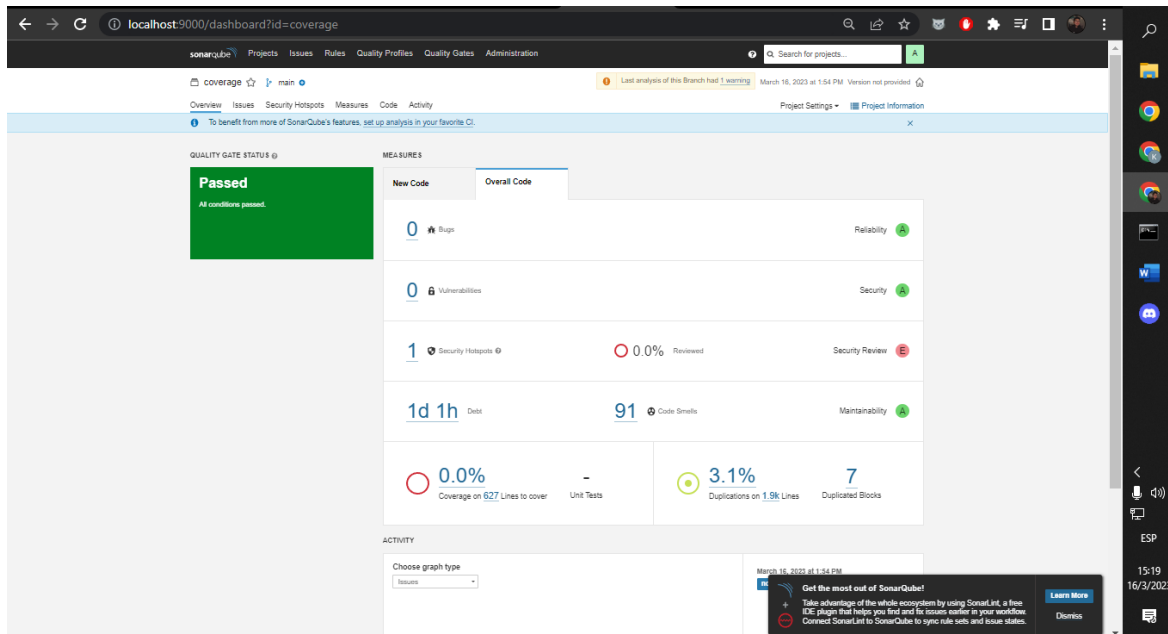
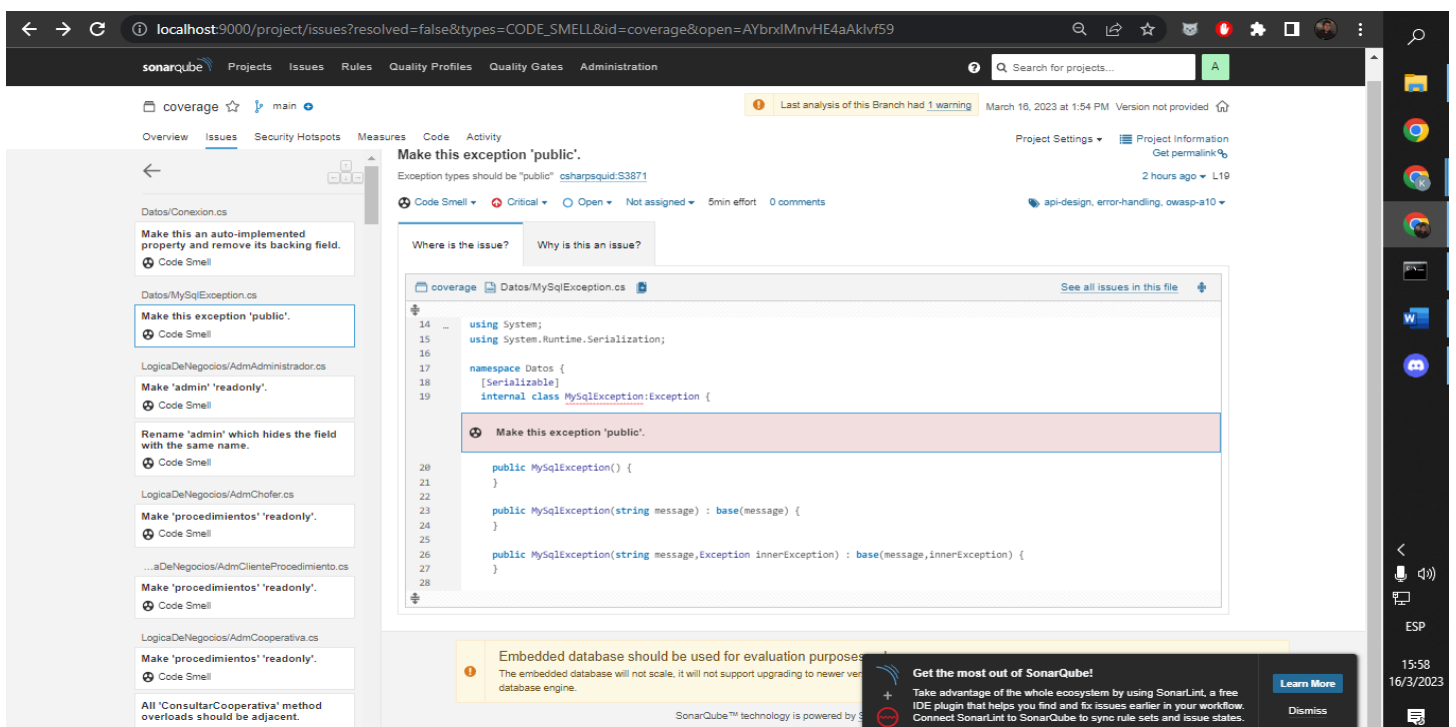


