

UNIVERSIDAD PRIVADA-DE-TACNA



INGENIERIA DE SISTEMAS

TITULO:

Trabajo Final de Unidad

CURSO:

BASE DE DATOS II

DOCENTE(ING):

Patrick Cuadros Quiroga

Integrantes:

Orlando Antonio Acosta Ortiz	(2015052775)
Orestes Ramirez Ticona	(2015053236)
Nilson Laura Atencio	(2015053846)
Roberto Zegarra Reyes	(2010036175)
Richard Cruz Escalante	(2013047247)
Wilfredo Vilca Chambilla	(2006028540)

Índice

1. PROBLEMÁTICA	1
2. MARCO TEÓRICO	2
2.1. Entity Framework	2
2.2. API REST	2
2.3. Consultas LINQ	2
2.4. Pruebas Unitarias	3
3. DESARROLLO	4
3.1. Desarrollo del Sistema	4
3.2. Analisis	11
3.3. Diseño	12
3.4. Pruebas	12
4. ANALISIS E INTERPRETACION DE RESULTADOS	15
5. CONCLUSIONES	18

1. PROBLEMÁTICA

El siguiente trabajo se desarrolla con el Sistema de Cajero Automatico, el cual busca que el cualquier cajero del Banco Pichincha pueda realizar acciones de forma mas automatizada y con toda seguridad:

- La Gestion de Cajero, Retiros, Depositos, Consultas
- Gestión de las usuarios que cuenten con una cuenta en el banco PICHINCHA.
- La venta de distintos tipos de planes.
- La realización de reportes sobre las transacciones realizadas del día, del mes, y sobre los clientes correspondientes a cada local.
- El sistema permitirá realizar la autenticación de los usuarios.

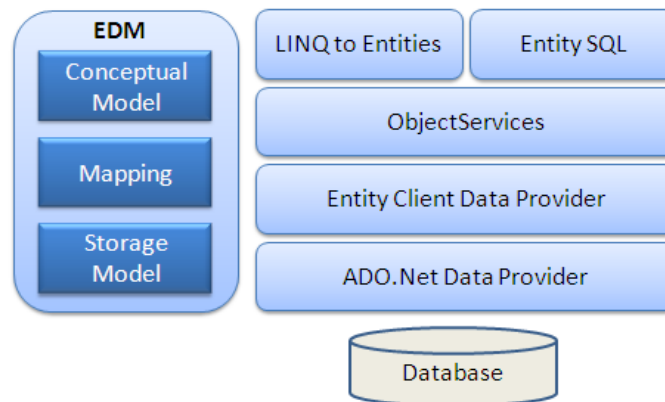
La motivación principal de este proyecto es que los cajeros del Banco Pichincha tengan una automatizacion mas eficaz, lo que le permitirá realizar sus tareas sin inconvenientes.

2. MARCO TEORICO

2.1. Entity Framework

Entity Framework es un marco de ORM de código abierto para aplicaciones .NET admitidas por Microsoft. Permite a los desarrolladores trabajar con datos utilizando objetos de clases específicas del dominio sin centrarse en las tablas y columnas de la base de datos subyacente donde se almacenan estos datos. Con el Entity Framework, los desarrolladores pueden trabajar en un nivel más alto de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código en comparación con las aplicaciones tradicionales.[2]

Su arquitectura:



2.2. API REST

Ha pasado más de una década desde que Roy Fielding, un científico informático estadounidense y uno de los autores principales de la especificación HTTP, introdujo Representational State Transfer (REST) como un estilo de arquitectura. A lo largo de los años, REST ha ganado impulso gracias a su popularidad para la creación de servicios web.

Al no tener estado, y al crear identificadores únicos para permitir el almacenamiento en caché, la creación de capas y la capacidad de lectura, las API REST hacen uso de los verbos HTTP existentes (GET, POST, PUT y DELETE) para crear, actualizar y eliminar nuevos recursos. El término REST se usa con frecuencia para describir cualquier URL que devuelva JSON en lugar de HTML.

2.3. Consultas LINQ

Permite consultar cualquier objeto enumerable en ADO.NET mediante el uso del modelo de programación de Language-Integrated Query (LINQ) [1] para recuperar datos de la base de datos subyacente. El proveedor de la base de datos traducirá estas consultas LINQ al lenguaje de consulta específico de la base de datos (por ejemplo, SQL para una base de datos relacional).

