
Profesor: Fernando Salas Castro

I Cuatrimestre 2020

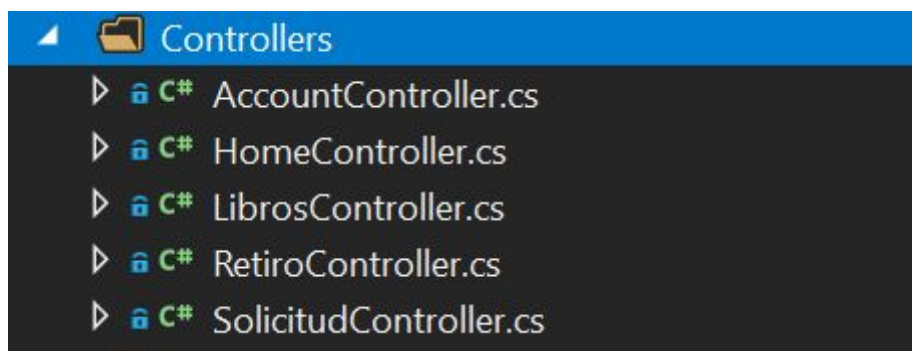
Manual Técnico

Componentes del Proyecto

Models (Datos)



Controllers (Lógica)



Account Controller

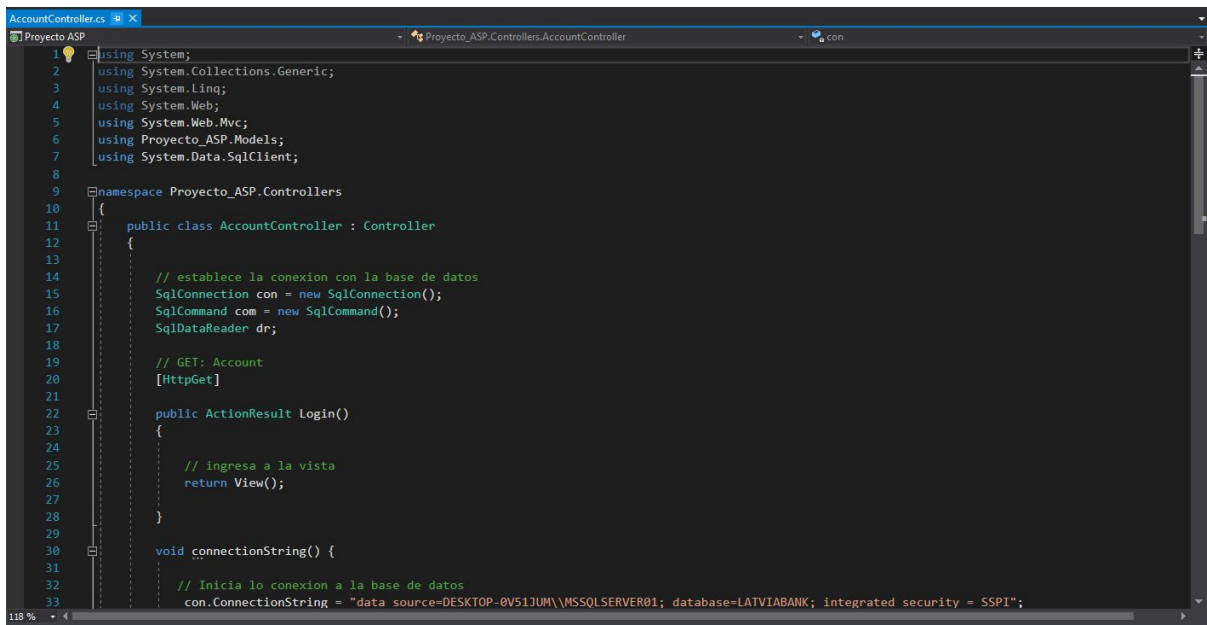
Funcionalidad: la funcionalidad de este controlador es establecer una conexión con la base de datos para así permitir o denegar el acceso a los administradores al programa.

Métodos

- Login()
- connectionString()
- Verify(Account acc)

Variables

- SqlConnection con = new SqlConnection();
- SqlCommand com = new SqlCommand();
- SqlDataReader dr;
- Name
- Password



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6 using Proyecto_ASP.Models;
7 using System.Data.SqlClient;
8
9 namespace Proyecto_ASP.Controllers
10 {
11     public class AccountController : Controller
12     {
13
14         // establece la conexion con la base de datos
15         SqlConnection con = new SqlConnection();
16         SqlCommand com = new SqlCommand();
17         SqlDataReader dr;
18
19         // GET: Account
20         [HttpGet]
21
22         public ActionResult Login()
23         {
24
25             // ingresa a la vista
26             return View();
27         }
28
29         void connectionString() {
30
31             // Inicia lo conexion a la base de datos
32             con.ConnectionString = "data source=DESKTOP-0V51JUM\\MSSQLSERVER01; database=LATVIABANK; integrated security = SSPI";
33         }
34     }
35 }
```

```
AccountController.cs
Proyecto ASP
Proyecto_ASP.Controllers.AccountController
con

31
32 // Inicia la conexión a la base de datos
33 con.ConnectionString = "data source=DESKTOP-0V51JUM\\MSSQLSERVER01; database=LATVIABANK; integrated security = SSPI";
34
35 }
36
37
38 [HttpPost]
39 public ActionResult Verify(Account acc) {
40
41 // Establece conexión con la tabla de usuarios para comprobar los datos
42 connectionString();
43 con.Open();
44 con.Connection = con;
45 con.CommandText = "select * from CAT_USERS where USERNAME='"+acc.Name+"' and PASSWORD='"+acc.Password+"'";
46 dr = con.ExecuteReader();
47 if (dr.Read())
48 {
49 // Si todo corresponde envía al Menu
50 con.Close();
51 Response.Redirect("http://localhost:53651/Home/Index/22");
52 return View("Create");
53 }
54 else
55 {
56 // Si no corresponde vuelve a enviar al login
57 con.Close();
58 Response.Redirect("http://localhost:53651/Account/Login");
59
60 return View("Error");
61
62
63 }
```

HomeController

Funcionalidad: Se encarga de administrar la vista del Menú

Métodos

- Index()

```
HomeController.cs
Proyecto ASP
Proyecto_ASP.Controllers.HomeController
Index()