REQUISITOS DEL PROYECTO



El Sistema esta codificado con C# el cual se debió implementar varios archivos externos debido que es (.net). Se realizo el testing estático sobre código utilizando la herramienta SonarQube y se observo que no existen bugs y 0 vulnerabilidades este análisis, por ahora se puede decir que es un sistema seguro.

Se reviso (**code smell**) y existen una gran cantidad de malos olores.



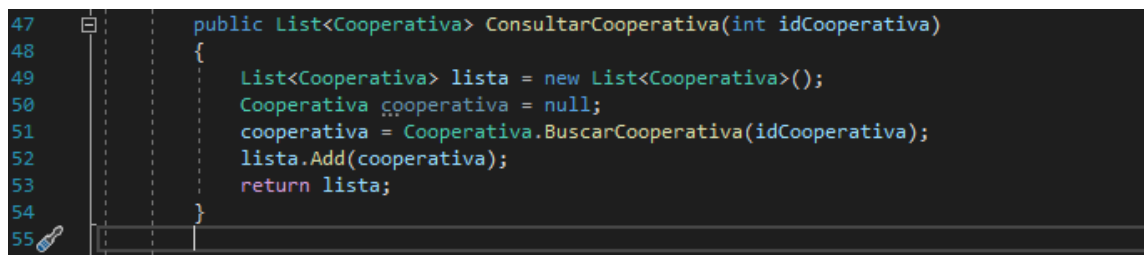
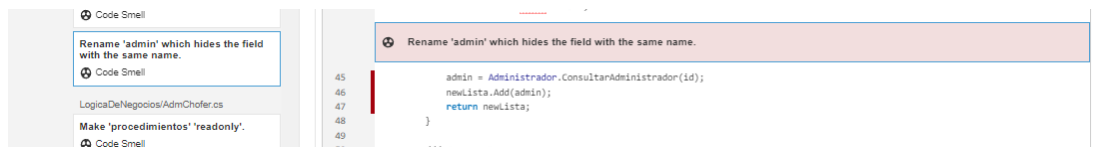
- Clases que deben ser pública.



- Atributos globales que únicamente se utiliza para leer datos.



- Asignación innecesaria para un valor null.



- Condición no completa puede generar deficiencia en el diseño.



