Proyecto integrador: “Innovación digital para el crecimiento económico de las PyMEs de México”

Pensamiento computacional orientado a objetos (Gpo 303)

Sofía Zugasti Delgado A00837478

Mto. Daniel Pérez Rojas

25/11/2022

Introducción

Las PyMEs son todas aquellas empresas con equipos de trabajo de 10 o menos personas hasta un rango de 100 empleados, y así mismo, que producen un ingreso anual de ventas de hasta 250 millones de pesos (BBVA México, 2022). En la actualidad estas empresas representan el 99.8% de todas las existentes en México y son claves para el desarrollo del país ya que aportan el 42% del PIB total y generan el 78% del empleo (INEGI, 2022). Sin embargo, de mayo de 2019 a julio de 2021, 1.6 millones de empresas cerraron definitivamente según el Estudio sobre Demografía de los Negocios (EDN) 2021. Esto se dio porque solo 3 de cada 10 PyMEs se encontraban en internet y cuando de pronto, todo el país tuvo que mantenerse en aislamiento debido a la contingencia del COVID-19, estas empresas se vieron altamente afectadas (Paredes, 2018). No obstante, esto dejó dos lección muy importante: la digitalización para agregar el ecommerce y el envío a domicilio son el futuro de la prosperidad para cualquier pequeña, mediana o microempresa.

Solución propuesta

A raíz de esta necesidad, surgió una propuesta para ayudar al personal de todas estas empresas con la gestión de los pagos en línea como también de los envíos a domicilio. Esta propuesta consiste en la implementación de una base de datos electrónica de los clientes de la empresa. El objetivo de esto, es ayudar al recopilamiento de los datos digitales que hará que el proceso de envío de productos sea más eficaz, dentro del sistema se podrán agregar nuevos clientes, eliminar clientes registrados y mostrar la consulta general de todos los clientes inscritos.

Ahora bien, el programa consta del ingreso de los datos del cliente en las clases correspondientes. En primera instancia está la clase “Cliente” en la cual se almacenan los atributos de nombre, dirección y cuenta bancaria, así como los métodos o procedimientos de agregar, eliminar o mostrar clientes. Después, como herencia de la clase cliente, se tiene una relación “has a” con las clases Dirección y CuentaBancaria, las cuales tienen también sus propios atributos. En la clase Dirección se encuentran los atributos de calle, código postal, colonia y número. Por el otro lado, en la clase de CuentaBancaria se incluyen los atributos del nombre del titular, el número de tarjeta, el CVV de la cuenta, la fecha de vencimiento.

```
class CuentaBancaria {
public:
  string titular;
  double clabe;
  int ccv;
  string fecha;
  CuentaBancaria(string, double, int, string);
  CuentaBancaria();
};

//Clase que modela el objeto Dirección, atributo de l
class Direccion {
public:
  string calle;
  int cp;
  string colonia;
  int numero;
  Direccion(string, int, string, int);
  Direccion();
};
```

Imagen 1. Clase CuentaBancaria y Clase Dirección con sus atributos y métodos públicos.

```
41 class Cliente {
42 private:
43   string nombre;
44   CuentaBancaria cuenta;
45   Direccion direccion;
46
47 public:
48   Cliente(string, CuentaBancaria, Direccion);
49   Cliente();
50   string getNombre();
51   CuentaBancaria getCuenta();
52   Direccion getDireccion();
53   void setNombre(string n);
54   void setCuenta(CuentaBancaria c);
55   void setDireccion(Direccion d);
56 };
```

Imagen 2. Clase principal llamada Cliente con sus atributos y métodos privados y públicos.

