

Actividad 2 – Conexión y Tablas

Lenguajes de Programación II

Ingeniería en Desarrollo de Software

Tutor: Miguel Ángel Rodríguez Vega

Alumno: José Manuel Ramos Vega

Fecha: 04 de diciembre de 2023



Índice


Introducción.....	3
Descripción	4
Justificación.....	5
Desarrollo	6
Conexión.....	4
Tablas.....	7
Código.....	9
Conclusión	14
Referencias.....	15

1- Introducción

Una base de datos bien diseñada brinda a los usuarios acceso a información fundamental. Al seguir los principios de esta página, puedes diseñar una base de datos que funcione bien y se adapte a tus necesidades futuras. Explicaremos los aspectos básicos sobre el diseño de una base de datos y cómo perfeccionarlo para obtener resultados óptimos.

Cuando tus tablas de base de datos se conviertan en tablas, estarás listo para analizar las relaciones entre esas tablas. La cardinalidad se refiere a la cantidad de elementos que interactúan entre dos tablas relacionadas. Identificar la cardinalidad te ayuda a asegurarte de que has dividido los datos en tablas de la forma más eficiente.

Cada entidad puede, potencialmente, tener una relación con todas las demás. Una relación redundante es aquella que se expresa más de una vez. Por lo general, puedes eliminar una de las relaciones sin perder información importante. Por ejemplo, si la entidad "estudiantes" tiene una relación directa con otra entidad llamada "profesores", pero también tiene una relación con profesores indirectamente mediante "clases", querrás eliminar la relación entre "estudiantes" y "profesores".



2- Descripción

Se necesita una estructura de clases que permita a la empresa UNI controlar los distintos tipos de empleados, así como sus datos personales. Esto se hará a través de clases, herencia de clases y atributos. Las clases, por su parte, deberán ser usadas desde una aplicación donde se gestione la siguiente información: ● Número de Empleado: (autogenerado, numérico) ● Nombre:

(capturable, alfanumérico) ● Apellido Paterno: (capturable, alfanumérico) ● Apellido Materno: (capturable, alfanumérico) ● Fecha de Nacimiento: (capturable tipo fecha) ● RFC: (calculado conforme al nombre y fecha de nacimiento, alfanumérico) ● Centro de Trabajo: (capturable, alfanumérico, elegible desde el número de clave

con base en el catálogo de puestos)

- Puesto: (capturable, alfanumérico)

- Descripción del Puesto: (capturable, alfanumérico)

- Directivo: (bandera para indicar tipo de empleado; para directivo 1; para empleado

normal 0) Existe un tipo de empleado denominado Directivo, el cual presenta, además de las cualidades anteriores, atributos particulares de su tipo.

Los atributos de los directivos son: ● Número del centro que supervisa (numérico, capturable) ●

Prestación de combustible (bandera que indica si el directivo recibe apoyo de combustible).

3- Justificación

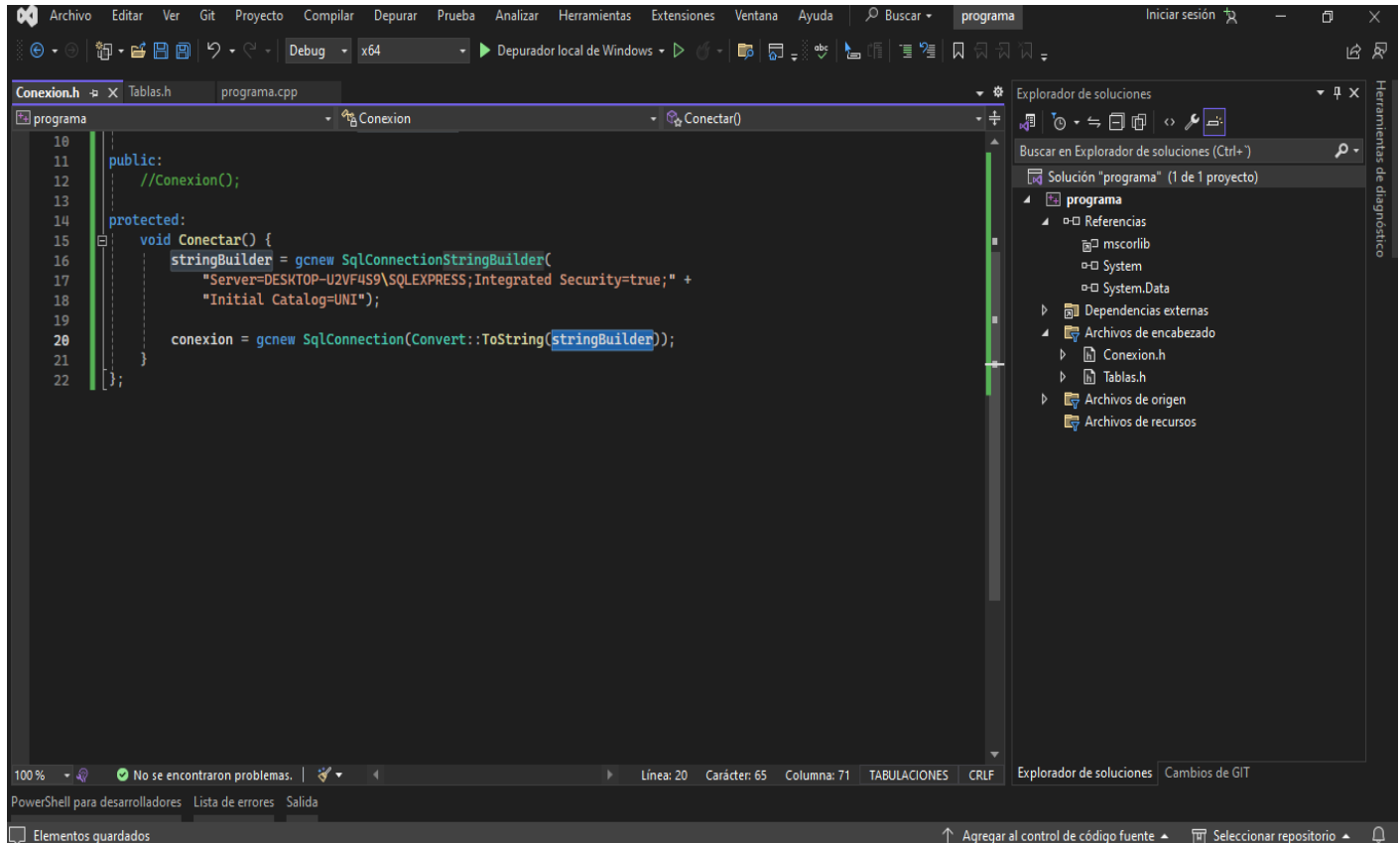
Aunque las particularidades de los datos espaciales con los que trabajamos en un SIG han hecho necesarias modificaciones y adaptaciones sobre el esquema de trabajo de las bases de datos genéricas, en esencia los fundamentos de estas siguen constituyendo el elemento primordial sobre el que la arquitectura de gestión de datos espaciales se apoya. En esta sección, veremos de forma introductoria esos fundamentos de bases de datos genéricas, aplicables a cualquier otro ámbito además del de los SIG, para posteriormente poder tratar el caso particular de los datos espaciales. La situación real, sin embargo, es habitualmente mucho más compleja, y utilizar un esquema de colaboración como el anterior puede ser imposible, carecer por completo de sentido, o tener un buen número de consecuencias negativas. A medida que aumenta el número de usuarios, resulta menos recomendable que cada uno trabaje con sus propios datos y se los hagan llegar entre ellos a medida que los necesitan (una realidad que, desgraciadamente, se presenta con más frecuencia de lo recomendable). No debe olvidarse que un conjunto más amplio de usuarios que trabajan de esta forma y son ellos mismos quienes gestionan sus propios datos, implica directamente un número también más elevado de aplicaciones informáticas y de formatos de archivo, complicando enormemente el trabajo coordinado en cuanto el equipo tiene un tamaño medio.