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6
7 namespace Proyecto_ASP.Controllers
8 {
9     public class HomeController : Controller
10     {
11         public ActionResult Index()
12         {
13             // Muestra la vista Home(Menu)
14             return View();
15         }
16     }
17 }
```

LibrosController

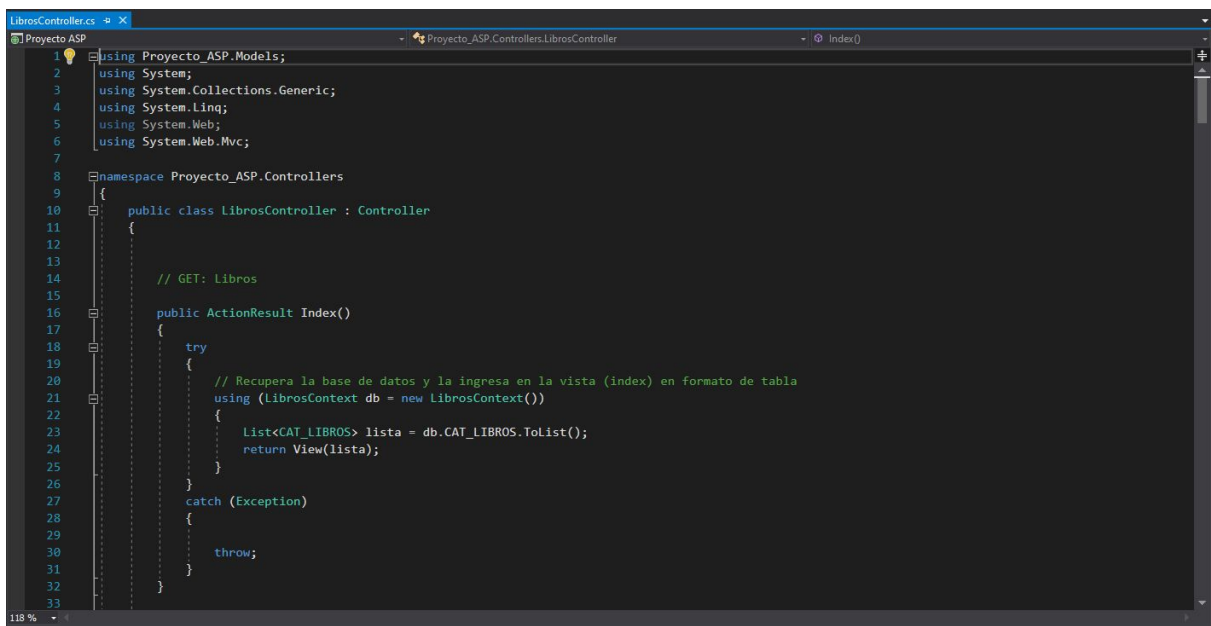
Funcionalidad: la funcionalidad de este controlador es establecer una conexión con la base de datos para así Agregar/Eliminar/Editar Libros y de igual forma poder dar de baja libros que se encuentren dañados

Métodos

- Index()
- Buscar()
- Agregar()
- Editar()
- Editar(CAT_LIBROS Libro)
- Detalles(int id)
- Eliminar(int id)
- Librosdañados()
- Back()
- IndexDañados()
- AgregarDañados(int id)
- AgregarDañados(CAT_LIBROS_DAÑADOS Libro)
- DetallesDañados(int id)
- EliminarDañados(int id)

Variables

- List<CAT_LIBROS> lista
- int id
- LibrosContext db
- string libro
- int id_isbn
- CAT_LIBROS Blibro
- CAT_LIBROS Libro
- List<CAT_LIBROS_DAÑADOS> lista
- CAT_LIBROS_DAÑADOS Libro



```
1  using Proyecto_ASP.Models;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Web;
6  using System.Web.Mvc;
7
8  namespace Proyecto_ASP.Controllers
9  {
10     public class LibrosController : Controller
11     {
12
13         // GET: Libros
14
15         public ActionResult Index()
16         {
17             try
18             {
19                 // Recupera la base de datos y la ingresa en la vista (index) en formato de tabla
20                 using (LibrosContext db = new LibrosContext())
21                 {
22                     List<CAT_LIBROS> lista = db.CAT_LIBROS.ToList();
23                     return View(lista);
24                 }
25             }
26             catch (Exception)
27             {
28                 throw;
29             }
30         }
31     }
32 }
33
```

```
LibrosController.cs x
Proyecto ASP Proyecto_ASP.Controllers.LibrosController Index()
34 // Busqueda
35
36 public ActionResult Buscar(FormCollection item)
37 {
38     try
39     {
40         // Metodo de busqueda especifica atraves del isbn
41         string libro = item["id"];
42         LibrosContext db = new LibrosContext();
43         int id_isbn = Convert.ToInt32(libro);
44
45         var datos = db.CAT_LIBROS.Where(x => x.isbn == id_isbn).Select(x => x).ToList();
46         // una vez encontrado actualiza la vista (index) y el formato tabla
47         return View("~/Views/Libros/Index.cshtml", datos);
48     }
49     catch (Exception)
50     {
51     }
52     throw;
53 }
54
55 // Agregar Libros
56
57 public ActionResult Agregar()
58 {
59     // Metodo para mostrar la vista agregar
60     return View();
61 }
62
63
64 // Metodo encargado de procesar la respuesta de la vista de agregar y asignar a la base de datos el nuevo libro
65 [HttpPost]
66
```

```
LibrosController.cs x
Proyecto ASP Proyecto_ASP.Controllers.LibrosController Index()
162 // Detalles Libros
163 public ActionResult Detalles(int id)
164 {
165     try
166     {
167         // muestra la vista detalles con los datos encontrados en la base de datos
168         using (var db = new LibrosContext())
169         {
170             CAT_LIBROS Blibro = db.CAT_LIBROS.Find(id);
171             return View(Blibro);
172         }
173     }
174     catch (Exception)
175     {
176     }
177     throw;
178 }
179
180 // Eliminar libros
181 public ActionResult Eliminar(int id)
182 {
183     try
184     {
185         // El metodo busca atravez del isbn el libro y procede a removerlo de la base de datos
186         using (var db = new LibrosContext())
187         {
188             CAT_LIBROS Blibro = db.CAT_LIBROS.Find(id);
189         }
190     }
191     catch (Exception)
192     {
193     }
194     throw;
195 }
```

```
LibrosController.cs
Proyecto ASP
- Proyecto_ASP.Controllers.LibrosController
- Index()

65 // Metodo encargado de procesar la respuesta de la vista de agregar y asignar a la base de datos el nuevo libro
66 [HttpPost]
67 [ValidateAntiForgeryToken]
68 public ActionResult Agregar(CAT_LIBROS Libro)
69 {
70
71     // Verificamos el modelo
72     if (!ModelState.IsValid)
73         return View();
74
75     try
76     {
77         using (var db = new LibrosContext())
78         {
79             // Añadimos los datos a la base de datos
80             db.CAT_LIBROS.Add(Libro);
81             db.SaveChanges();
82             return RedirectToAction("Index");
83         }
84     }
85     catch (Exception ex)
86     {
87         // En caso de que el modelo este incorrecto mostrara un fallo
88         ModelState.AddModelError("", "Fallo al ingresar libro -" + ex.Message);
89         return View();
90     }
91 }
92
93 // Editar Libros
94
95 public ActionResult Editar(int id)
```

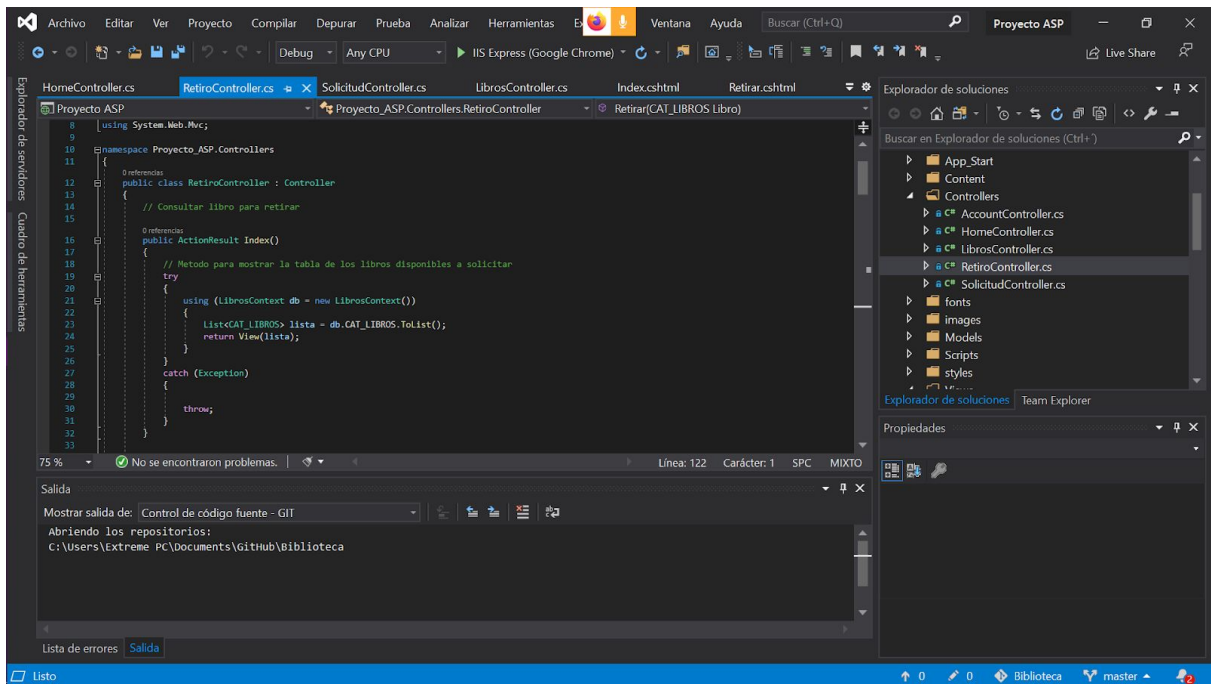
```
LibrosController.cs
Proyecto ASP
- Proyecto_ASP.Controllers.LibrosController
- Index()