Hay 3 tecnologías ADO.NET LINQ distintas:

- LINQ to DataSet: proporciona una capacidad de consulta mas rica y optimizada sobre DataSet.

- LINQ to SQL: permite consultar directamente los esquemas de las base de datos de SQL Server.
- LINQ to Entities: permite consultar Entity Data Model.

2.4. Pruebas Unitarias

Las pruebas unitarias son la mejor forma de probar el código de una aplicación a lo largo de su desarrollo. Estas pruebas permiten asegurar que los métodos devuelven resultados correctos, teniendo en cuenta los argumentos que se le pasan y, además, se pueden utilizar para hacer Test Driven Development. Se trata de una técnica en la que se escriben antes que las clases y los métodos.[4]

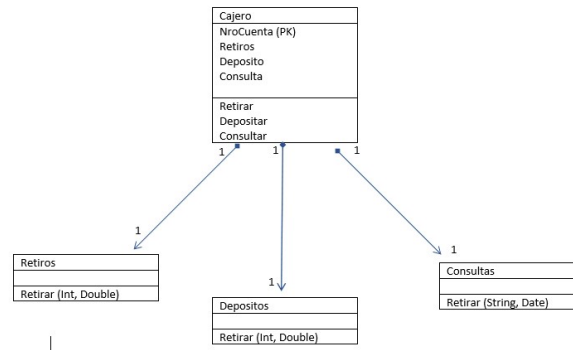
No obstante, tras realizar la integración con otros módulos deberá revisarse de nuevo la interfaz.

3. DESARROLLO

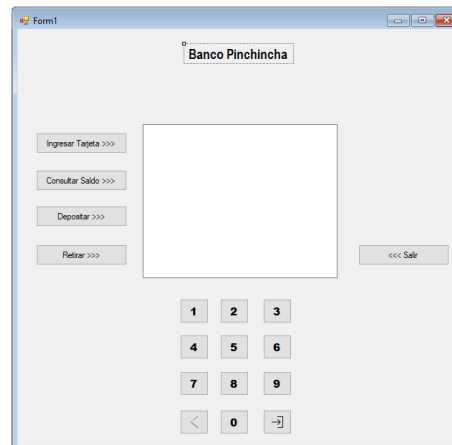
3.1. Desarrollo del Sistema

1. Pasos de desarrollo de la aplicación

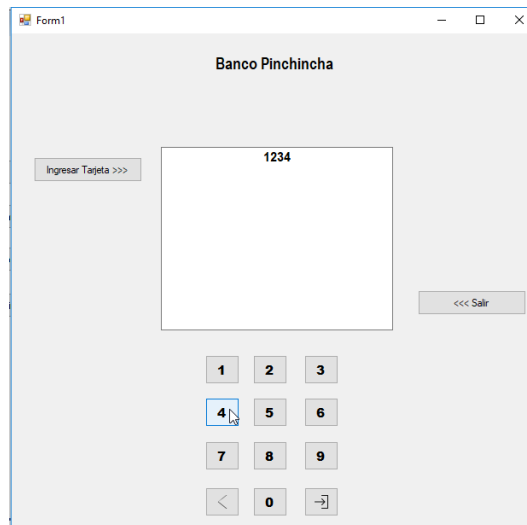
a) Paso 1: Diagrama de Clases de solución Cajero



b) Paso2: Crear un Windows form para desarrollar nuestra aplicación de escritorio:



c) Paso 3: Codificar los botones según lo planeado para nuestra aplicación, por ejemplo para esta aplicación hemos decidido virtualizar un cajero automático.