4- Desarrollo

Link del Código: <https://github.com/Joseramos28/LENGUAJESDEPROGRAMACION2.git>

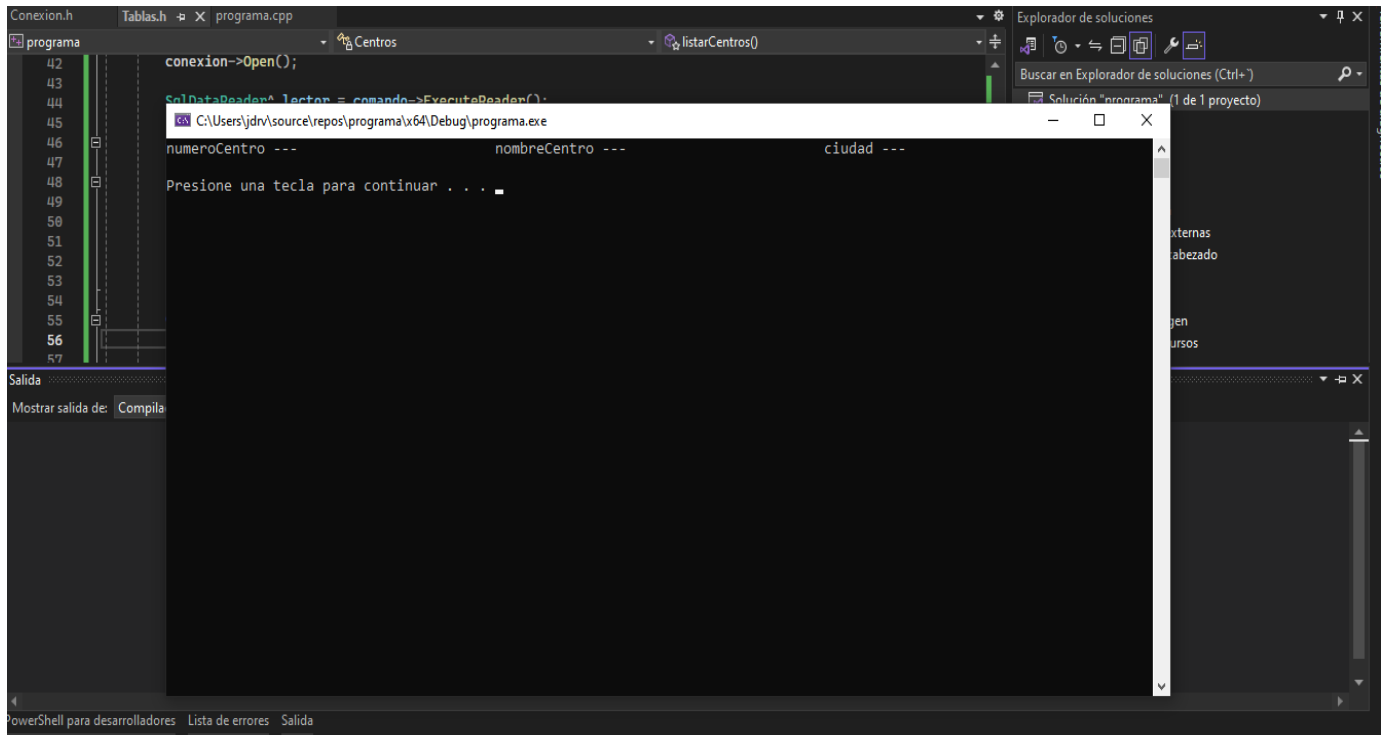
-Conexión:

En esta imagen demuestro la captura que se hizo de la conexión

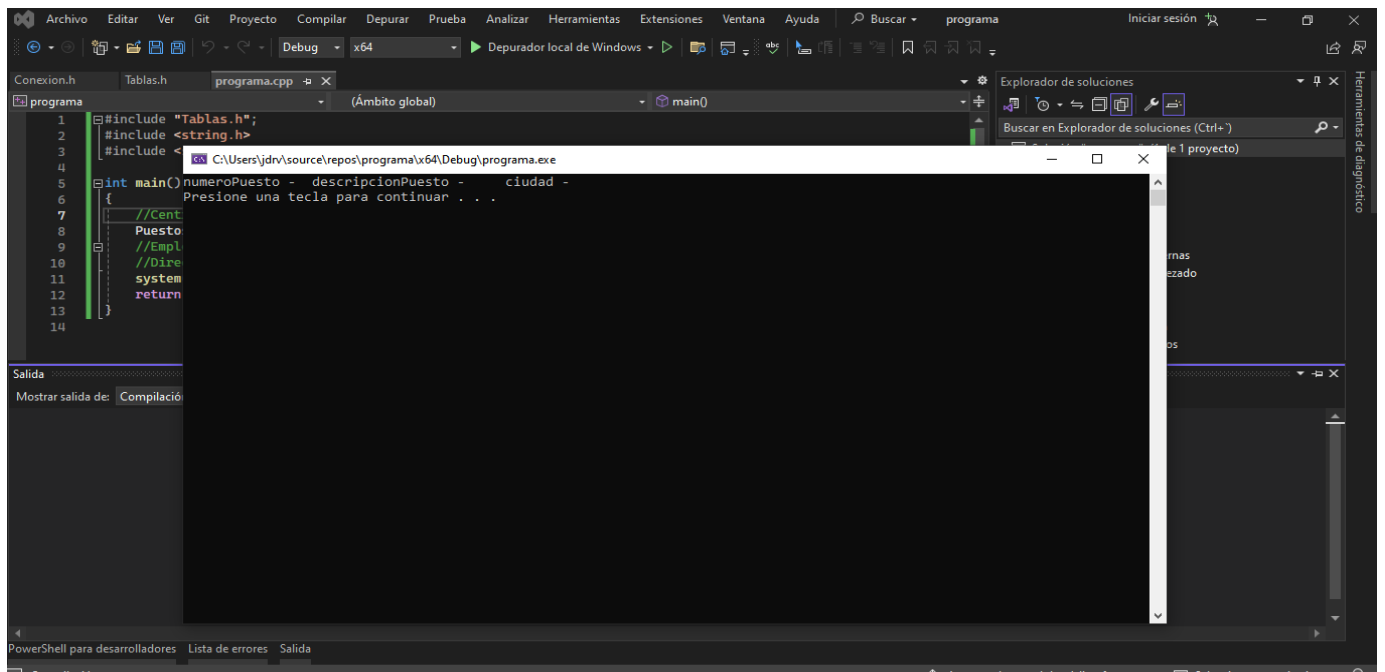


-Tablas:

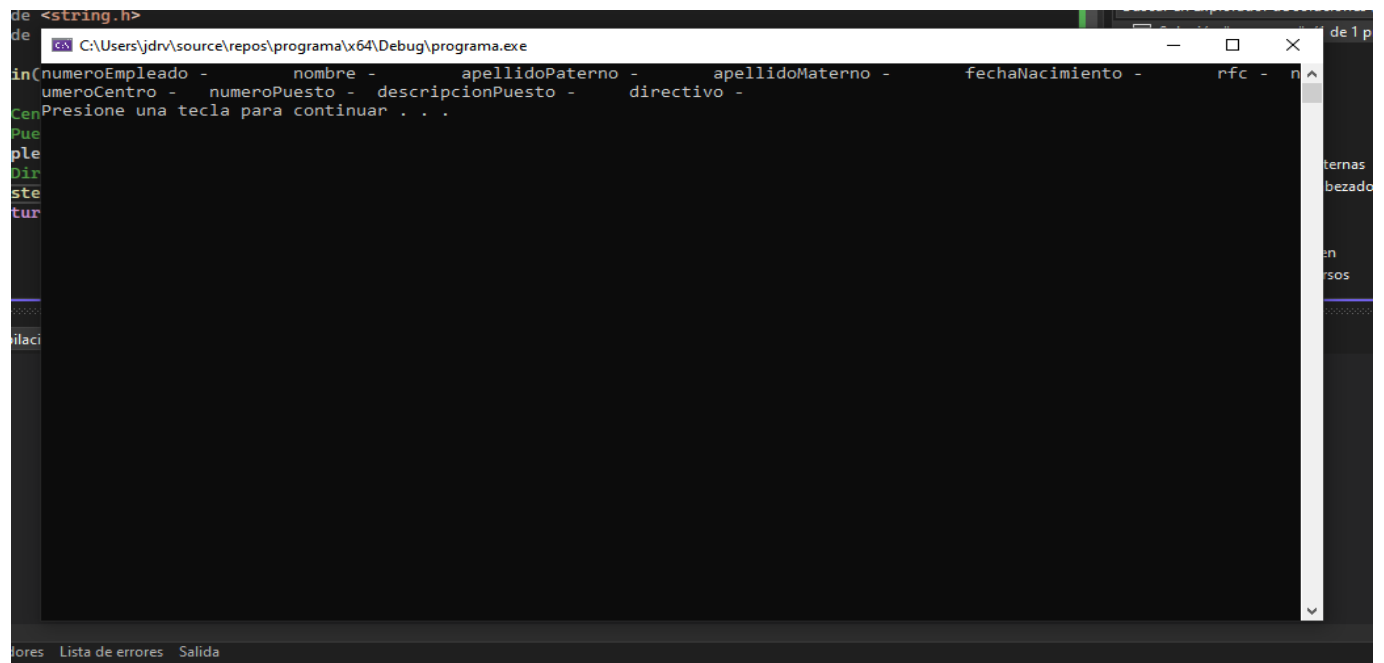
Centros



Puestos

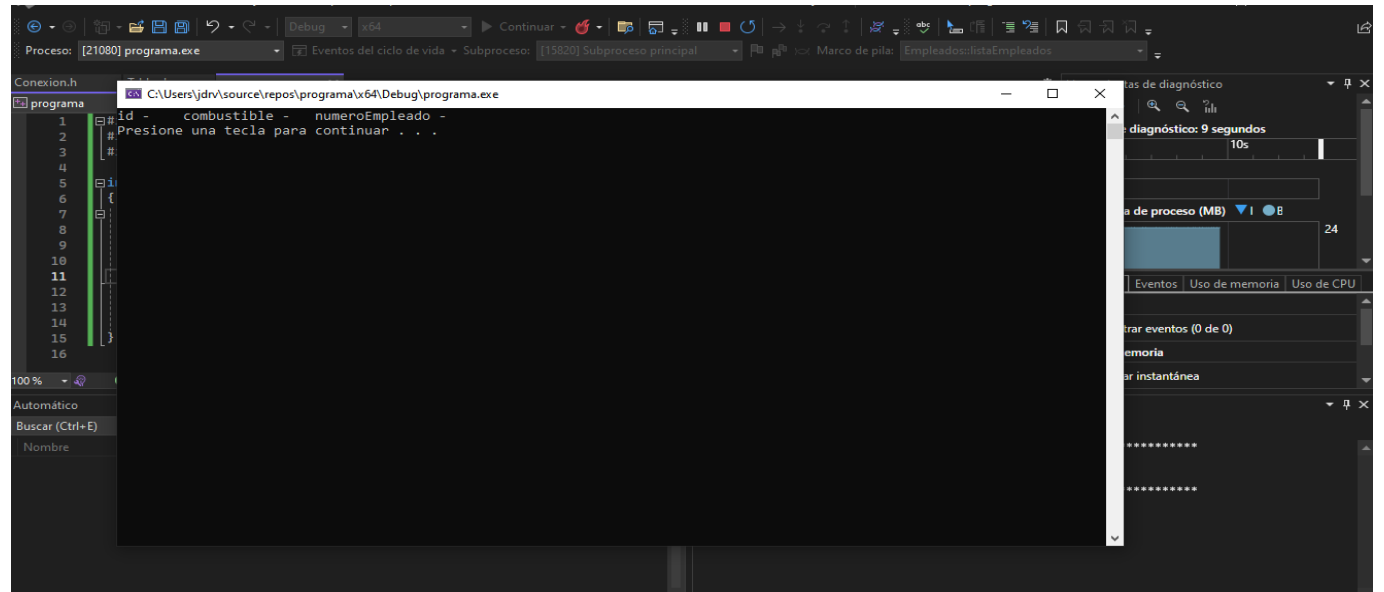


Empleados



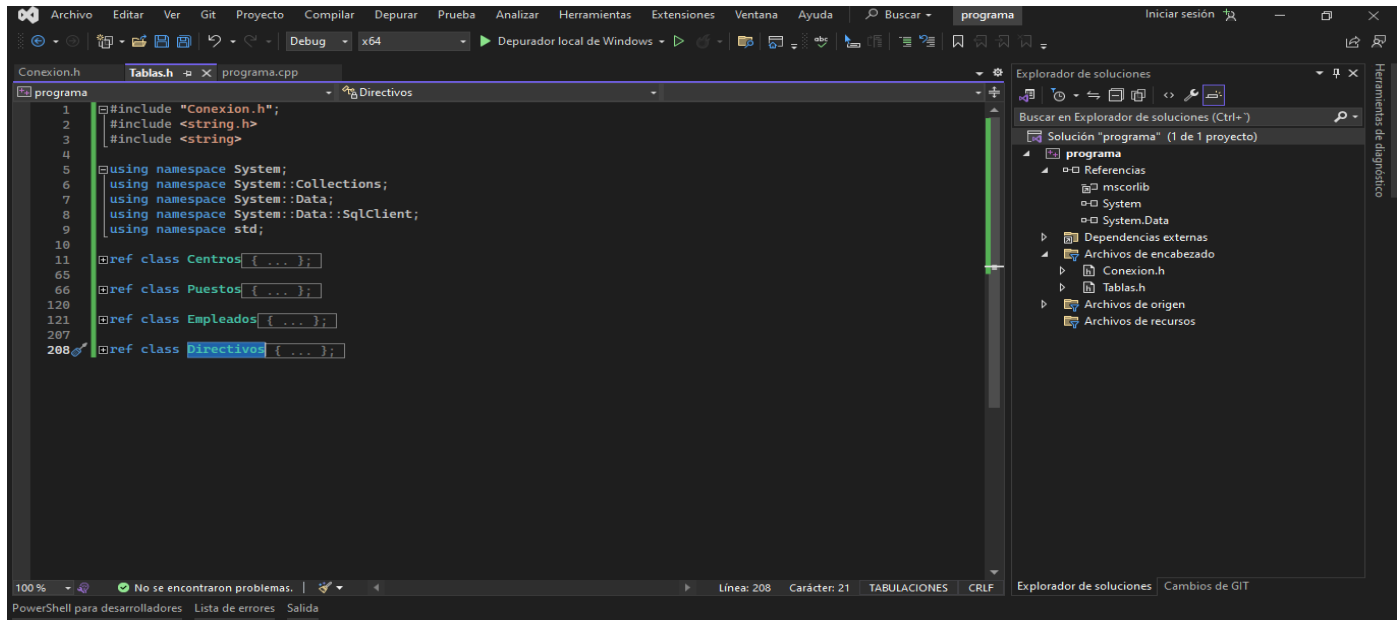
```
de <string.h>
de
C:\Users\jdrv\source\repos\programa\x64\Debug\programa.exe