Constructores

Para las clases Direccion y CuentaBancaria, los constructores fueron sencillos pues solamente requerían de ingresar datos de tipo ya determinados para C++. El constructor de CuentaBancaria se encarga de crear el objeto del mismo tipo, que contiene los datos que se solicitan para realizar un pago en línea (titular de la tarjeta, ccv, fecha de expiración, número Clabe).

```
110
111  //Constructor default de
     CuentaBancaria
112 ▼CuentaBancaria::CuentaBancaria() {
113  }
114
115  //Constructor que recibe los datos
     de cuenta del usuario para
     construir la variable del mismo
     tipo
116 ▼CuentaBancaria::CuentaBancaria(stri
     ng ti, double claBe, int c_cv,
     string fe) {
117    titular = ti;
118    clabe = claBe;
119    ccv = c_cv;
120    fecha = fe;
121  }
```

Imagen 3. Constructor del objeto CuentaBancaria.

Para la clase Direccion es el mismo caso, recibe los datos necesarios para realizar una entrega (calle, colonia, código postal, número de casa) y así construir el objeto de tipo Direccion para construir posteriormente el objeto Cliente.

```
 98
 99  //Constructor default de Direccion
100 ▼Direccion::Direccion() {
101  }
102
103  //Constructor que recibe los datos
     de dirección del usuario para
     construir la variable del mismo
     tipo
104 ▼Direccion::Direccion(string ca, int
      c_p, string co, int num) {
105    calle = ca;
106    cp = c_p;
107    colonia = co;
108    numero = num;
109  }
```

Imagen 4. Constructor del objeto Dirección.

Los objetos de la clase Cliente reciben 3 datos o variables: primeramente el nombre asociado a esa cuenta, del usuario en cuestión; seguido, la clase solicita un valor de tipo CuentaBancaria, es entonces que manda a llamar a dicha clase para recibir las variables dadas en esa clase y construir el objeto para usarse en la clase Cliente; lo mismo sucede con la clase Direccion, que es solicitada al final como tercer dato necesario para la clase Cliente, es entonces que actúa el constructor de la clase solicitada y construye la variable de tipo de dicha clase, y añadirla al objeto Cliente.

```
58  //Constructor default de la clase
    Cliente
59  Cliente::Cliente() {
60  }
61
62  //Constructor que recibe los datos
    para registrar al usuario, siendo
    el nombre, la cuenta y su
    dirección, estos 2 últimos
    pertenecientes a clases propias
63  Cliente::Cliente(string n,
    CuentaBancaria c, Direccion d) {
64    nombre = n;
65    cuenta = c;
66    direccion = d;
67  }
```

Imagen 5. Constructor del objeto Cliente.

Métodos

Para construir el elemento Cliente, además de los constructores, se implementaron métodos Get y Set ya que, al haber tener variables específicas del objeto, se privatizan para reservarlas a esa clase, y para poder manipularlas y darles uso para la construcción del objeto, se utiliza primero el Get que regrese la variable y se obtenga acceso a su modificación, y con el método Set, le damos valor a través de asignarle el valor de otra variable y así, el programa interprete cualquier otra variable con las específicas de la clase en cuestión.

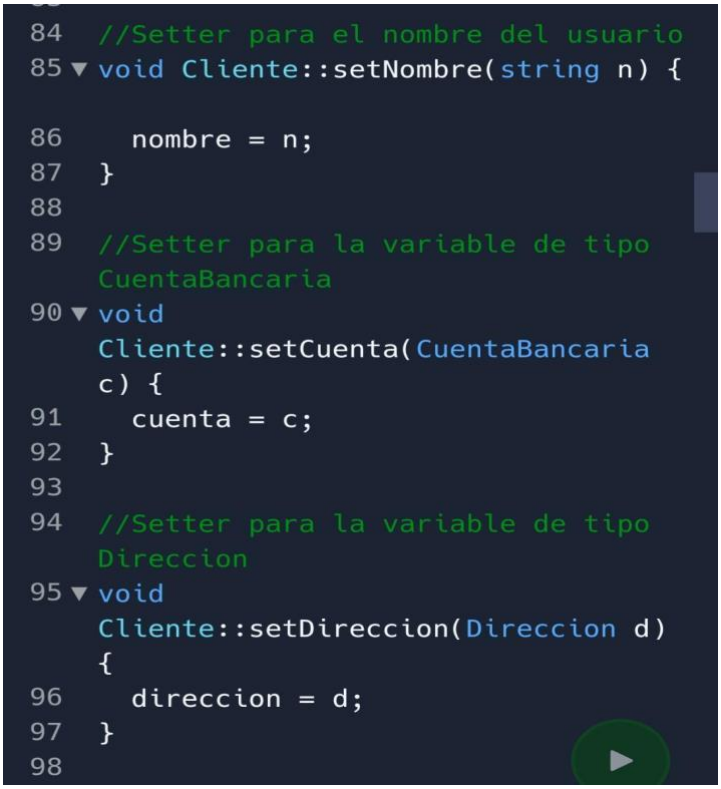
Para la clase Cliente, se emplea para las variables *nombre, cuenta* y direccion, que no es más que donde se guarde la información del usuario, y se obtiene y manipula con getNombre, getCuenta y getDireccion (que regresa el tipo de dato especifico sin recibir algo) y setNombre,