d) Paso 4: Muestra de algo de código de los botones:

```

1 referencia
private void Button1_Click(object sender, EventArgs e)
{
    TxtPantalla.Text = TxtPantalla.Text + "1";
}

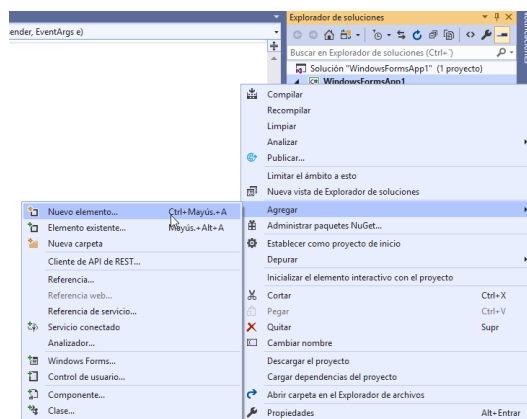
1 referencia
private void BtnMiro2_Click(object sender, EventArgs e)
{
    TxtPantalla.Text = TxtPantalla.Text + "2";
}

1 referencia
private void BtnMiro3_Click(object sender, EventArgs e)
{
    TxtPantalla.Text = TxtPantalla.Text + "3";
}

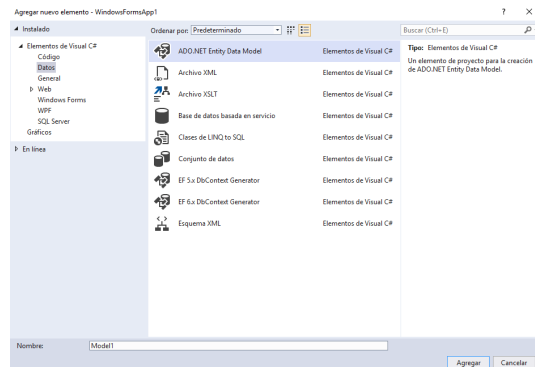
1 referencia
private void BtnMiro4_Click(object sender, EventArgs e)
{
    TxtPantalla.Text = TxtPantalla.Text + "4";
}

```

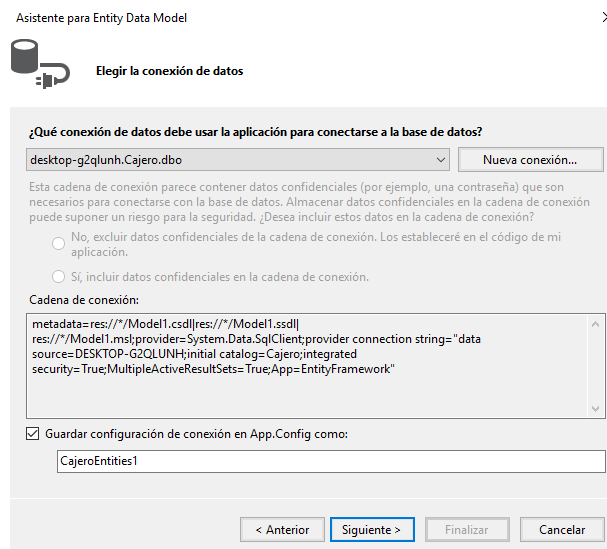
e) Paso 5: aplicar un ORM a nuestra aplicación: aplicaremos el ORM Entity framework(define que es y para que sirve entity framework)



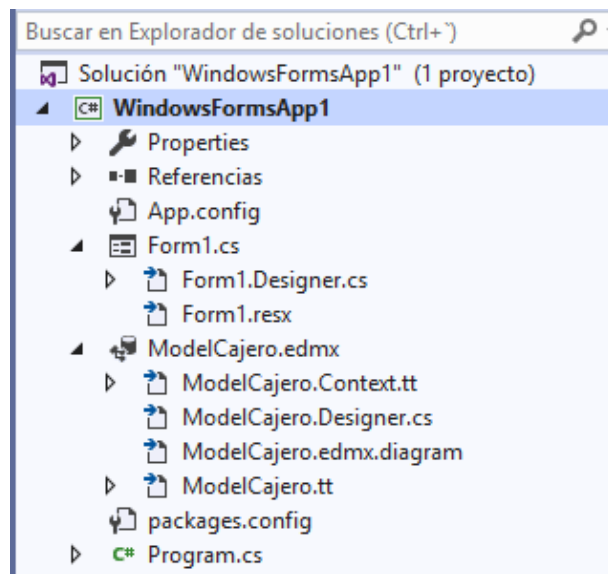
f) Paso 6: Agregaremos el modelo ADO.NET entity data model



g) Paso 7: ELuego configuraremos con la base de datos que se encuentra en Nuestro servidor SQL



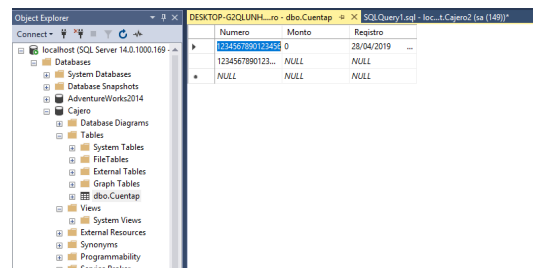
h) Paso 8: una vez configurado podremos comprobar que ya podemos interactuar con la base de datos a través Entity Framework.