95 // Editar Libros
96
97 public ActionResult Editar(int id)
98 {
99     // Metodo para mostrar la vista Editar a travez de la busqueda del isbn del libro
100     try
101     {
102         if (!ModelState.IsValid)
103             return View();
104         using (var db = new LibrosContext())
105         {
106             CAT_LIBROS Blibro = db.CAT_LIBROS.Find(id);
107             return View(Blibro);
108         }
109     }
110     catch (Exception)
111     {
112         throw;
113     }
114 }
115
116 // Metodo encargado de procesar la respuesta de la vista de Editar y re-asignar a la base de datos el libro
117
118 [HttpPost]
119 [ValidateAntiForgeryToken]
```



```
LibrosController.cs - x
Proyecto ASP - Proyecto_ASP.Controllers.LibrosController - Index()

124 // Metodo encargado de procesar la respuesta de la vista de Editar y re-asignar a la base de datos el libro
125
126 [HttpPost]
127 [ValidateAntiForgeryToken]
128 public ActionResult Editar(CAT_LIBROS Libro)
129 {
130     try
131     {
132         using (var db = new LibrosContext())
133         {
134             // Cambiamos los datos que se encuentran en la base de datos
135             CAT_LIBROS BLibro = db.CAT_LIBROS.Find(Libro.isbn);
136             BLibro.isbn = Libro.isbn;
137             BLibro.nombre = Libro.nombre;
138             BLibro.autor = Libro.autor;
139             BLibro.editorial = Libro.editorial;
140             BLibro.edicion = Libro.edicion;
141             BLibro.escuela = Libro.escuela;
142             BLibro.unidades = Libro.unidades;
143             BLibro.tematica = Libro.tematica;
144             BLibro.asignatura = Libro.asignatura;
145             db.SaveChanges();
146             return RedirectToAction("Index");
147         }
148     }
149     catch (Exception)
150     {
151     }
152     throw;
153 }
154
155
156
```



```
LibrosController.cs - X
Proyecto ASP - Proyecto_ASP.Controllers.LibrosController - Index()

185 // Eliminar libros
186 public ActionResult Eliminar(int id)
187 {
188     try
189     {
190         // El metodo busca atravez del isbn el libro y procede a removerlo de la base de datos
191         using (var db = new LibrosContext())
192         {
193             CAT_LIBROS Blibro = db.CAT_LIBROS.Find(id);
194             db.CAT_LIBROS.Remove(Blibro);
195             db.SaveChanges();
196             return RedirectToAction("Index");
197         }
198     }
199     catch (Exception)
200     {
201     }
202     throw;
203 }
204
205
206
207
208
209
210 public ActionResult Librosdañados()
211 {
212     // Redireccionamiento a la base de datos de los libros dañados
213     Response.Redirect("http://localhost:53651/LibrosDa%C3%B1ados/IndexDa%C3%B1ados");
214     return View();
215 }
216
217
118 %
```

```
LibrosController.cs - X
Proyecto ASP - Proyecto_ASP.Controllers.LibrosController - Librosdañados()

218 public ActionResult Back()
219 {
220     // Redireccionamiento al Menu
221     Response.Redirect("http://localhost:53651/Home/Index/22");
222     return View();
223 }
224
225
226
227
228
229 public ActionResult IndexDañados()
230 {
231     // Metodo para mostrar la base de datos que contiene los libros dañados
232     try
233     {
234         using (LibrosContext db = new LibrosContext())
235         {
236             List<CAT_LIBROS_DAÑADOS> lista = db.CAT_LIBROS_DAÑADOS.ToList();
237             return View(lista);
238         }
239     }
240     catch (Exception)
241     {
242     }
243     throw;
244 }
245
246 //Programación para agregar libros dañados mostrando la vista para asignar el daño
247 public ActionResult AgregarDañados(int id)
248 {
249     using (var db = new LibrosContext())
250     {
251     }
252 }
118 %
```

```
LibrosController.cs - X
Proyecto ASP - Proyecto_ASP.Controllers.LibrosController - AgregarDaños(CAT_LIBROS_DAÑADOS Libro)

244     }
245     }
246     //Programación para agregar libros dañados mostrando la vista para asignar el daño
247     public ActionResult AgregarDaños(int id)
248     {
249         using (var db = new LibrosContext())
250         {
251             CAT_LIBROS_Blibro = db.CAT_LIBROS.Find(id);
252             return View(Blibro);
253         }
254     }
255
256
257     // Metodo que escucha la vista y asigna los datos
258     [HttpPost]
259     [ValidateAntiForgeryToken]
260     public ActionResult AgregarDaños(CAT_LIBROS_DAÑADOS Libro)
261     {
262         if (!ModelState.IsValid)
263             return View();
264
265         try
266         {
267             using (var db = new LibrosContext())
268             {
269                 db.CAT_LIBROS_DAÑADOS.Add(Libro);
270
271                 db.SaveChanges();
272                 return RedirectToAction("IndexDaños");
273             }
274         }
275         catch (Exception ex)
276         {
277             ModelState.AddModelError("", "Fallo al reportar libro dañado - " + ex.Message);
278             return View();
279         }
280     }
281
282
283
```

```
LibrosController.cs - X
Proyecto ASP - Proyecto_ASP.Controllers.LibrosController - AgregarDaños(CAT_LIBROS_DAÑADOS Libro)

284     // Metodo para obtener detalles del libro dañado en una vista exclusiva
285     public ActionResult DetallesDaños(int id)
286     {
287         try
288         {
289             using (var db = new LibrosContext())
290             {
291                 CAT_LIBROS_DAÑADOS Blibro = db.CAT_LIBROS_DAÑADOS.Find(id);
292                 return View(Blibro);
293             }
294         }
295         catch (Exception)
296         {
297             throw;
298         }
299     }
300
301
302
303
304
305
306     // Eliminar Libros Daños
307
308     public ActionResult EliminarDaños(int id)
309     {
310         try
311         {
312             // Metodo para eliminar un libro que se encuentre dañado en la base de datos
313             using (var db = new LibrosContext())
314             {
315                 CAT_LIBROS_DAÑADOS Blibro = db.CAT_LIBROS_DAÑADOS.Find(id);
316                 db.CAT_LIBROS_DAÑADOS.Remove(Blibro);
317                 db.SaveChanges();
318                 return RedirectToAction("IndexDaños");
319             }
320         }
321     }
322
323
```

```
LibrosController.cs
Proyecto ASP
Proyecto_ASP.Controllers.LibrosController
AgregarDañados(CAT_LIBROS_DANADOS Libro)