setCuenta y setDireccion (que no regresan algún valor pero reciben los datos del tipo para operar.

```
69  //Getter que regresa la variable
    privada "nombre"
70  string Cliente::getNombre() {
71    return nombre;
72  }
73
74  //Getter que regresa la variable
    privada "cuenta"
75  CuentaBancaria Cliente::getCuenta()
     {
76    return cuenta;
77  }
78
79  //Getter que regresa la variable
    privada "direccion"
80  Direccion Cliente::getDireccion() {

81    return direccion;
82  }
83
```

Imagen 6. Getters para obtención de variables a manipular.



```
84  //Setter para el nombre del usuario
85  void Cliente::setNombre(string n) {

86    nombre = n;
87  }
88
89  //Setter para la variable de tipo
    CuentaBancaria
90  void
    Cliente::setCuenta(CuentaBancaria
    c) {
91    cuenta = c;
92  }
93
94  //Setter para la variable de tipo
    Direccion
95  void
    Cliente::setDireccion(Direccion d)
    {
96    direccion = d;
97  }
98
```

Imagen 7. Setters para darle un valor usable a las variables privadas de la clase Cliente.

Agregar, eliminar y mostrar: Uso de Vector

El empleado que maneje el sistema tendrá las 3 opciones mencionadas con anterioridad; si este selecciona agregar, se le pedirán los datos del cliente los cuales son los atributos de la Clase Cliente que también solicita los atributos de las clases CuentaBancaria y Direccion, después se declarará un objeto de tipo CuentaBancaria, Dirección y Cliente, a los que se les asignan los datos pedidos al empleado y se agregarán a un vector llamado vectorClientes.

```
  Direccion direccionDefault(calleA, cpA, coloniaA, numeroCasaA);
  CuentaBancaria cuentaDefault(nombreTitularA, clabeA, ccvA, fechaA);
  Cliente clienteDefault(nombreA, cuentaDefault, direccionDefault);

//Una vez construído el objeto cliente, el programa lo agrega al final del vector
  vectorCliente.push_back(clienteDefault);
}
```

Imagen 8.. Declaración de los Objetos y acción de agregar al vectorCliente.

Si selecciona eliminar, se mostrará una lista numerada de todos los clientes con sus respectivos datos, donde se le pedirá al empleado ingresar el número del cliente que desea eliminar, después eliminaremos el cliente elegido con el índice de dicho elemento.

```
//Se elimina el cliente y se recorren los demás elementos para alinearse y rellenar el espacio vacío
  cout << "Ingrese el número del cliente a eliminar: "<<endl;
  cin >> clienteEliminado;
  vectorCliente.erase(vectorCliente.begin() + (clienteEliminado - 1));
  cout << "Cliente eliminado exitosamente!\n" << endl;
}
```

Imagen 9. Eliminación del cliente seleccionado.

