

Universidad Tecnológica de Durango

Tecnologías de la Información

programación Orientada a Objetos

Actividades

“Proyecto final”

Alumnos:

- Edmundo Cardoza Reveles
- Ángel de Jesús Avitia

2°B

Docente:

- Ing. Dagoberto Fiscal Gurrola, M.T.I.

Contenido

Tabla de ilustraciones	3
Objetivo General.....	3
Objetivo unidad 3	4
Actividad del reporte	5
diagrama UML.....	5
Diagrama base de datos.	6
Codificación.....	7
Ejecución	13
Conclusiones.....	17
Bibliografía	18

Tabla de ilustraciones

Ilustración 1 Diagrama UML.....	6
Ilustración 2 Diagrama relacional.....	7
Ilustración 3 Módulos	7
Ilustración 4 Primer modulo conexión	9
Ilustración 5 Clase empleados	10
Ilustración 6 Clase interfaz	11
Ilustración 7 Clase Clientes	12
Ilustración 8 Inicio de sesión	13
Ilustración 9 Menu Empleados	14
Ilustración 10 Ejecucion	15
Ilustración 11 Base de datos	15
Ilustración 12 Menú Clientes.....	16

Objetivo General

Generar aplicaciones de software mediante el Paradigma Orientado a Objetos aplicando buenas prácticas en un lenguaje de programación para la solución de problemas específicos.

Programación Orientada a objetos.

Edmundo Cardoza Reveles
Ángel de Jesús Avitia

Objetivo unidad 3

Codificar clases empleando el paradigma de la Programación Orientada a Objetos para el desarrollo de aplicaciones.

Programación Orientada a objetos.

Edmundo Cardoza Reveles
Ángel de Jesús Avitia

Actividad del reporte

diagrama UML

En la siguiente imagen se puede ver las clases utilizadas en el programa de gestión de banco se pueden observar varias asociaciones entre las clases, así como dependencias.

Programación Orientada a objetos.

Edmundo Cardoza Reveles
Ángel de Jesús Avitia

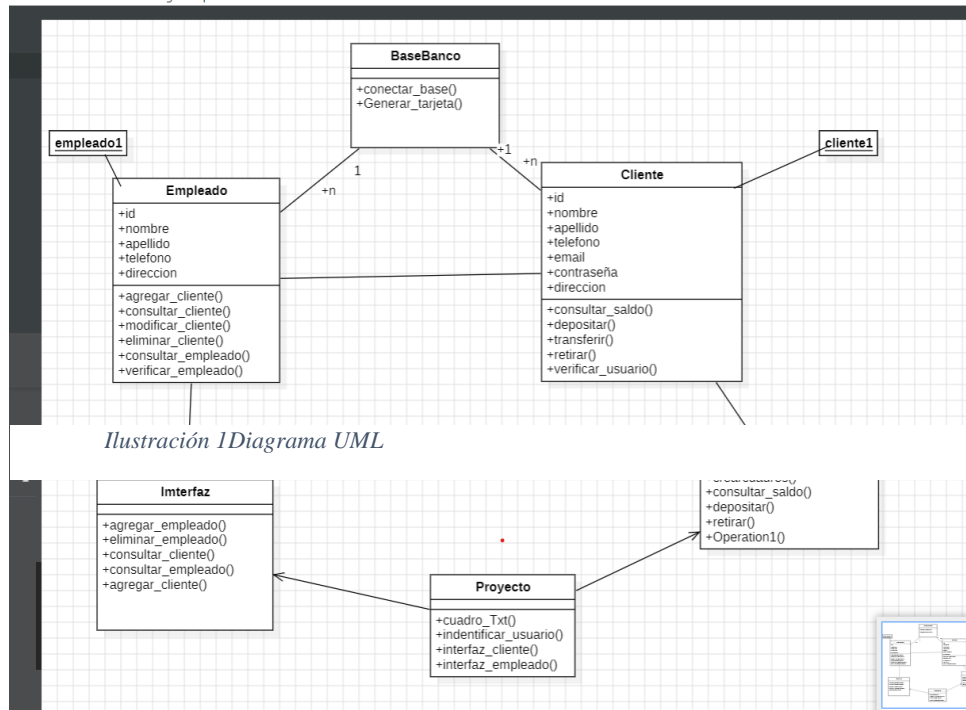


Diagrama base de datos.