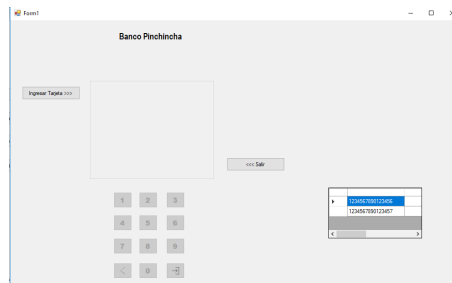
- Después escribir la siguiente consulta.

```
1 referencia
public void llamarDatos()
{
    using (CajeroEntities db = new CajeroEntities())
    {
        var list = db.Cuentas;
        foreach (var cuenta in list)
        {
            dgvDatos.ColumnCount = 4;
            dgvDatos.Rows.Add(cuenta.Numero);
        }
    }
}
```

- i) Paso 9: Demostraremos el de entity framework con un Login. Para lograr esto liste los datos de Tarjeta que se encuentran en la base de datos en un DataGridView (Por motivos de Seguridad este DataGridView no será visible para el Usuario) con el siguiente código:



- j) Paso 10: Demostraremos el de entity framework con un Login. Para lograr esto liste los datos de Tarjeta que se encuentran en la base de datos en un DataGridView (Por motivos de Seguridad este DataGridView no será visible para el Usuario) con el siguiente código:



k) Paso 11: Una vez obtenidos los datos podremos confirmar si el usuario existe o no en la base de datos: Código:

```

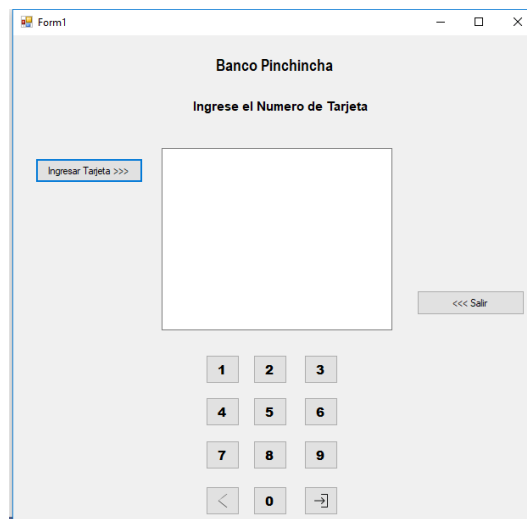
1 referencia
public void Logear()
{
    using (CajeroEntities db = new CajeroEntities())
    {
        NroCuenta = TxtPantalla.Text;

        for (int i = 0; i < DgvDatos.RowCount; i++)
        {
            string dato;
            dato = Convert.ToString(DgvDatos.Rows[i].Cells[0].Value);

            if (NroCuenta == dato)
            {
                lblGuia.Text = "Bienvenido";
                MostrarBotones();
                break;
            }
            else
            {
                lblGuia.Text = "Usted No esta Registrado";
                OcultarBotones();
            }
        }
    }
}

```

l) Paso 12: Se muestra el formulario principal del cajero



m) Paso 13: Si existe el formulario

The screenshot shows a Windows application window titled "Form1". Inside the window, the text "Banco Pinchincha" is centered at the top. Below it, the word "Bienvenido" is displayed. On the left side, there is a vertical stack of four buttons: "Ingresar Tarjeta >>>", "Consultar Saldo >>>", "Depositar >>>", and "Retirar >>>". In the center, a white rectangular box displays the card number "1234567890123456". To the right of this box is a button labeled "<<< Salir". At the bottom of the window, there is a numeric keypad with buttons for digits 1 through 9, 0, a left arrow, and a right arrow. The right arrow button is highlighted with a blue border.

n) Paso 14: Si en caso que no existe

Form1

Banco Pinchinch

Usted No esta Registrado

Ingresar Tarjeta >>>

1234567890123458

<<< Salir

1

2

3

4

5

6

7

8

9

<

0

→

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
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
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

	Numero	Monto	Registro
	1234567890123456	150	28/04/2019 ...
	1234567890123457	200	NULL
»*	NULL	NULL	NULL

Form1

Banco Pinchinch

Bienvenido 1234567890123456

Ingresar Tarjeta >>>

Consultar Saldo >>>

Depositar >>>

Retirar >>>

Usted Tiene : 150 Soles.