- Esta línea no se ejecutará condicionalmente; solo lo estará la primera línea de este bloque de 2 líneas. El resto se ejecutará incondicionalmente.

```

37
38 MySqlCommand mySqlCommand = conector.ConectarProcedimiento(consulta, con.conectar());
39 if (idConsulta==1) mySqlCommand.Parameters.AddWithValue("@Dato", tiempo);
40 else mySqlCommand.Parameters.AddWithValue("@Dato", dato);
41 MySqlDataReader lector = mySqlCommand.ExecuteReader();
42 bool x = true;
43 while (lector.Read())
44 {
45     if (x) dato=IdReserva;
46     else dato=IdReserva;
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
85
```

- Implementación de 2 propiedades

- No se define el tipo de dato para el retorno de valor a la clase

```

34     public ControlExcepcion(String message) : base(message)
35     {
36         Console.WriteLine(this.message);
37     }
38 }
39
40

```

- Clases que deben ser privadas por temas de seguridad

Make this exception 'public'.

Code Smell

LogicaDeNegocios/AdmAdministrador.cs

Make 'admin' 'readonly'.

Code Smell

Rename 'admin' which hides the field with the same name.

Cefe Sanni

All 'Pago' method overloads should be adjacent.

```

15         private static List<string> boleto = new List<string>();
16         public static List<Pago> InfoBoleto = new List<Pago>();

```

Change the visibility of 'InfoBoleto' or make it 'const' or 'readonly'.

Make this field 'private' and encapsulate it in a 'public' property.

- Crear variables que ya reciben datos

Code Smell

Make this an auto-implemented property and remove its backing field.

Code Smell

LogicaDeNegocios/Bus.cs

Make this an auto-implemented property and remove its backing field.

Code Smell

Make this an auto-implemented property and remove its backing field.

Code Smell

Make this an auto-implemented property and remove its backing field.

Code Smell

LogicaDeNegocios/Chofer.cs

```

55 -         /// </summary>
56         /// <param name="cbOrigen">The cb origen.</param>
57         /// <param name="cbDestino">The cb destino.</param>
58         public void LlamarCombos(ComboBox cbOrigen, ComboBox cbDestino)
59         {
60             List<string> ciudad = new List<string>();
61
62             ciudad = procedimientos.CargarCiudad();
63             if (ciudad.Count!=0)
64             {
65                 foreach (string ciudadfx in ciudad)
66                 {
67                     cbOrigen.Items.Add(ciudadfx);
68                     cbDestino.Items.Add(ciudadfx);
69                 }

```

Remove this useless assignment to local variable 'ciudad'.

```

    /// </summary>
    String message = null;
    //constructor parametrizado
    /// <summary>
    /// Initializes a new instance of the <see cref="ControlExcepcion" /> class.
    /// </summary>
    /// <param name="message">The message.</param>
    35 referencias
    public ControlExcepcion(String message) : base(message)
    {
        Console.WriteLine(this.message);
    }
}

```

Complejidad Ciclomática

Complejidad Ciclomática (274)

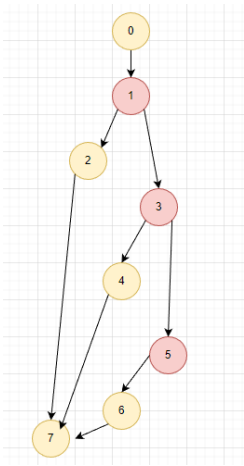
Project Overview	coverage / LogicaDeNegocios	View as Tree	1 1 to select files	to navigate	28 files
> Reliability	Cyclomatic Complexity 274				
> Security	Properties				0
> Security Review	AdmAdministrador.cs				3
> Maintainability	AdmChofer.cs				9
> Coverage	AdmClienteProcedimiento.cs				9
> Duplications	AdmCooperativa.cs				11
> Size	Administrador.cs				13
> Complexity	AdmPago.cs				2
> Cyclomatic Complexity	AdmReporte.cs				8
> Cognitive Complexity	AdmRuta.cs				1
> Issues	AdmVendedor.cs				9
	Boleto.cs				7
	Bus.cs				8
	Chofer.cs				18
	Ciente.cs				12
	ConectorDeProcedimientos.cs				1
	ConsultaProcedimientosGenerarBoleto.cs				4

- AdmVendedor Consultar Vendedor ()

```
List<Vendedor> vendedor = null;
if (String.IsNullOrEmpty(datoVendedor))
{
    throw new ControlExcepcion("Campo vacio por favor rellenar");
}
else
{
    if (datoVendedor == "Iniciar_data_grid_datos")
    {
        datoVendedor = "";
    }
    vendedor = procedimientos.BuscarVendedor(datoVendedor);

    if (vendedor.Count == 0)
    {
        throw new ControlExcepcion("Vendedor no encontrado");
    }
}

return vendedor;
}
```