in(numeroEmpleado - nombre - apellidoPaterno - apellidoMaterno - fechaNacimiento - rfc - n
umeroCentro - numeroPuesto - descripcionPuesto - directivo -
Presione una tecla para continuar . . .
```

Directivos

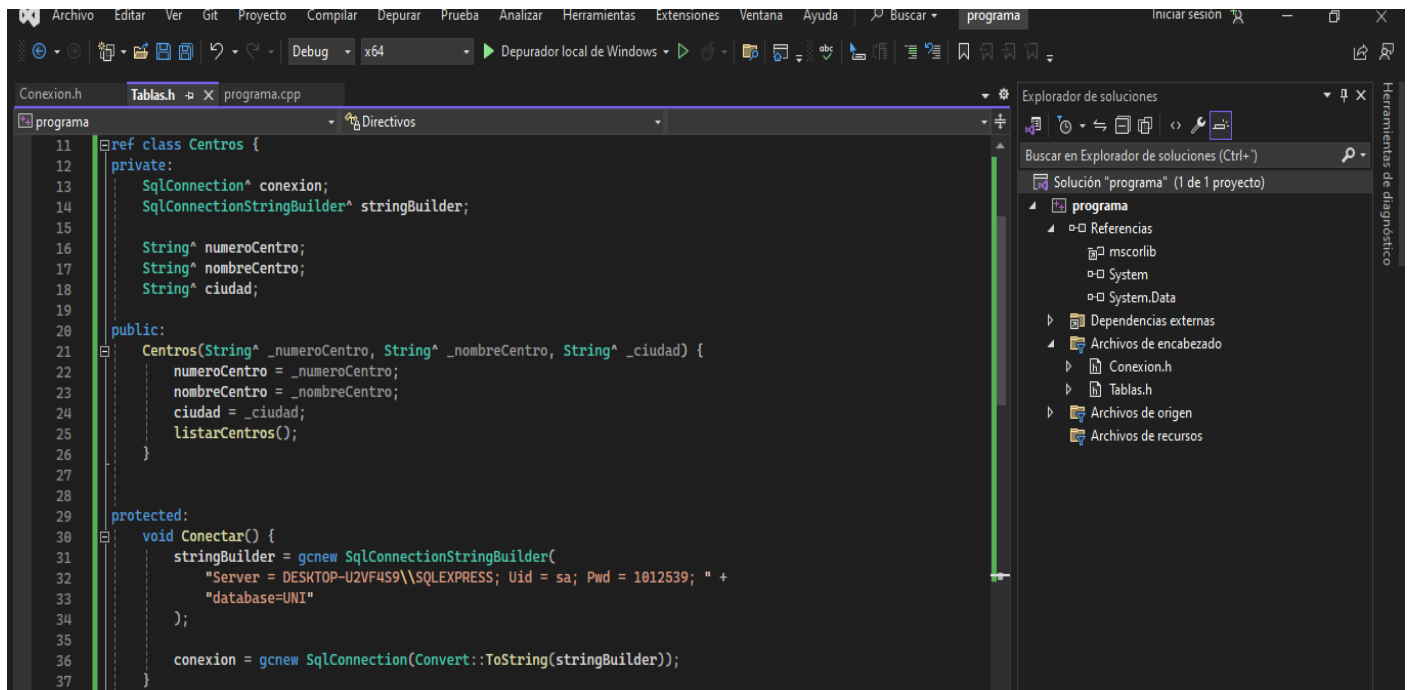


-Código:

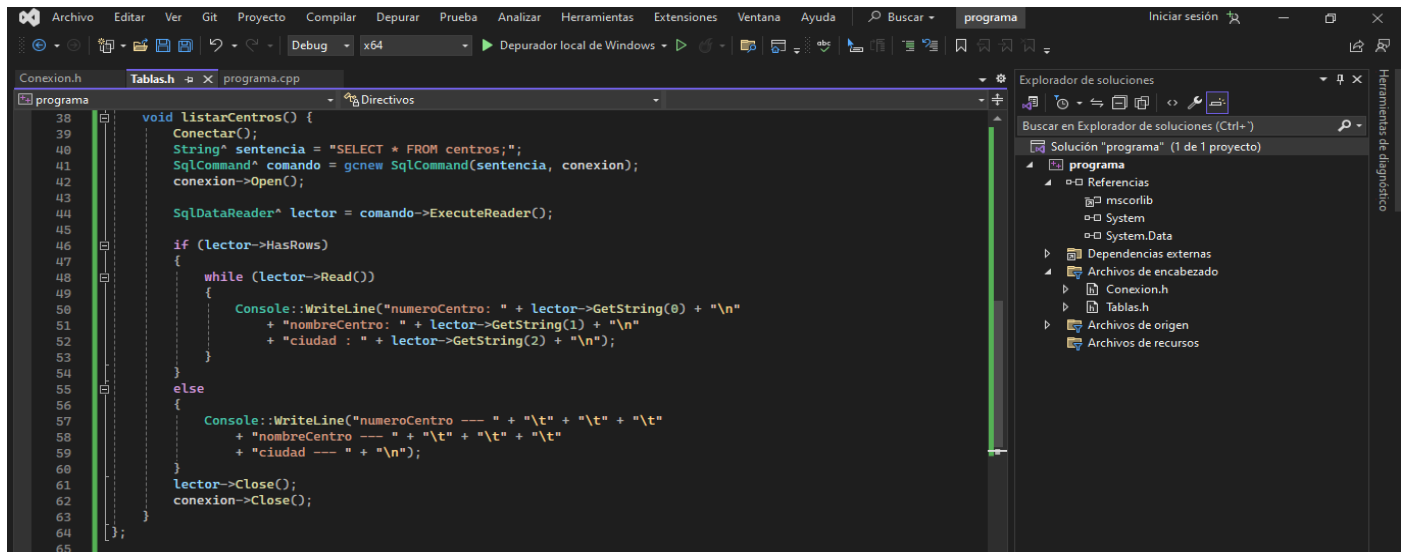
archivo Conexión.h con una clase por cada tabla.



En cada clase se crean los atributos y el constructor

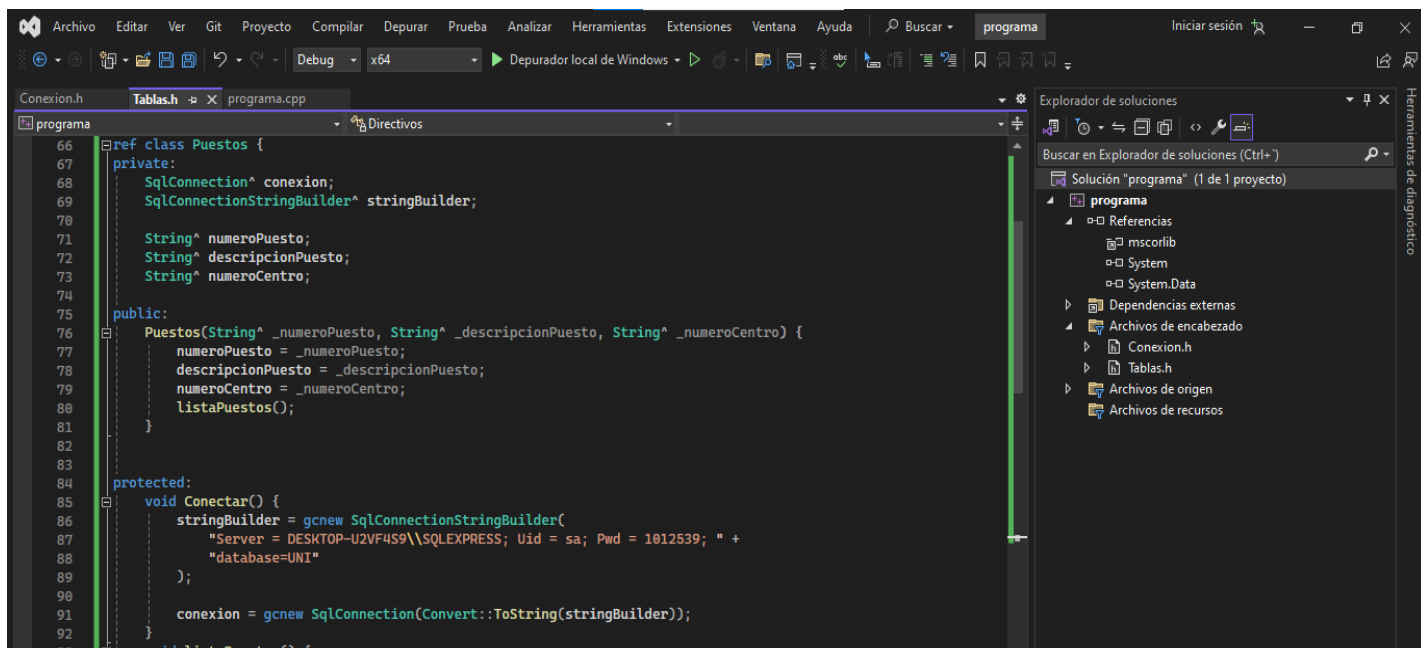


Posteriormente se hace el metodo para leer la informacion de la tabla. (Lo mismo para las demas tablas)

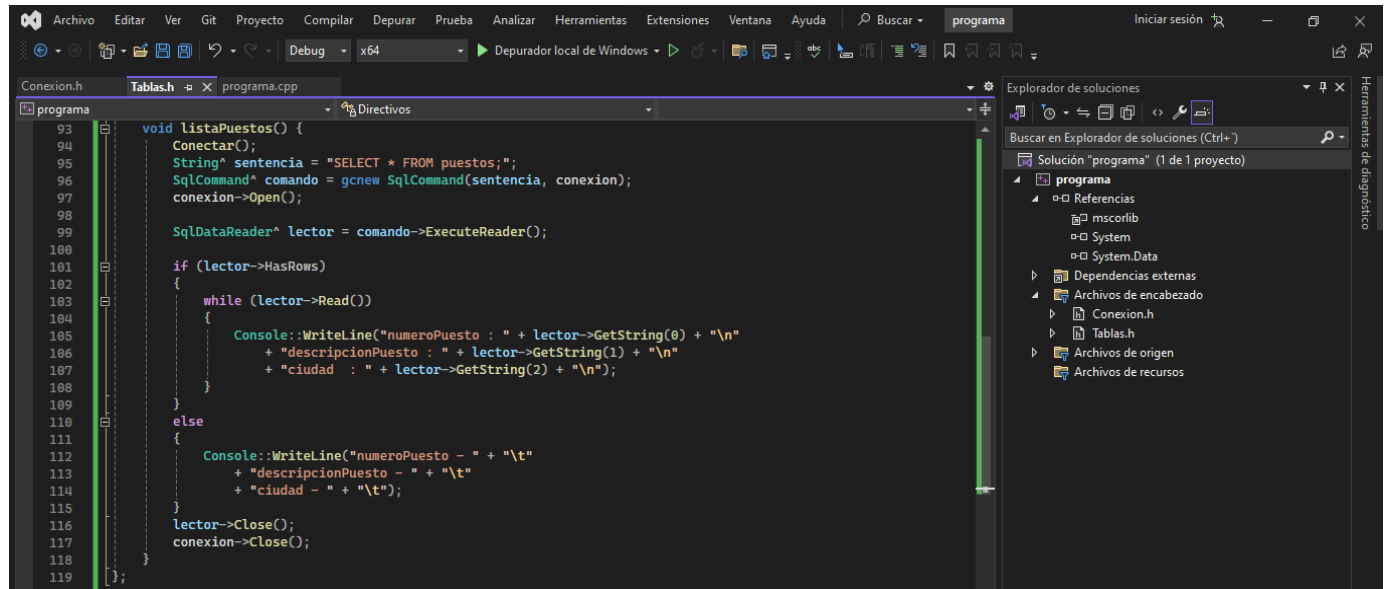


```
38 void listarCentros() {
39     Conectar();
40     String^ sentencia = "SELECT * FROM centros;";
41     SqlCommand^ comando = gcnew SqlCommand(sentencia, conexion);
42     conexion->Open();
43
44     SqlDataReader^ lector = comando->ExecuteReader();
45
46     if (lector->HasRows)
47     {
48         while (lector->Read())
49         {
50             Console.WriteLine("numeroCentro: " + lector->GetString(0) + "\n"
51                               + "nombreCentro: " + lector->GetString(1) + "\n"
52                               + "ciudad : " + lector->GetString(2) + "\n");
53         }
54     }
55     else
56     {
57         Console.WriteLine("numeroCentro --- " + "\t" + "\t" + "\t" + "\t"
58                           + "nombreCentro --- " + "\t" + "\t" + "\t" + "\t"
59                           + "ciudad --- " + "\n");
60     }
61     lector->Close();
62     conexion->Close();
63 }
64
65
```