<<< Salir

1

2

3

4

5

6

7

8

9

<

0

→

Form1

Banco Pinchinch

Bienvenido 1234567890123457

Ingresar Tarjeta >>>

Consultar Saldo >>>

Depositar >>>

Retirar >>>

Usted Tiene : 200 Soles.

<<< Salir

1

2

3

4

5

6

7

8

9

<

10

→

Banco Pinchinch

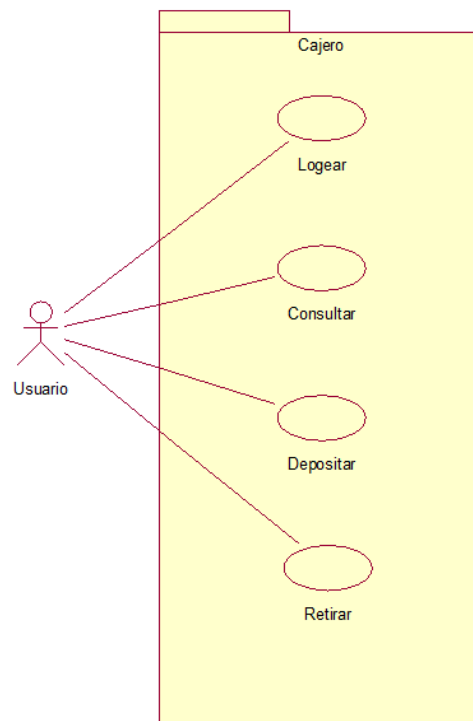
- ñ) Paso 14: Y así hemos demostrado la funcionalidad con un ORM. Ha quedado demostrado la simplicidad del código que hemos usado.

Numero	Monto	Registro
1234567890123456	150	28/04/2019 ...
1234567890123457	0	28/04/2019 ...
NULL	NULL	NULL

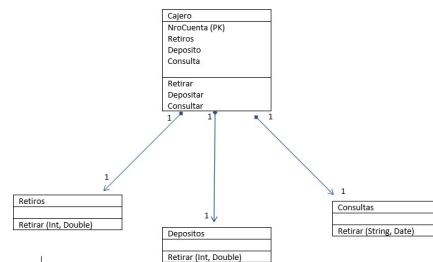
3.2. Analisis

1. Escribiendo consultas con el operador PIVOT

a) Analisis de desarrollo del Sistema - Caso de uso del Sistema



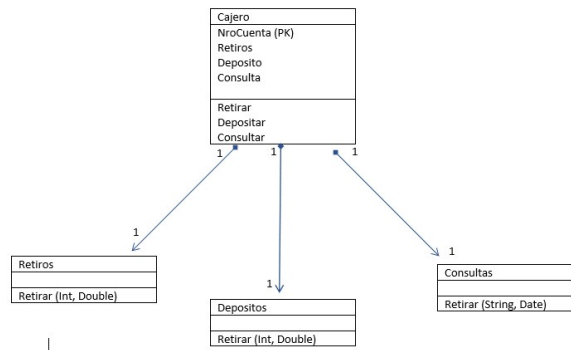
- Diagrama de Entidad Relacion



3.3. Diseño

1. Diagrama de Clases, Modelo Entidad Relación

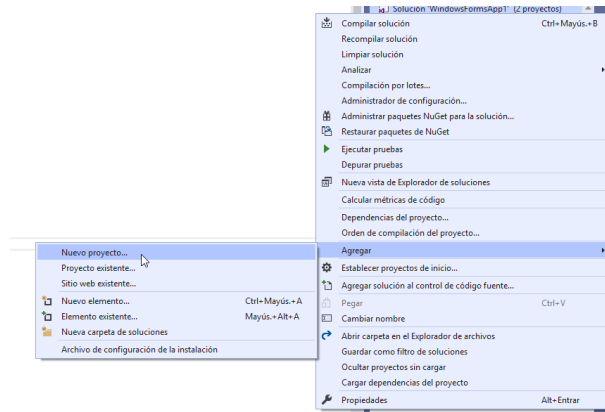
- a) El diagrama de entidad Relacion fue desarrollado con el fin de que la base de datos tenga una estructura relacional optima