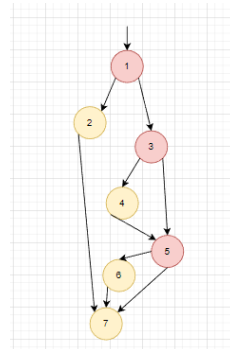


- AdmCliente **Consultar Cliente ()**

```
public List<Cliente> ConsultarCliente(string Dato )
{
    List<Cliente> client = null;
    if (String.IsNullOrEmpty(Dato))
    {
        throw new ControlExcepcion("Campo vacio por favor rellenar");
    }
    else
    {
        if (Dato == "Iniciar_data_grid_datos")
        {
            Dato = "";
        }
        client = procedimientos.BuscarCliente(Dato);

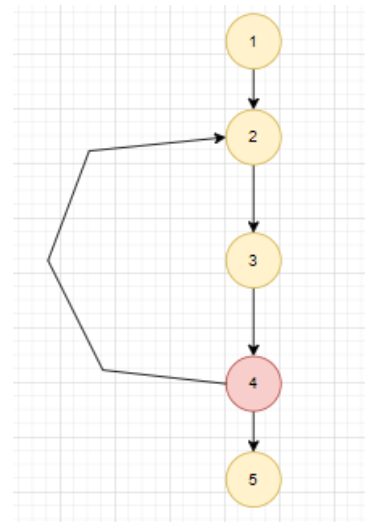
        if (client.Count == 0)
        {
            throw new ControlExcepcion("Cliente no encontrado");
        }
    }

    return client;
}
```



- CalcularTotalPagar()**

```
/// <summary>
/// Se realiza el metodo para obtener el total a pagar
/// </summary>
/// <param name="precio"></param>
/// <returns></returns>
public double calcularTotalPagar(List<double> precio)
{
    double totalPagar = 0;
    foreach(double p in precio)
    {
        totalPagar += p;
    }
    return totalPagar;
}
```