296     }
297 }
298
299     catch (Exception)
300     {
301     }
302     throw;
303 }
304
305 // Eliminar Libros Dañados
306
307 public ActionResult EliminarDañados(int id)
308 {
309     try
310     {
311         // Metodo para eliminar un libro que se encuentre dañado en la base de datos
312         using (var db = new LibrosContext())
313         {
314             CAT_LIBROS_DANADOS Blibro = db.CAT_LIBROS_DANADOS.Find(id);
315             db.CAT_LIBROS_DANADOS.Remove(Blibro);
316             db.SaveChanges();
317             return RedirectToAction("IndexDañados");
318         }
319     }
320     catch (Exception)
321     {
322     }
323     throw;
324 }
325
326
327
328
329
330
331 }
```

Retiro controller

Funcionalidad: la funcionalidad de este controlador es establecer una conexión con la base de datos para así administrar las solicitudes de los libros y generar tickets con la información correspondiente

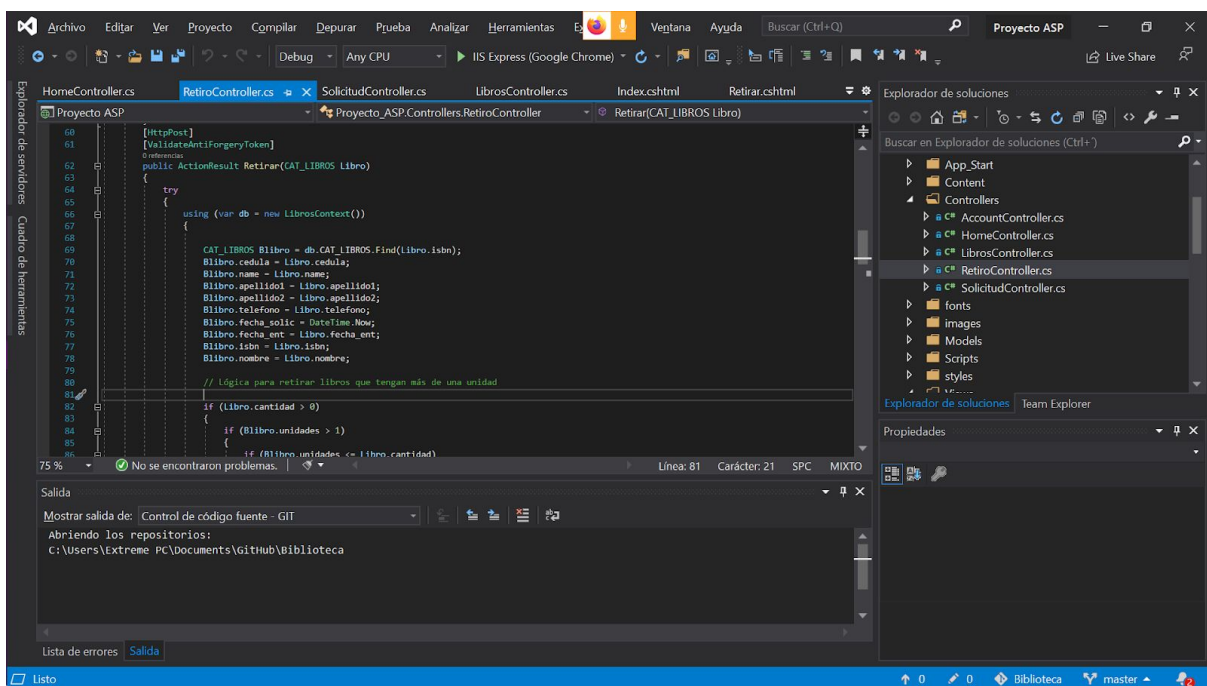
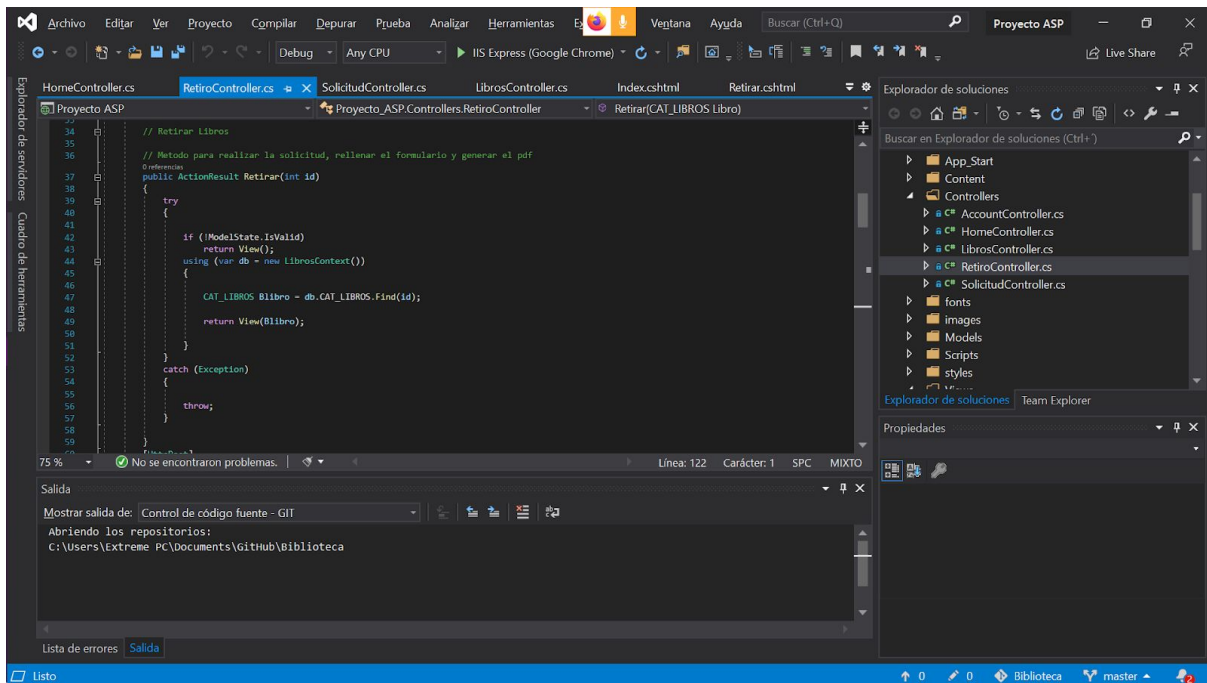
Métodos