Si selecciona la opción de mostrar elementos, se imprimirán en la consola todos los clientes con sus respectivos datos ordenados según su ingreso, por último si selecciona Salir, el programa finalizará su ejecución.

```
//Esta sección reutiliza el código en la opción 2 para sólo desplegar todos los clientes existentes
  else if (seleccion == 3) {
    for (int i = 0; i < vectorCliente.size(); ++i) {
      cout << "---Cliente " << i + 1 << "---" << endl;
      cout << "Nombre: " << vectorCliente[i].getNombre() << endl;

      cout << "--DATOS DE DIRECCIÓN--" << endl;
      cout << "Calle: " << vectorCliente[i].getDireccion().calle << endl;
      cout << "Código Postal: " << vectorCliente[i].getDireccion().cp << endl;
      cout << "Colonia: " << vectorCliente[i].getDireccion().colonia << endl;
      cout << "numeroCasa: " << vectorCliente[i].getDireccion().numero
           << endl;

      cout << "--DATOS DE CUENTA BANCARIA--" << endl;
      cout << "Titular: " << vectorCliente[i].getCuenta().titular << endl;
      cout << "Clabe: " << vectorCliente[i].getCuenta().clabe << endl;
      cout << "CCV: " << vectorCliente[i].getCuenta().ccv << endl;
      cout << "Fecha: " << vectorCliente[i].getCuenta().fecha << endl;
      cout << endl;
    }
```

Imagen 10. Impresión de los datos de todos los clientes.

Diagrama UML

Para el desarrollo del programa se creó primero un diagrama de clases UML en el cual se traza la estructura del programa concreto, por medio de la modelación de objetos, clases, atributos operaciones y relaciones.