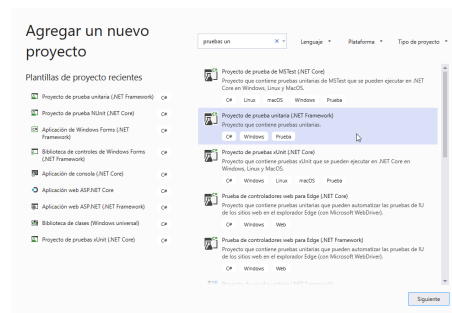
3.4. Pruebas

1. Pruebas Unitarias

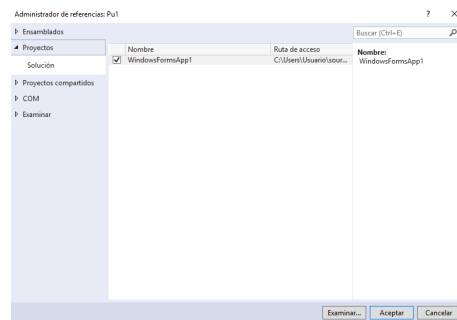
- a) Paso 1: Pruebas unitarias: haremos pruebas unitarias para los métodos deposito, retiro.
- Agregaremos un test de pruebas unitarias



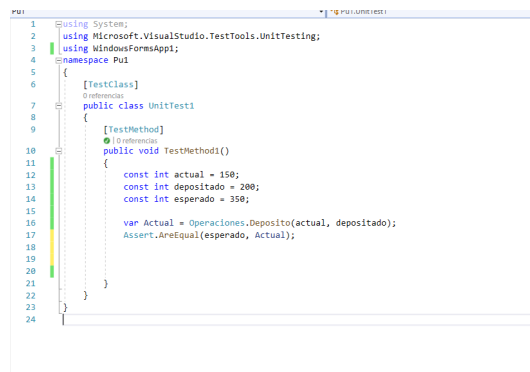
- b) Paso 2: Seguidamente el nuevo y tipo de proyecto.



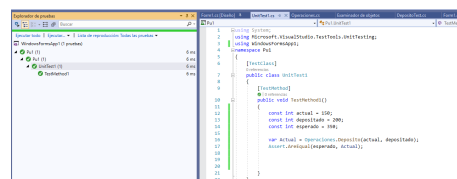
c) Paso 3: Luego referenciaremos el proyecto principal



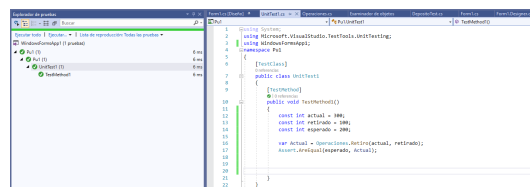
d) Paso 4: Luego probaremos los métodos :



e) Paso 5: Esperamos mientras carga la prueba que se va desarrollar



f) Paso 6: Retiros



g) Paso 7: Finalmente aplicamos el código que se ejecutará

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using WindowsFormsApp1;

namespace Pu1
{
    [TestClass]
    [0 referencias]
    public class UnitTest1
    {
        [TestMethod]
        [0 referencias]
        public void TestMethod1()
        {
            const int actual = 300;
            const int retirado = 100;
            const int esperado = 200;