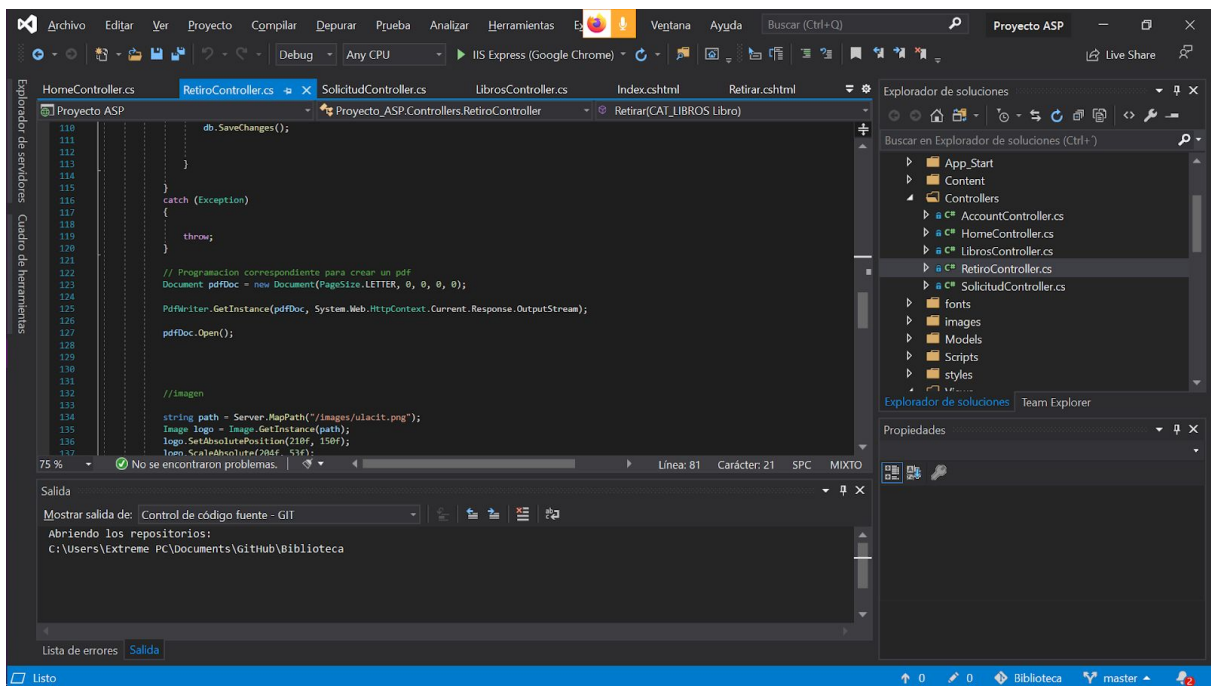
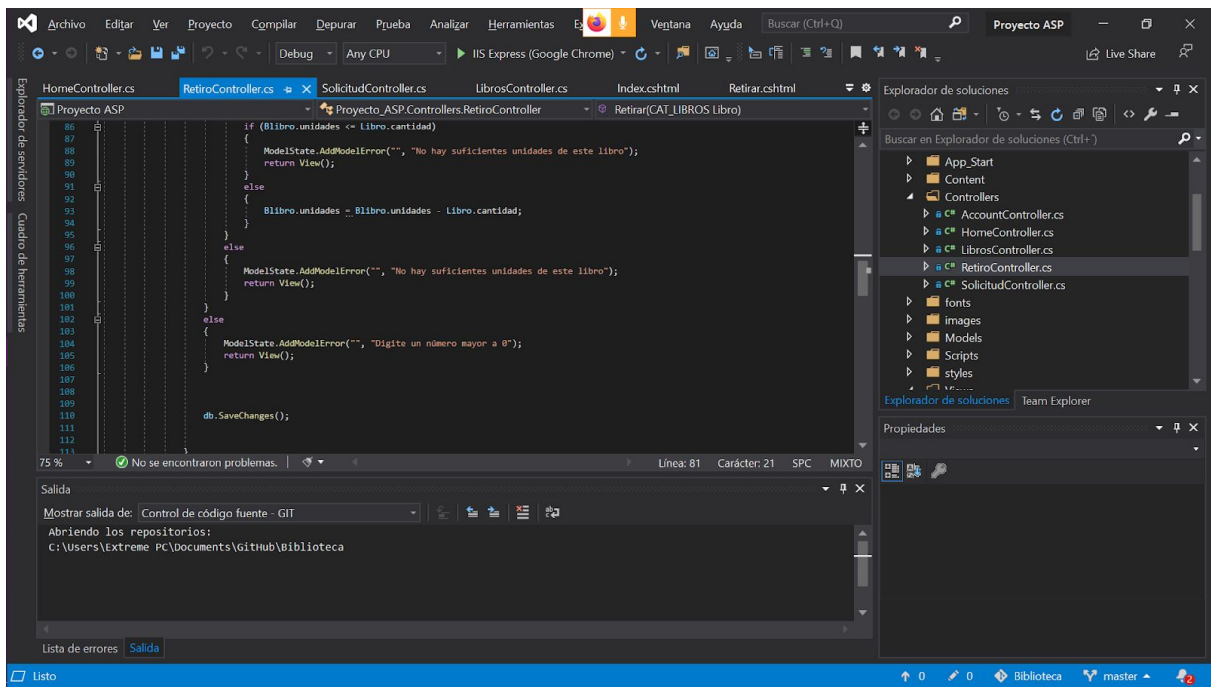
- Index()
- Retirar(int id)
- Retirar(CAT_LIBROS Libro)
- Back()

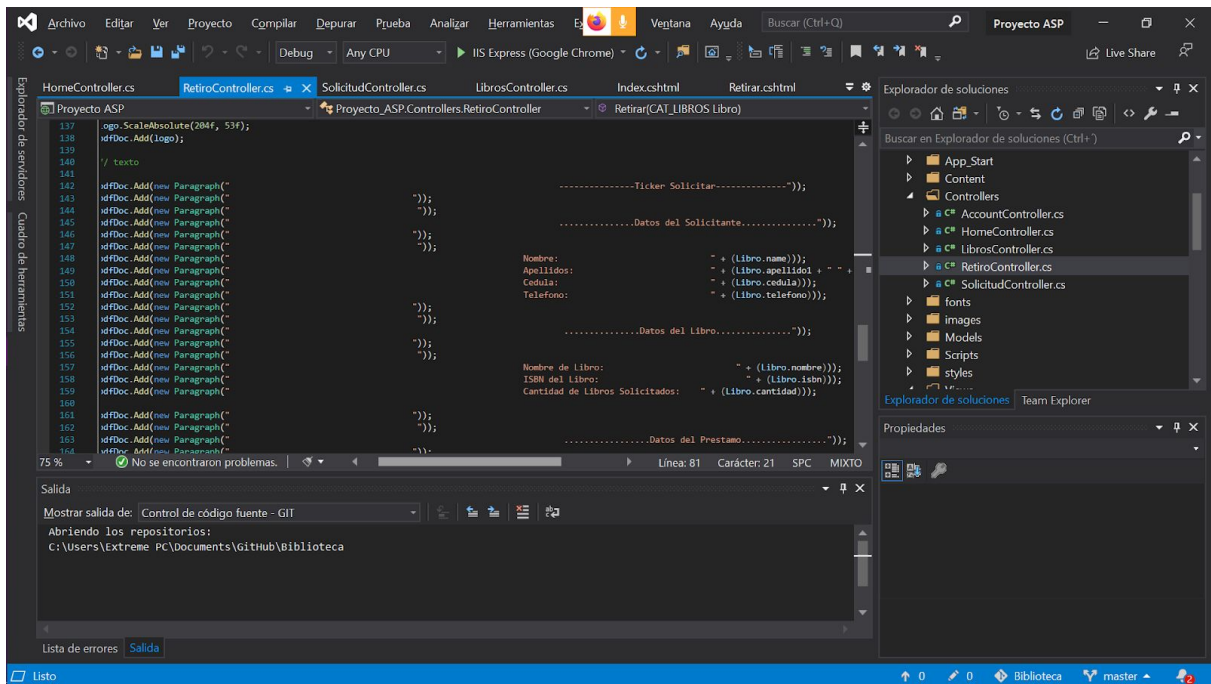
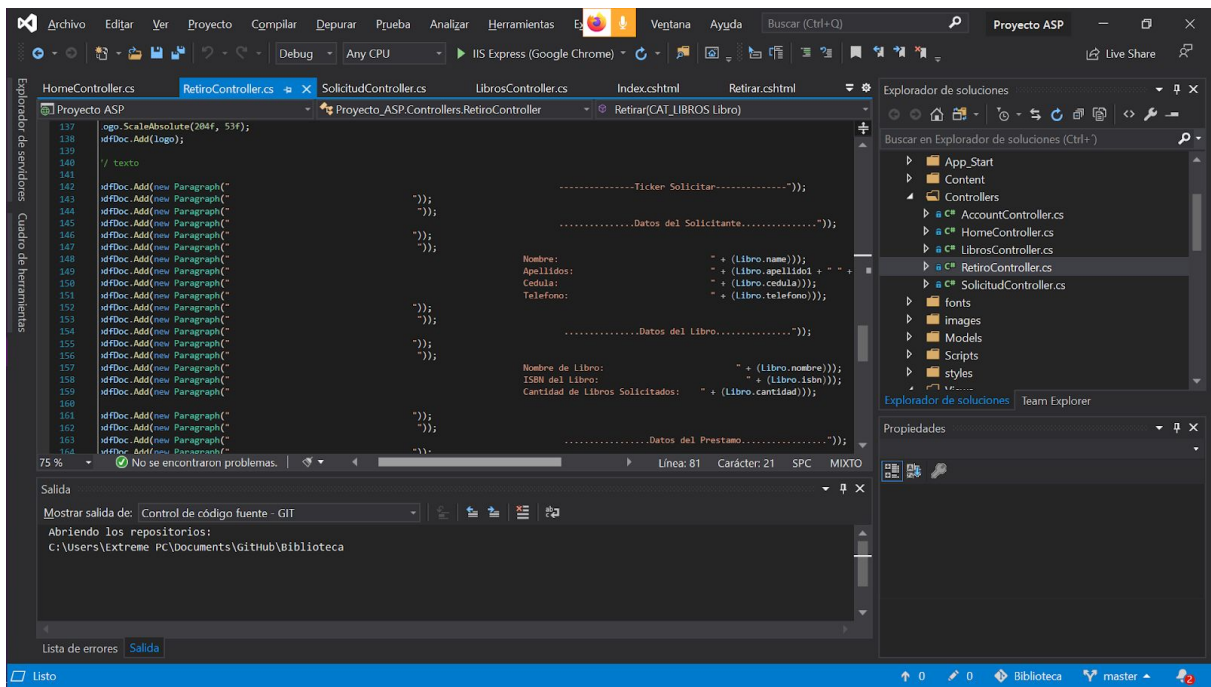
Variables

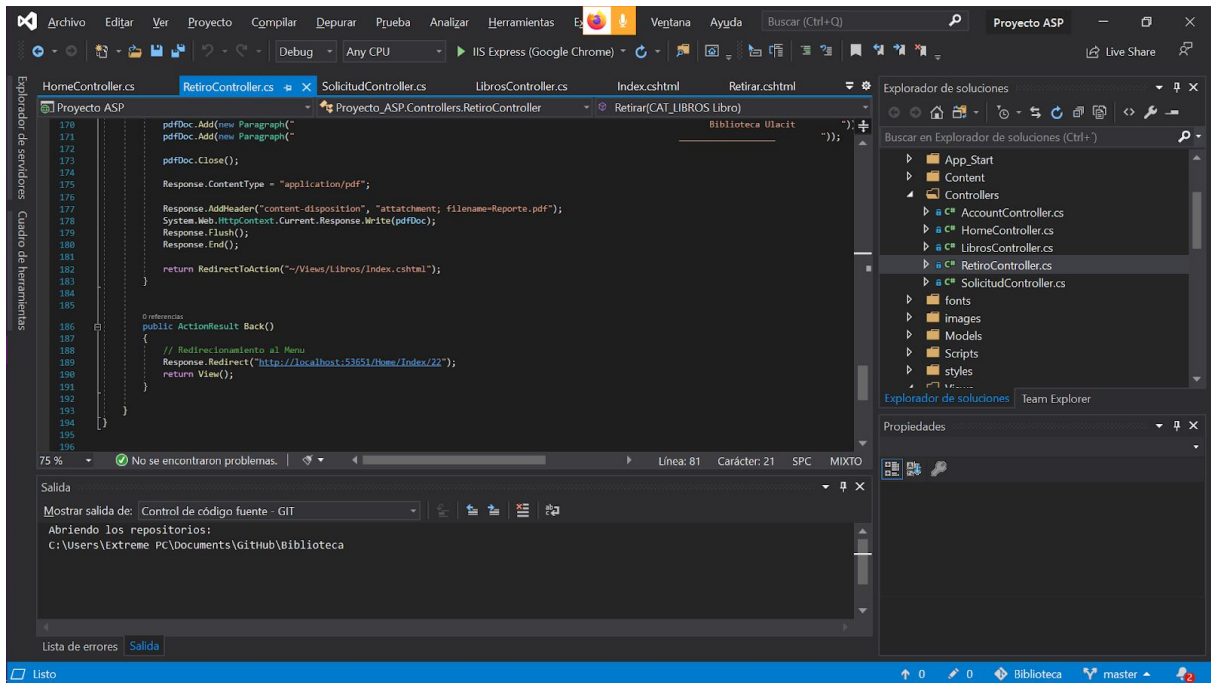
- LibrosContext db
- int id
- CAT_LIBROS Blibro

- CAT_LIBROS Libro
- List<CAT_LIBROS> lista
- string path







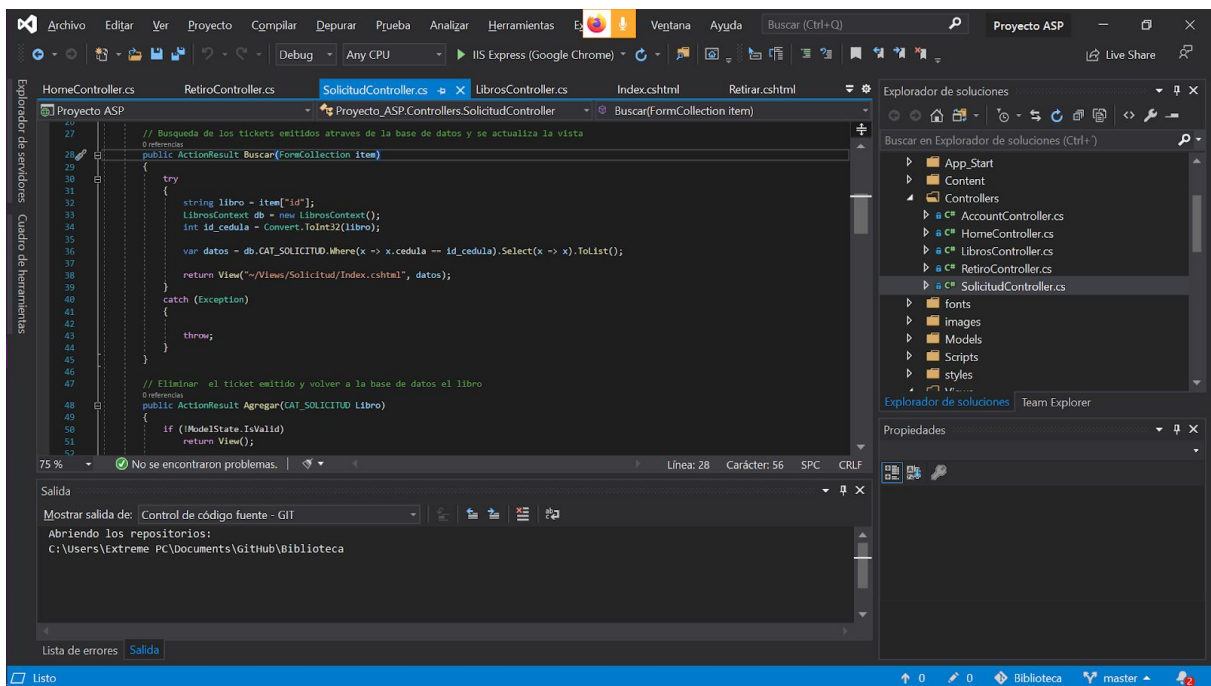
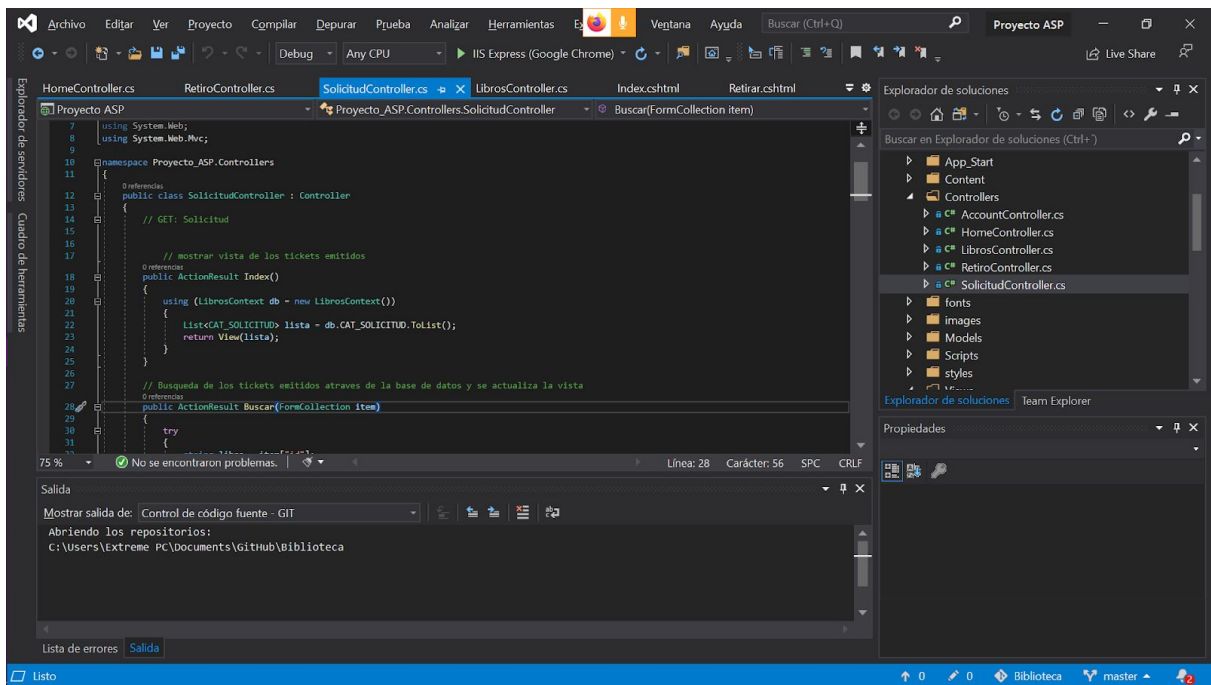


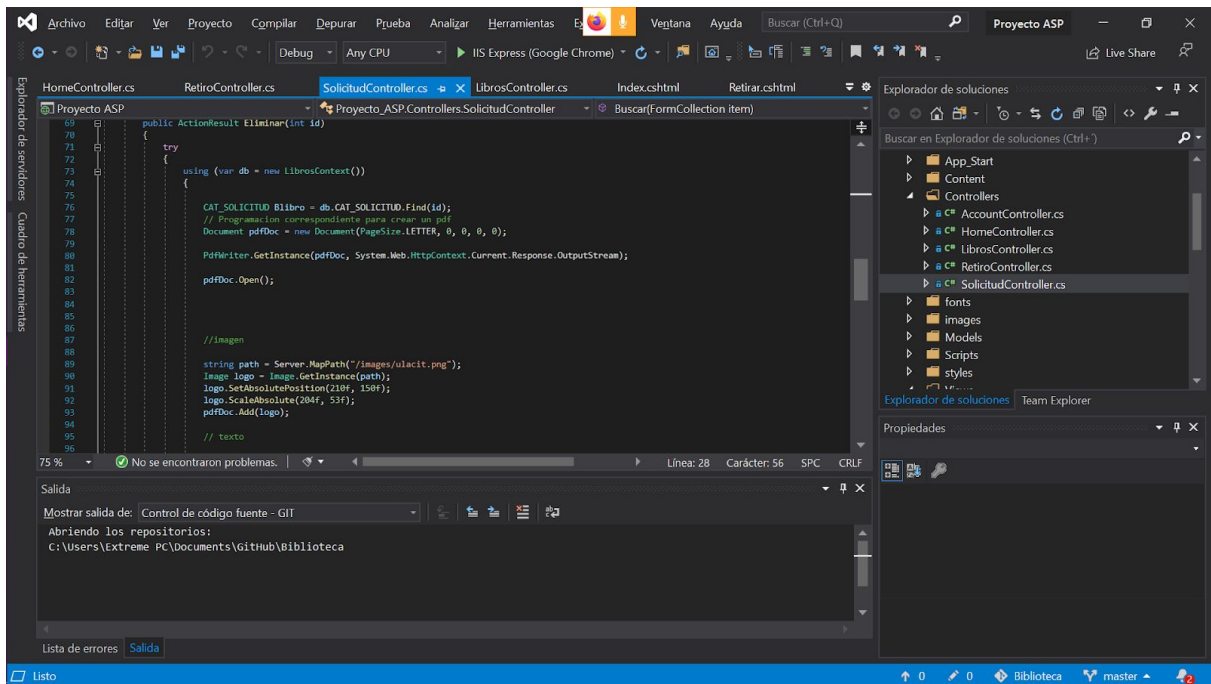