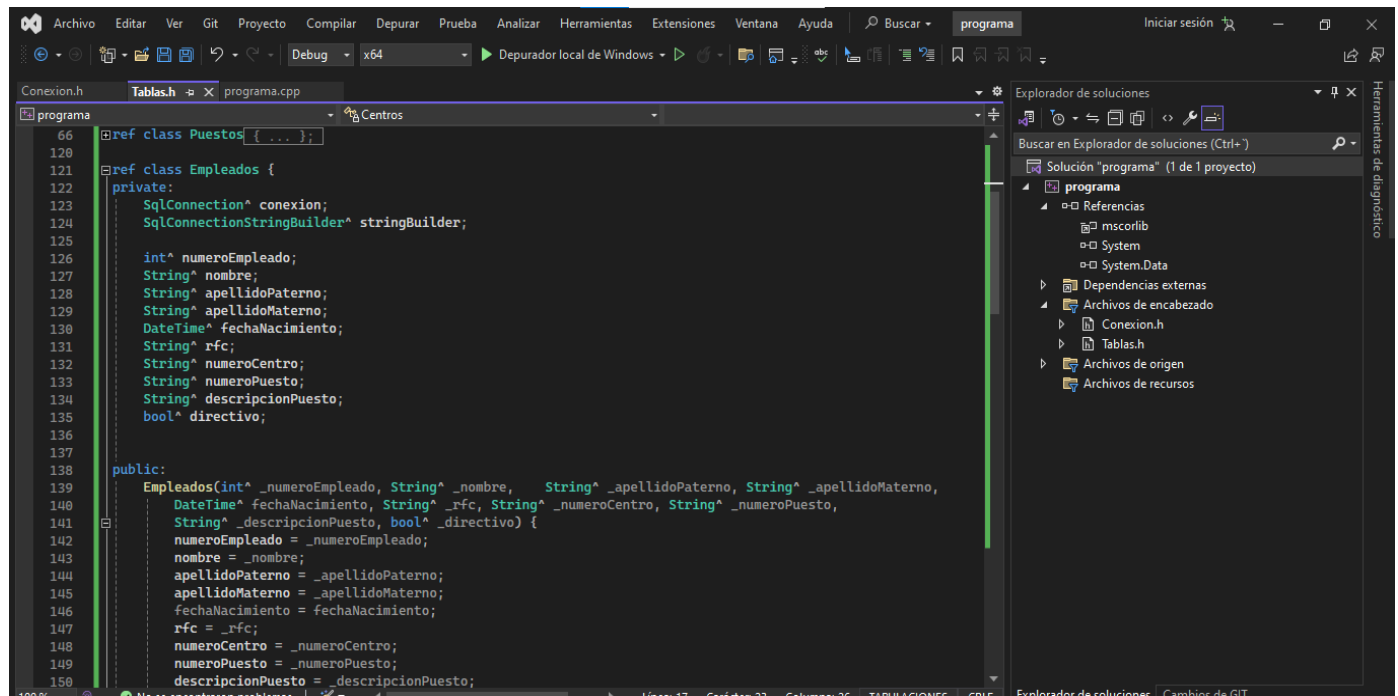
En cada clase se crean los atributos y el constructor

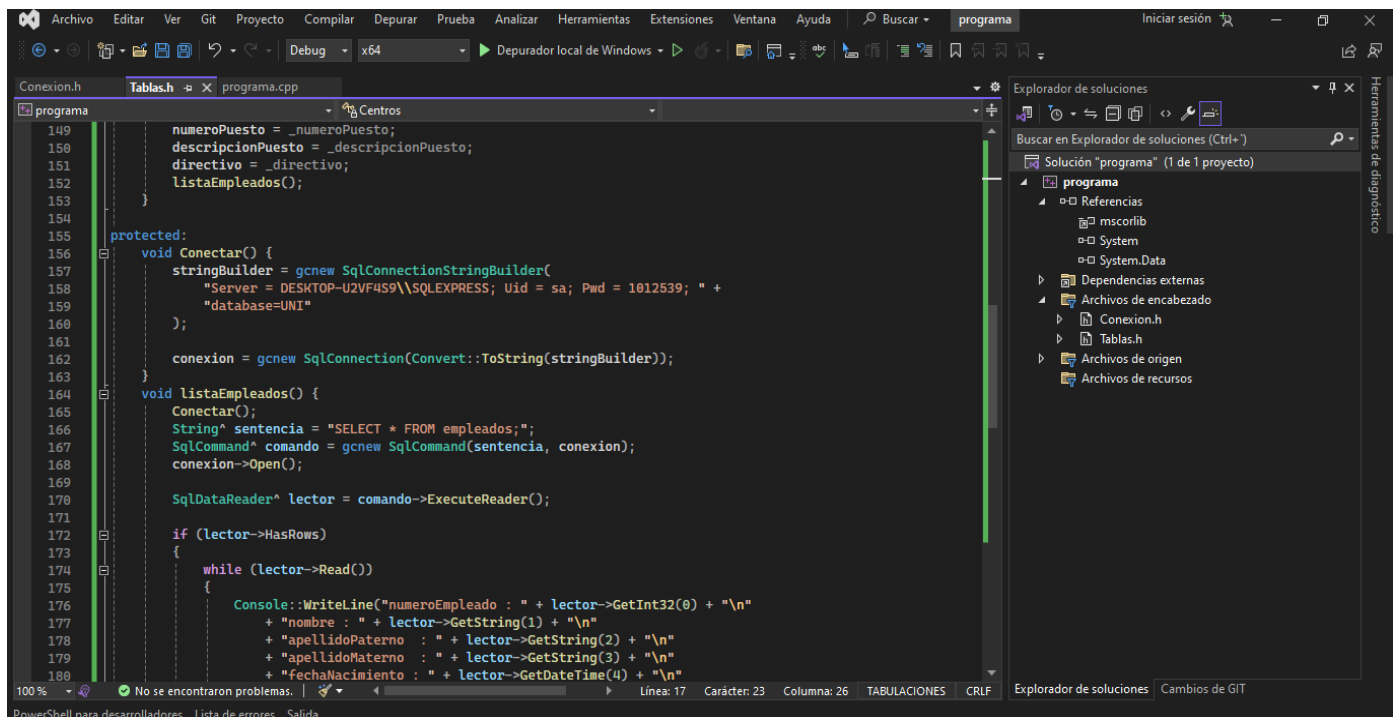


```
66 ref class Puestos {
67 private:
68     SqlConnection^ conexion;
69     SqlConnectionStringBuilder^ stringBuilder;
70
71     String^ numeroPuesto;
72     String^ descripcionPuesto;
73     String^ numeroCentro;
74
75 public:
76     Puestos(String^ _numeroPuesto, String^ _descripcionPuesto, String^ _numeroCentro) {
77         numeroPuesto = _numeroPuesto;
78         descripcionPuesto = _descripcionPuesto;
79         numeroCentro = _numeroCentro;
80         listaPuestos();
81     }
82
83
84 protected:
85     void Conectar() {
86         stringBuilder = gcnew SqlConnectionStringBuilder(
87             "Server = DESKTOP-U2VF4S9\\SQLEXPRESS; Uid = sa; Pwd = 1012539; " +
88             "database=UNI"
89         );
90
91         conexion = gcnew SqlConnection(Convert::ToString(stringBuilder));
92     }
93
94     void listaPuestos() {
95
96     }
97
98 }
99
100
```