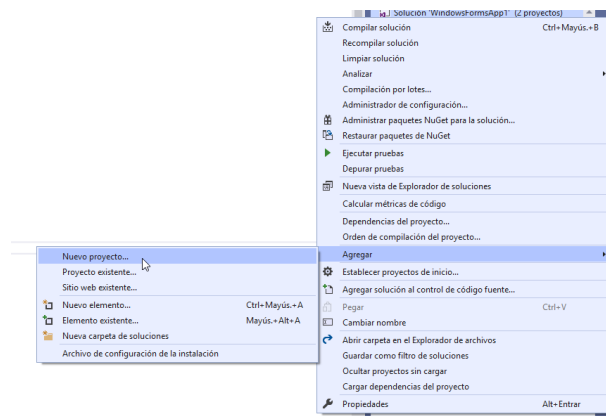
            var Actual = Operaciones.Retiro(actual, retirado);
            Assert.AreEqual(esperado, Actual);
        }
    }
}
```


4. ANALISIS E INTERPRETACION DE RESULTADOS

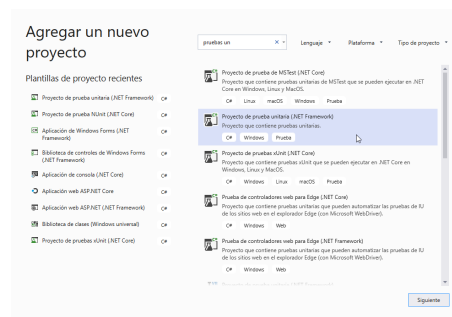
-Al compilar el proyecto de prueba, las pruebas aparecen en el Explorador de pruebas. Si el Explorador de pruebas no está visible, elija Prueba en el menú de Visual Studio, elija Ventanas y, después, Explorador de pruebas. -Se puede elegir Ejecutar todas para ejecutar todas las pruebas o bien Ejecutar para elegir un subconjunto de pruebas que se desea ejecutar. Después de ejecutar un conjunto de pruebas, aparecerá un resumen de la serie de pruebas en la parte inferior de la ventana Explorador de pruebas. Seleccione una prueba para ver los detalles de esa prueba en el panel inferior.

1. Pruebas Unitarias

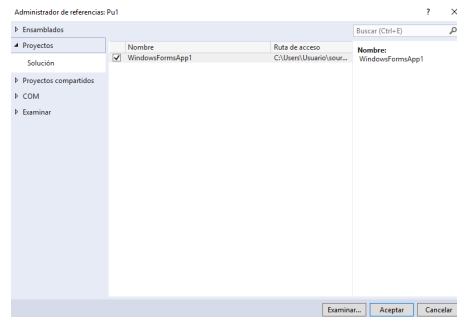
- a) Paso 1: Pruebas unitarias: haremos pruebas unitarias para los métodos deposito, retiro.
- Agregaremos un test de pruebas unitarias



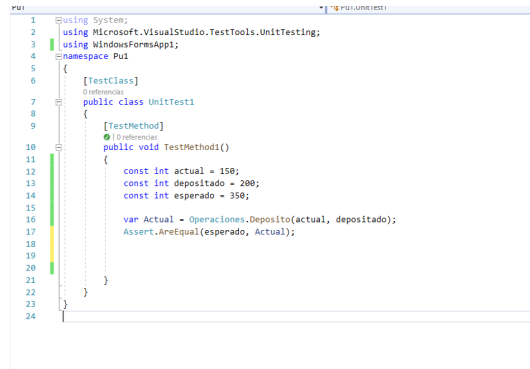
- b) Paso 2: Seguidamente el nuevo y tipo de proyecto.



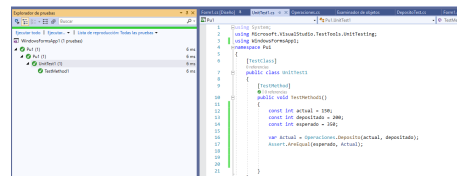
- c) Paso 3: Luego referenciaremos el proyecto principal



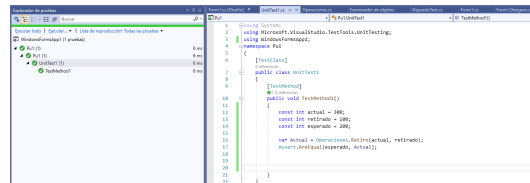
d) Paso 4: Luego probaremos los métodos :



e) Paso 5: Esperamos mientras carga la prueba que se va desarrollar



f) Paso 6: Retiros



g) Paso 7: Finalmente aplicamos el código que se ejecutará

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using WindowsFormsApp1;

namespace Pu1
{
    [TestClass]
    [0 referencias]
    public class UnitTest1
    {
        [TestMethod]
        [0 referencias]
        public void TestMethod1()
        {
            const int actual = 300;
            const int retirado = 100;
            const int esperado = 200;

            var Actual = Operaciones.Retiro(actual, retirado);
            Assert.AreEqual(esperado, Actual);
        }
    }
}
```

5. CONCLUSIONES

- [1] Torres, M. (2012). PROGRAMACION ORIENTADA A OBJETOS CON VISUAL BASIC 2012. Editorial Macro. Pág. 370[[1]
- [2] Elman, J. y Lavin, M. (2014). Django ligero: utilizando REST, WebSockets y Backbone. Editor .o'Reilly Media, Inc. Pág. 61
- [3] Hugon, J. (2014). C# 5: desarrolle aplicaciones Windows con Visual Studio 2013. Ediciones ENI. Pág. 269