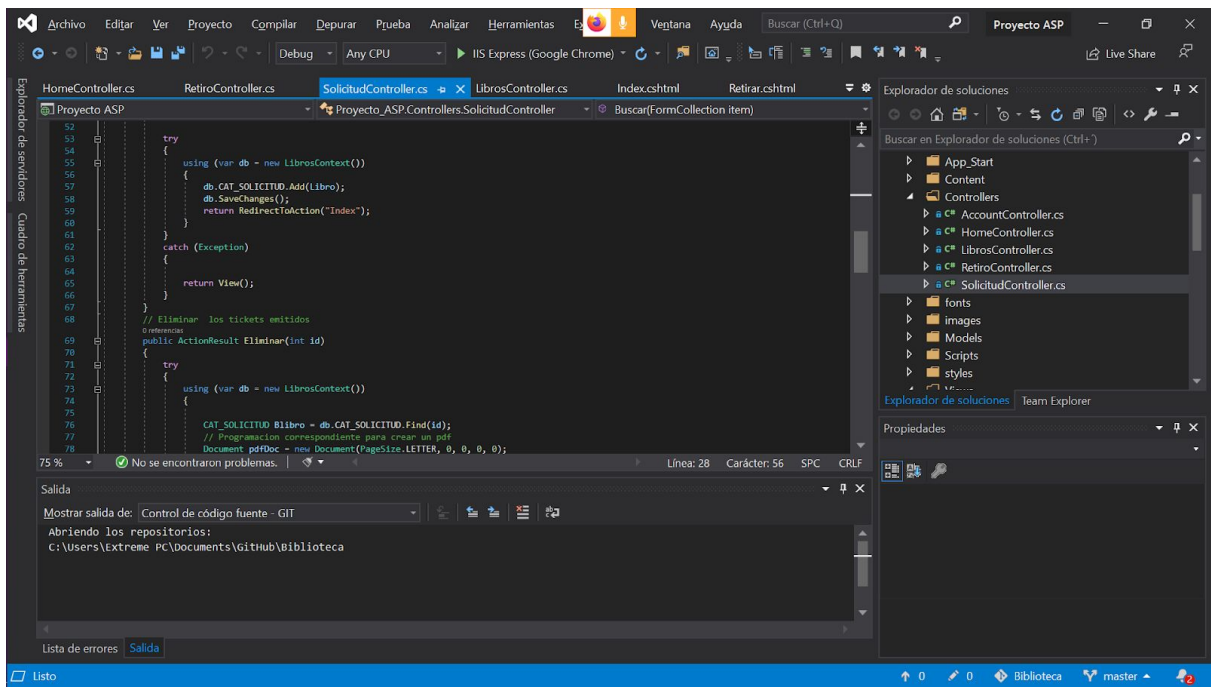
Solicitud Controller

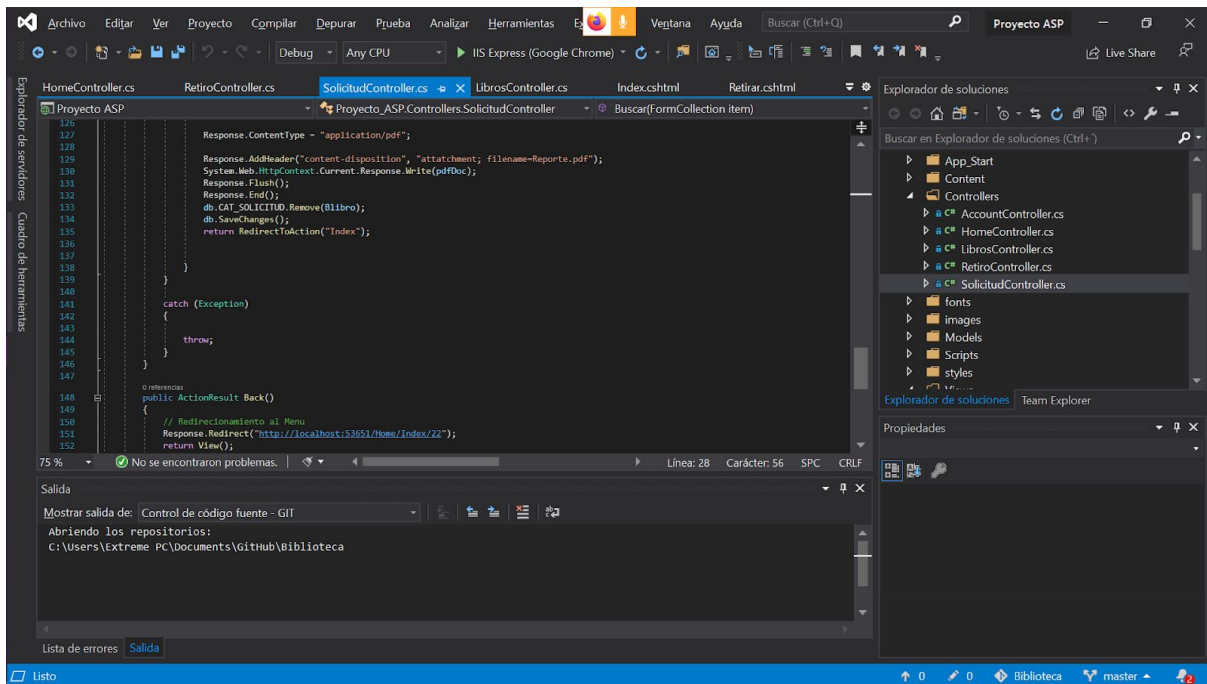
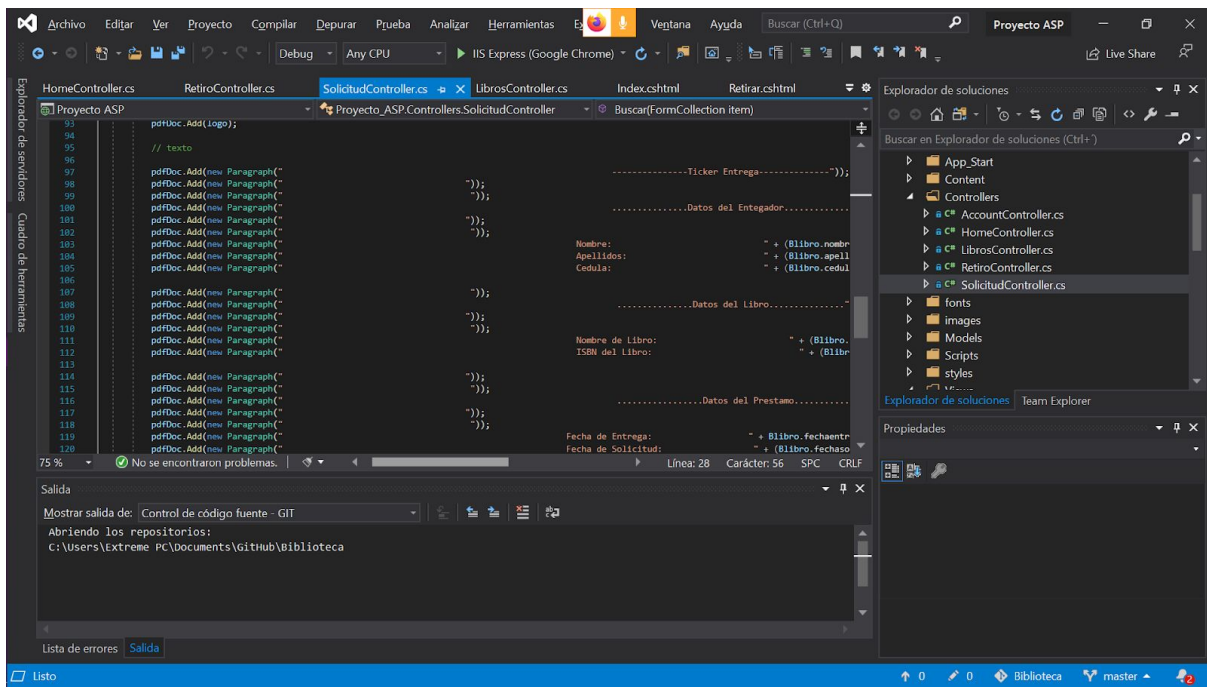
Funcionalidad: la funcionalidad de este controlador es establecer una conexión con la base de datos para así administrar la entrega de los libros y generar tickets con la información correspondiente

Métodos

- List<CAT_SOLICITUD> lista
- int id
- LibrosContext db
- string libro
- int id_cedula
- string path







Views(UI)

Parte de la interfaz Gráfica de todo el Programa