En cada clase se crean los atributos y el constructor



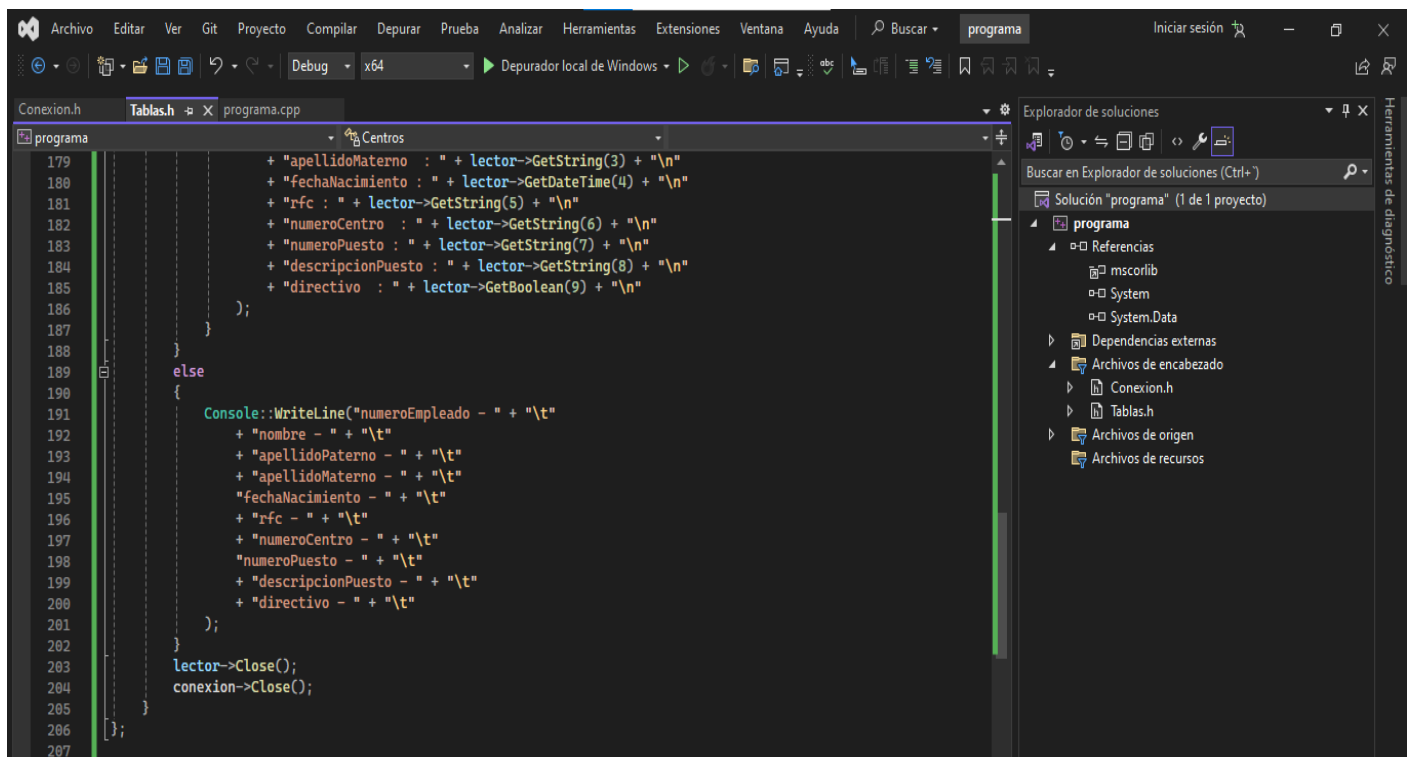


```
149     numeroPuesto = _numeroPuesto;
150     descripcionPuesto = _descripcionPuesto;
151     directivo = _directivo;
152     listaEmpleados();
153 }
154
155 protected:
156 void Conectar() {
157     stringBuilder = gcnew SqlConnectionStringBuilder(
158         "Server = DESKTOP-U2VF4S9\\SQLEXPRESS; Uid = sa; Pwd = 1012539; " +
159         "database=UNI"
160     );
161
162     conexion = gcnew SqlConnection(Convert::ToString(stringBuilder));
163 }
164 void listaEmpleados() {
165     Conectar();
166     String^ sentencia = "SELECT * FROM empleados;";
167     SqlCommand^ comando = gcnew SqlCommand(sentencia, conexion);
168     conexion->Open();
169
170     SqlDataReader^ lector = comando->ExecuteReader();
171
172     if (lector->HasRows)
173     {
174         while (lector->Read())
175         {
176             Console::WriteLine("numeroEmpleado : " + lector->GetInt32(0) + "\n"
177                 + "nombre : " + lector->GetString(1) + "\n"
178                 + "apellidoPaterno : " + lector->GetString(2) + "\n"
179                 + "apellidoMaterno : " + lector->GetString(3) + "\n"
180                 + "fechaNacimiento : " + lector->GetDateTime(4) + "\n"
181                 + "rfc : " + lector->GetString(5) + "\n"
182                 + "numeroCentro : " + lector->GetString(6) + "\n"
183                 + "numeroPuesto : " + lector->GetString(7) + "\n"
184                 + "descripcionPuesto : " + lector->GetString(8) + "\n"
185                 + "directivo : " + lector->GetBoolean(9) + "\n"
186             );
187         }
188     }
189     else
190     {
191         Console::WriteLine("numeroEmpleado - " + "\t"
192             + "nombre - " + "\t"
193             + "apellidoPaterno - " + "\t"
194             + "apellidoMaterno - " + "\t"
195             + "fechaNacimiento - " + "\t"
196             + "rfc - " + "\t"
197             + "numeroCentro - " + "\t"
198             + "numeroPuesto - " + "\t"
199             + "descripcionPuesto - " + "\t"
200             + "directivo - " + "\t"
201         );
202     }
203     lector->Close();
204     conexion->Close();
205 }
206
207 }
```