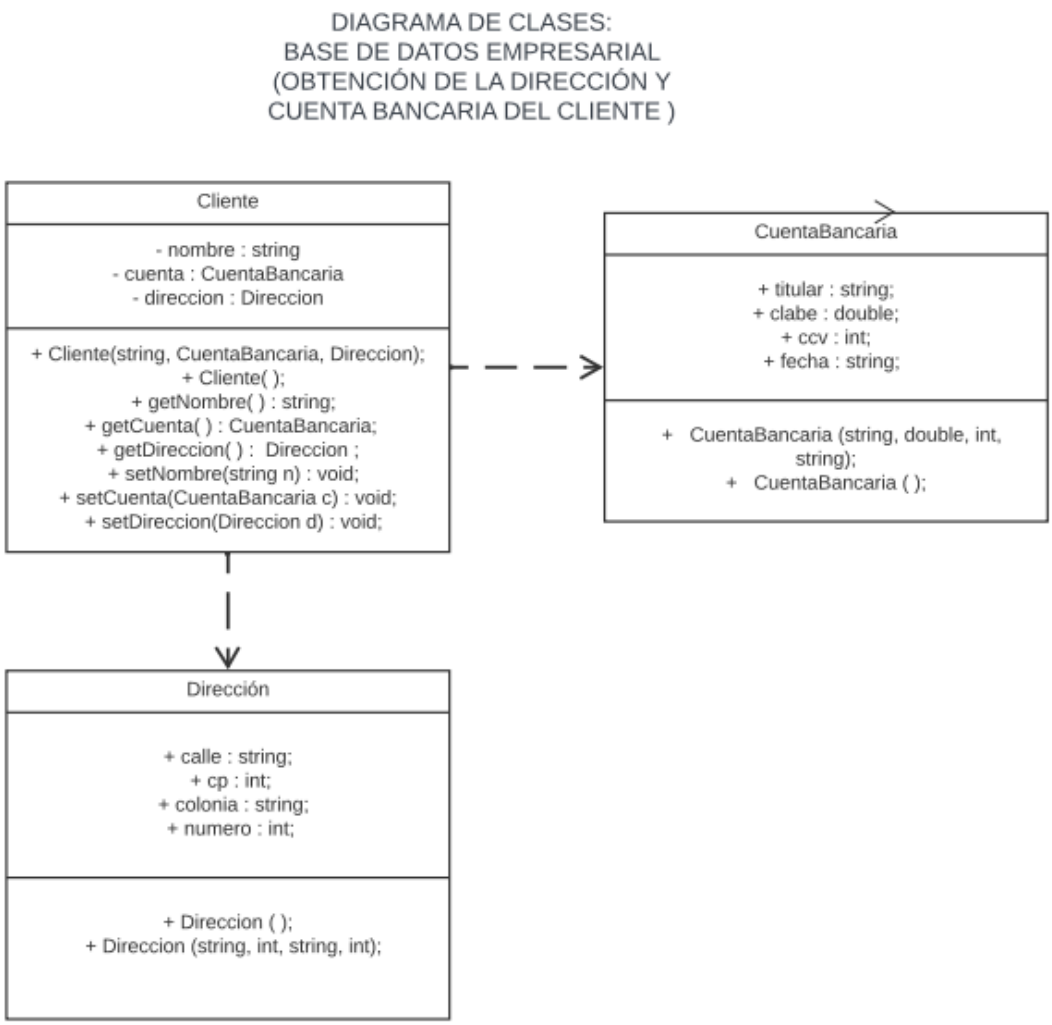


Imagen 11. Diagrama de clase UML del código propuesto.

En este diagrama UML se almacenan las tres clases que están incluidas en el programa. En primera instancia se encuentra el diagrama de la clase Cliente la cual representa la entidad del sistema en el cual se encuentran los atributos de tipo privados (debido a que me muestran solo en la clase Cliente) llamados “nombre”, definido como un objeto tipo string, y otros dos atributos que son objetos de otras dos clases, llamados “cuenta” que se establece con la clase CuentaBancaria y Direccion que se conforma por la clase Direccion.

Debajo de los atributos, se encuentra la operación pública Cliente, en donde el método get se encarga de mostrar el valor de los atributos privados “nombre” y de los valores de las otras clases “Cuenta” y Direccion”. Por último el método set modifica los valores de los atributos para que no se devuelvan dichos valores en cada uno de los atributos.

Más tarde, las clases Direccion y CuentaBancaria tienen constructores de relación “has-a” con la clase Cliente. En los dos diagramas se muestran los atributos públicos con sus respectivos tipos de objetos, así como las operación públicas Direccion y CuentaBancaria.

Dificultades presentadas en la implementación

En la planeación de nuestra propuesta, se tuvo que investigar cuales eran las necesidades para la organización de un buen sistema para pequeñas y medianas empresas por transacciones electrónicas, así que decidimos centrarnos en la información de los clientes, solicitando datos necesarios para la compra/venta en línea como Dirección y datos de la Cuenta Bancaria del cliente.

Así es como empezamos a tener algunas problemáticas al implementar el programa, ya que debíamos plasmar nuestra propuesta mediante el uso de la programación orientada a objetos donde aplicaremos todos los conceptos vistos en la materia, así que primeramente diseñamos un diagrama de clases UML para organizar las clases con sus respectivos atributos y métodos que cumplieran con lo requieran. A partir de eso, empezamos a programar en c + + guiándonos de ejercicios anteriores donde aplicamos el uso de clases, métodos y atributos, así como la aplicación de herencia con una clase principal y subclases. A su vez, utilizamos vectores de dicho objetos, donde pudimos manipularlos agregando, eliminando y mostrando los elementos de dicho vector de objetos.

Conclusiones

Al realizar este proyecto pudimos implementar todos los conceptos que aprendimos en la unidad de formación de modelación computacional orientada a objetos. Y al implementarlos, pudimos entender que al definir objetos en programación es mucho más fácil llevar a cabo programas como el que realizamos previamente. En resumen, logramos implementar en el código todas las consideraciones solicitadas para la ejecución del proyecto integrador, donde pusimos en práctica nuestros conocimientos de POO, aplicando conceptos como: clases, atributos, métodos objetos y vectores, al mismo tiempo, que fue ejecutada una solución para un problemática de la vida cotidiana, en este caso un sistema dirigido a pequeñas y medianas empresas en el que se puedan manipular los datos de los clientes ingresados.