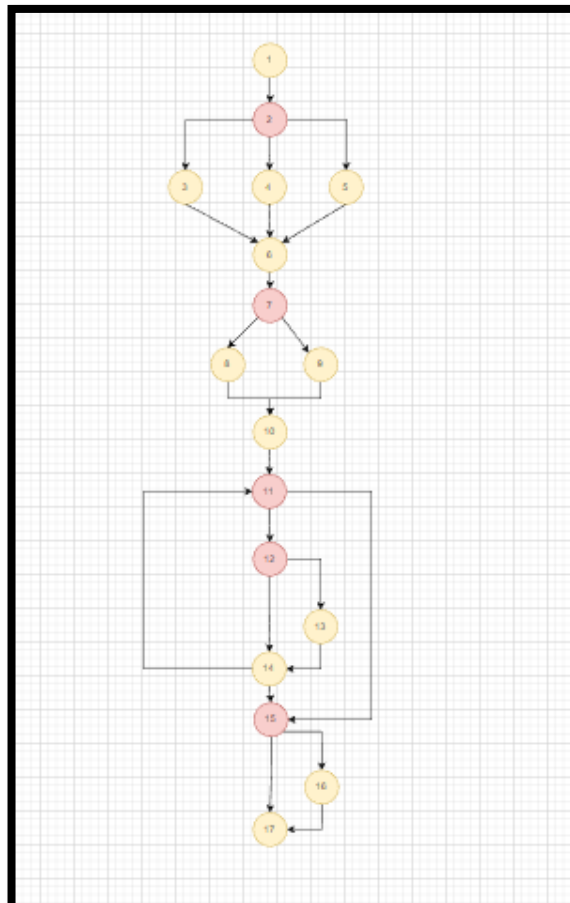


- LLenarDataGrid()

```

Conexion con = new Conexion();
ConectorDeProcedimientos conector = new ConectorDeProcedimientos();
string consulta = "";
switch (idConsulta)
{
    case 1: consulta = "BuscarReporteFecha";
            tiempo = Convert.ToDateTime(datao);
            break;
    case 2: consulta = "BuscarReporteCooperativa";
            break;
    case 3: consulta = "BuscarReporteCedula"; break;
    default: consulta = "BuscarReporte"; break;
}
try
{
    MySqlCommand mySqlCommand = conector.ConectarProcedimiento(consulta, con.conectar());
    if (idConsulta==1) mySqlCommand.Parameters.AddWithValue("@Data", tiempo);
    else mySqlCommand.Parameters.AddWithValue("@Data", datao);
    MySqlDataReader lector = mySqlCommand.ExecuteReader();
    bool x = true;
    while (lector.Read())
    {
        if (x) dataGridReporte.Rows.Clear();
        int numerofila = dataGridReporte.Rows.Count;
        dataGridReporte.Rows.Add(1);
        dataGridReporte.Rows[numerofila].Cells[0].Value = lector["Id_compra"].ToString();
        dataGridReporte.Rows[numerofila].Cells[1].Value = lector["Fecha_Compra"].ToString();
        dataGridReporte.Rows[numerofila].Cells[2].Value = lector["Fecha_Salida"].ToString();
        dataGridReporte.Rows[numerofila].Cells[3].Value = lector["CedulaComprador"].ToString();
        dataGridReporte.Rows[numerofila].Cells[4].Value = lector["Cedula_cliente"].ToString();
        dataGridReporte.Rows[numerofila].Cells[5].Value = lector["cooperativa"].ToString();
        dataGridReporte.Rows[numerofila].Cells[6].Value = lector["HoraSalida"].ToString();
        dataGridReporte.Rows[numerofila].Cells[7].Value = lector["Placa"].ToString();
        x = false;
    }
    con.cerrar();
    if (x)
    {
        throw new ControlExcepcion("No existe reporte");
    }
    return x;
}
catch
{
    throw new ControlExcepcion("No existe reporte");
}
}

```

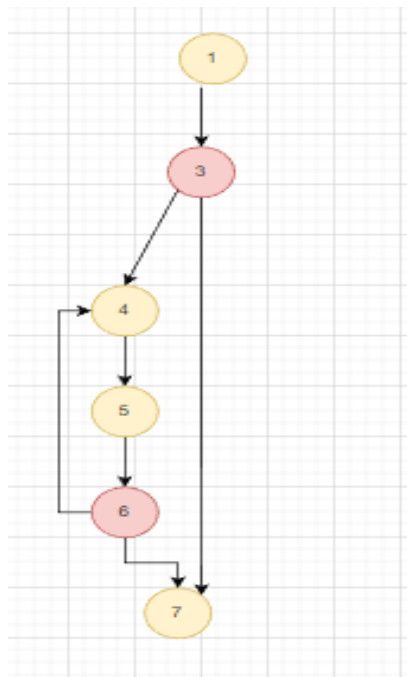


- LlenarComboAsiento ()

```

...
public void LlenarComboAsientos(int busId, ComboBox cbNumeroAsientos,int BoletoID)
{
    // Se crea una lista string llamada NumeroAsiento en la que se almacenara Los numeros de asientos de un bus en particular.
    List<string> NumeroAsiento = procedimientos.BuscarNumerosAsientos(busId, BoletoID);
    if (NumeroAsiento.Count != 0)
    {
        foreach (string asiento in NumeroAsiento)
        {
            cbNumeroAsientos.Items.Add(asiento);
        }
    }
}
}
}
}

```

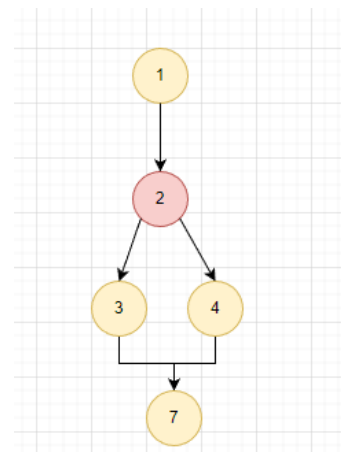


- Login()

```

{
    List<int> IdPersonaAndRol = procedimientos.IniciasSeccion(usuario, password);
    if(IdPersonaAndRol.Count > 0)
    {
        MessageBox.Show("Inicio de sesión realizado con éxito.");
    }
    else
    {
        throw new ControlExcepcion("Usuario y/o contrasena incorrecta");
    }
}
return IdPersonaAndRol;
}