100% No se encontraron problemas. | Línea: 17 Carácter: 23 Columna: 26 TABULACIONES CRLF

Explorador de soluciones

- Solución "programa" (1 de 1 proyecto)
- programa
 - Referencias
 - mscorlib
 - System
 - System.Data
 - Dependencias externas
 - Archivos de encabezado
 - Conexion.h
 - Tablas.h
 - Archivos de origen
 - Archivos de recursos



```
179         + "apellidoMaterno : " + lector->GetString(3) + "\n"
180         + "fechaNacimiento : " + lector->GetDateTime(4) + "\n"
181         + "rfc : " + lector->GetString(5) + "\n"
182         + "numeroCentro : " + lector->GetString(6) + "\n"
183         + "numeroPuesto : " + lector->GetString(7) + "\n"
184         + "descripcionPuesto : " + lector->GetString(8) + "\n"
185         + "directivo : " + lector->GetBoolean(9) + "\n"
186     );
187 }
188
189 else
190 {
191     Console::WriteLine("numeroEmpleado - " + "\t"
192         + "nombre - " + "\t"
193         + "apellidoPaterno - " + "\t"
194         + "apellidoMaterno - " + "\t"
195         + "fechaNacimiento - " + "\t"
196         + "rfc - " + "\t"
197         + "numeroCentro - " + "\t"
198         + "numeroPuesto - " + "\t"
199         + "descripcionPuesto - " + "\t"
200         + "directivo - " + "\t"
201     );
202 }
203 lector->Close();
204 conexion->Close();
205 }
206
207 }
```

Explorador de soluciones

- Solución "programa" (1 de 1 proyecto)
- programa
 - Referencias
 - mscorlib
 - System
 - System.Data
 - Dependencias externas
 - Archivos de encabezado
 - Conexion.h
 - Tablas.h
 - Archivos de origen
 - Archivos de recursos

En cada clase se crean los atributos y el constructor

```

287
288 ref class Directivos {
289 private:
290     SqlConnection^ conexion;
291     SqlConnectionStringBuilder^ stringBuilder;
292
293     int^ id;
294     bool^ combustible;
295     int^ numeroEmpleado;
296
297 public:
298     Directivos(int^ _id, bool^ _combustible, int^ _numeroEmpleado) {
299         id = _id;
300         combustible = _combustible;
301         numeroEmpleado = _numeroEmpleado;
302         listaDirectivos();
303     }
304
305 protected:
306     void Conectar() {
307         stringBuilder = gcnew SqlConnectionStringBuilder(
308             "Server = DESKTOP-U2VF4S9\\SQLEXPRESS; Uid = sa; Pwd = 1012539; " +
309             "database=UNI"
310         );
311
312         conexion = gcnew SqlConnection(Convert::ToString(stringBuilder));
313     }
314
315     void listaDirectivos() {
316         Conectar();
317         String^ sentencia = "SELECT * FROM directivos;";
318         SqlCommand^ comando = gcnew SqlCommand(sentencia, conexion);
319     }

```

Explorador de soluciones: Solución "programa" (1 de 1 proyecto)

- programa
 - Referencias
 - mscorlib
 - System
 - System.Data
 - Dependencias externas
 - Archivos de encabezado
 - Conexion.h
 - Tablas.h
 - Archivos de origen
 - Archivos de recursos

PowerShell para desarrolladores | Lista de errores | Salida

```

320
321     };
322
323     conexion = gcnew SqlConnection(Convert::ToString(stringBuilder));
324
325 void listaDirectivos() {
326     Conectar();
327     String^ sentencia = "SELECT * FROM directivos;";
328     SqlCommand^ comando = gcnew SqlCommand(sentencia, conexion);
329     conexion->Open();
330
331     SqlDataReader^ lector = comando->ExecuteReader();
332
333     if (lector->HasRows)
334     {
335         while (lector->Read())
336         {
337             Console::WriteLine("id : " + lector->GetInt32(0) + "\n"
338                 + "combustible : " + lector->GetBoolean(1) + "\n"
339                 + "numeroEmpleado : " + lector->GetInt32(2) + "\n");
340         }
341     }
342     else
343     {
344         Console::WriteLine("id - " + "\t"
345             + "combustible - " + "\t"
346             + "numeroEmpleado - " + "\t");
347     }
348     lector->Close();
349     conexion->Close();
350 }
351
352 };

```

Explorador de soluciones: Solución "programa" (1 de 1 proyecto)

- programa
 - Referencias
 - mscorlib
 - System
 - System.Data
 - Dependencias externas
 - Archivos de encabezado
 - Conexion.h
 - Tablas.h
 - Archivos de origen
 - Archivos de recursos

PowerShell para desarrolladores | Lista de errores | Salida

5- Conclusión

Como pudimos notar en el trabajo de investigación que realizamos, nos da una detallada información sobre las bases de datos, así como los gestores de base de datos, partes muy importantes como su arquitectura y sus áreas de aplicación, quizá esto no sea tan relevante como parece, pero es importante tenerlo en cuenta.

Nos da la pauta para que empecemos a crear y diseñar nuestras propias bases de datos, los pasos que requerimos para hacerlo y darle una solución correcta al problema que queremos trasladar a la bd, y para hacerlo requerimos de las formas normales, las cuales hacen que nuestra base de datos quede con una buena lógica, ya que, sin eso, sería un caos y no resolvería nada.

Otro punto importante fue el del algebra relacional, ya que nos enseña a poder hacer nuestras futuras consultas que realizaremos en una base de datos, para poder obtener los datos o valores que queremos, y poco a poco poder hacer consultas más específicas y mejores.

6- Referencias

Jiménez Piano, M. y V. Ortiz-Repiso Jiménez. Evaluación y calidad de sedes web. España: Trea, 2007.

Universidad Nacional Mayor San Marcos. Facultad de Educación. Unidad de Postgrado. “Tabla para evaluar informes de investigación”.