Sofía Zugasti Delgado: Tras la implementación de la situación problema, pude utilizar los conceptos vistos de POO para generar una innovación benéfica dirigida al crecimiento de la

digitalización de las PyMEs de México. Creo que me ayudó mucho aplicar los conocimientos aprendidos en esta clase en una situación como esta pues es un escenario que está ocurriendo en la actualidad. Sin embargo, me costó mucho plantear el pensamiento lógico de la situación y representarlo en el lenguaje de programación C++ ya que es mucho más complejo que Python el cual era el único lenguaje con el que yo estaba familiarizada. No obstante, creo que pude fortalecer mis competencias y aprender mucho de este reto.

REFLEXIÓN DE COMPETENCIAS

SICT0302-Toma de decisiones: Considero haber desarrollado la competencia al poder decidir y centrar el proyecto en abarcar una problemática específica de manera objetiva y no intentar abarcar a groso modo varias problemáticas sin darles solución, es entonces que delimitamos el acercamiento de las PYMES al servicio de compra en línea y otorgar mejor atención y servicio de atención al cliente.

SICT0303-Implementación de acciones: Esta competencia se desarrolló al contemplar que la problemática involucra ayuda en el desarrollo de PYMES a entrar en el mercado con apoyo de tecnologías computacionales para beneficiarse y adaptarse, así consideramos las limitaciones dentro del ambiente de una PYME y así priorizar qué elementos utilizar y qué programar o qué programa emplear que pueda ser verdaderamente aprovechado por una PYME, tal como una base de datos para atención a los usuarios.

SICT0401-Aplicación de estándares y normas: Al cumplir con los estándares de programación, fuí capaz de satisfacer una necesidad para un proyecto de emprendimiento como lo puede ser el ganarse a los usuarios para mantener el crecimiento, y qué mejor que disponer un servicio a su disposición como la facilidad de compra en línea para los usuarios al almacenar sus datos.

SEG0702-Tecnologías de vanguardia: Se seleccionó el uso de vectores y clases para facilitar el funcionamiento y guardado de una base de datos, desarrollando la competencia al programar correctamente una base de datos funcionando a partir de esas funciones en C++ y darle sencillez y eficiencia al programa.

Referencias

Zamarrón, I. (2022, September 27). *Coparmex empuja iniciativa de Ley del Emprendimiento para potenciar a pymes*. Forbes México. https://www.forbes.com.mx/coparmex-empuja-iniciativa-de-ley-del-emprendimiento-para-potenciar-a-pymes/

INEGI. (2022, June 2). *DEMOGRAFÍA DE LOS ESTABLECIMIENTOS MIPYME EN EL CONTEXTO DE LA PANDEMIA POR COVID-19*. INEGI Informa. https://www.inegi.org.mx/contenidos/saladeprensa/aproposito/2022/EAP_Demog_MIPYME22.pdf

BBVA México. (2022, March 30). *¿Qué son las Pymes y qué tipos hay?* BBVA México. https://www.bbva.mx/educacion-financiera/creditos/que-es-una-pyme.html

Paredes, S. (2018, December 28). *#SelecciónForbes2018 | Solo 3 de cada 10 Pymes mexicanas están en internet: Google*. Forbes México. https://www.forbes.com.mx/solo-3-de-cada-10-pymes-mexicanas-estan-en-internet-google/

Anexos:

Diagrama de clase en Lucidchart: https://lucid.app/lucidchart/c1b302bd-1d4d-4dbb-b825-aac8b2ee7ab9/edit?invitationId=inv_6f2ba137-6065-43c9-a2e4-332dd4b9cb0f&page=HWEp-vi-RSFO#

Código implementado en replit: https://replit.com/join/niyynvcrod-sofiazugasti