```

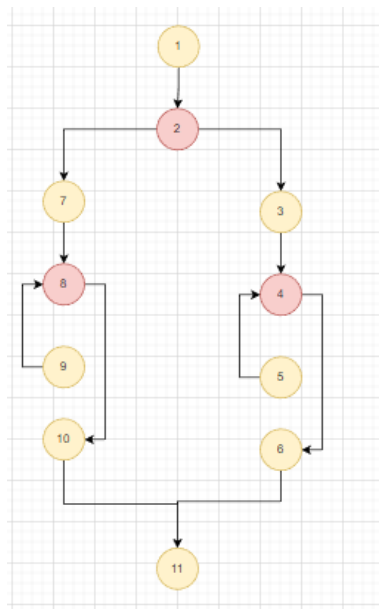


- LlenarCombo()

```

internal void LlenarCombos(int idCooperativa, Guna2ComboBox combo, int x)
{
    Conexion con = new Conexion();
    ConectorDeProcedimientos conector = new ConectorDeProcedimientos();
    if (x != 0)
    {
        Console.WriteLine("XXXXX" + x);
        try
        {
            Console.WriteLine(idCooperativa + " " + x);
            MySqlCommand mySqlCommand = conector.ConectarProcedimiento("spl_BuscarBus", con.conectar());
            mySqlCommand.Parameters.AddWithValue("@idCooperativa", idCooperativa);
            mySqlCommand.Parameters.AddWithValue("@idRuta", x);
            MySqlDataReader lector = mySqlCommand.ExecuteReader();
            while (lector.Read())
            {
                Console.WriteLine("jjjjjjjjjjjjjjjjjjjjjjjj");
                string anexar = lector["Placa"].ToString();
                combo.Items.Add(anexar);
            }
            con.cerrar();
        }
        catch (MySqlException ex)
        {
            Console.WriteLine("Error emitido por: " + ex);
        }
    }
    else
    {
        try
        {
            MySqlCommand mySqlCommand = conector.ConectarProcedimiento("spl_BuscarRuta", con.conectar());
            mySqlCommand.Parameters.AddWithValue("@idCooperativa", idCooperativa);
            MySqlDataReader lector = mySqlCommand.ExecuteReader();
            while (lector.Read())
            {
                string anexar = lector["idRuta"].ToString() + " De: " + lector["Salida"].ToString() + "
lector["Destino"].ToString();
                combo.Items.Add(anexar);
            }
            con.cerrar();
        }
        catch (MySqlException ex)
        {
        }
    }
}

```



- GuardarPdf()

```

public static List<string> GuardarPdf()
{
    List<string> list = new List<string>();
    string tablas = string.Empty;
    double precio=0;
    string cliente= string.Empty;
    Conexion con = new Conexion();
    ConectorDeProcedimientos conector = new ConectorDeProcedimientos();
    try
    {
        MySqlCommand mySqlCommand = conector.ConectarProcedimiento("ImprimirBoleto", con.conectar());
        mySqlCommand.Parameters.AddWithValue("@cantboleto", MNumeroboleto);
        MySqlDataReader lector = mySqlCommand.ExecuteReader();
        while (lector.Read())
        {
            tablas += "<tr>";
            tablas += "<td>" + lector["Id_compra"].ToString() + "</td>";
            tablas += "<td>" + lector["cooperativa"].ToString() + "</td>";
            tablas += "<td>" + lector["Placa"].ToString() + "</td>";
            tablas += "<td>" + lector["Cedula Comprador"].ToString() + "</td>";
            tablas += "<td>" + lector["Fecha_Salida"].ToString() + "</td>";
            tablas += "<td>" + lector["HoraSalida"].ToString() + "</td>";
            tablas += "<td>" + lector["Precio"].ToString() + "</td>";
            tablas += "</tr>";
            precio = precio + Convert.ToDouble(lector["Precio"]);
            MNumeroboleto--;
            cliente = lector["Nombre_cliente"].ToString();
        }
        list.Add(tablas);
        list.Add(Convert.ToString(precio));
        list.Add(cliente);
        con.cerrar();
    }
    catch (MySqlException ex)
    {
        Console.WriteLine("Error emitido por: " + ex);
    }
    return list;
}
}

```