- Views
 - Account
 - Create.cshtml
 - Error.cshtml
 - Login.cshtml
 - Home
 - Index.cshtml
 - Libros
 - Agregar.cshtml
 - AgregarDañados.cshtml
 - Detalles.cshtml
 - DetallesDañados.cshtml
 - Editar.cshtml

- LibroBaja.cshtml
- Retiro
 - Index.cshtml
 - Retirar.cshtml
- Shared
 - _Layout.cshtml
 - _LoginPartial.cshtml
 - Error.cshtml
 - Lockout.cshtml
 - PaginaP.cshtml
- Solicitud
 - Index.cshtml
- _LayoutPage1.cshtml
- ViewStart.cshtml

Diagrama UML

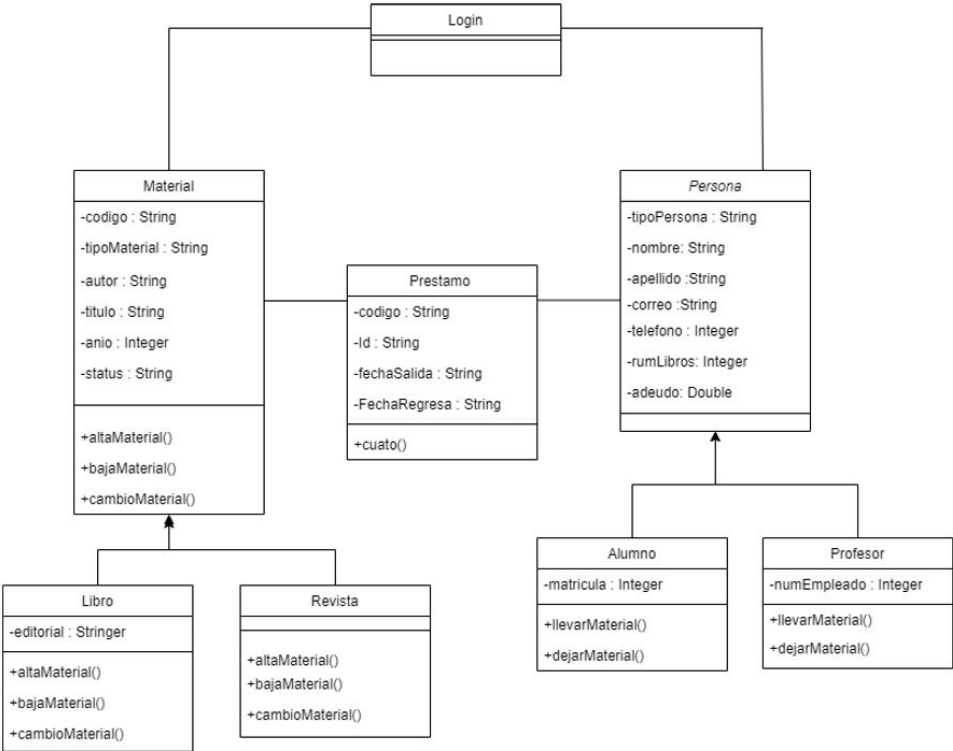


Diagrama de Flujo