Como se puede observar en la imagen esta es la base de datos que implementa el programa gestor de banco, en las relaciones podemos identificar varias llaves foráneas las cuales nos permiten identificar tanto al propietario de la cuenta, así como el empleado que lo ha registrado. Se tiene una tabla aparte en la cual se registran los movimientos para poder dar un seguimiento con mayor precisión.

Programación Orientada a objetos.

Edmundo Cardoza Reveles
Ángel de Jesús Avitia

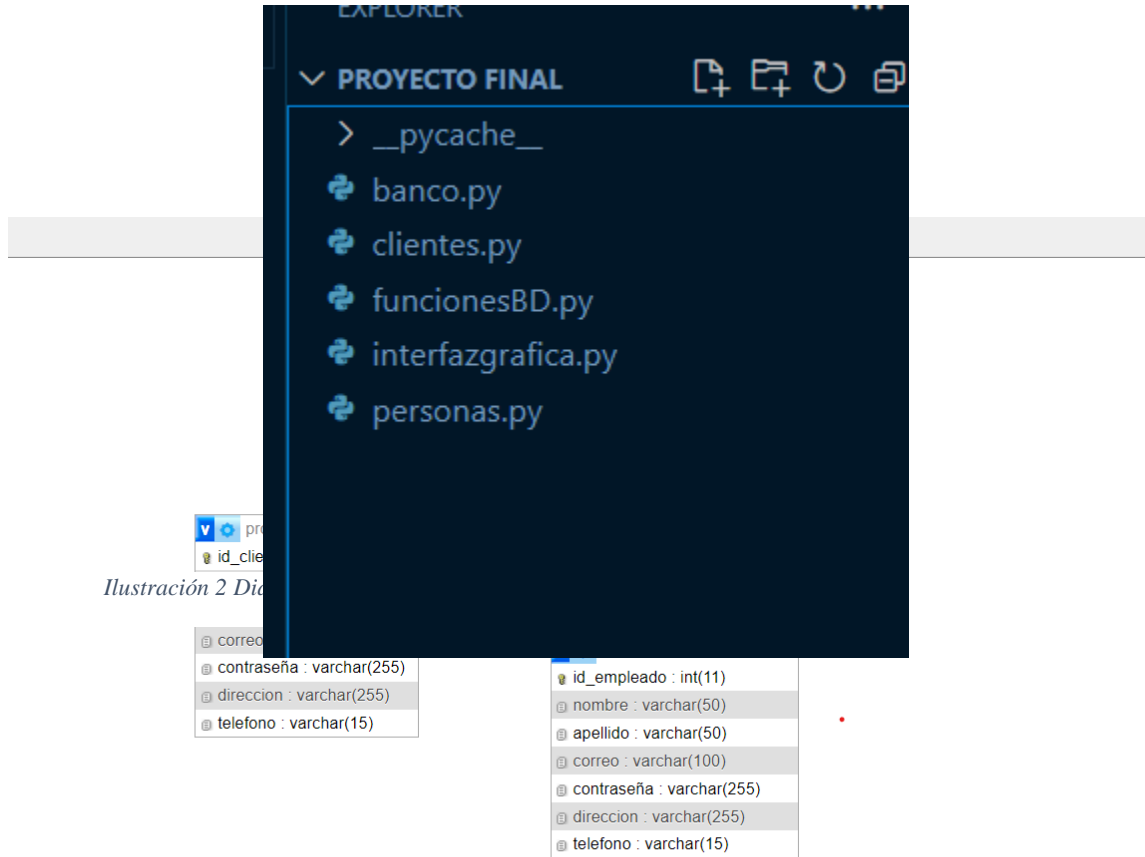


Ilustración 2 Diagrama de Base de Datos

Codificación.

En la siguiente imagen podemos ver la distribución modular que llevo acabo nuestro proyecto

Ilustración 3 Módulos

Programación Orientada a objetos.

Edmundo Cardoza Reveles
Ángel de Jesús Avitia

Programación Orientada a objetos.

Edmundo Cardoza Reveles
Ángel de Jesús Avitia

En la siguiente imagen se puede mostrar el primer modulo el cual integra la conexión y el método para generar los números aleatorios de una tarjeta.

```

1  import mysql.connector
2  import random
3  from mysql.connector import Error
4  class bd:
5
6      @staticmethod
7      def conectar():
8          """Conecta a la base de datos MySQL y devuelve el objeto de conexión."""
9          try:
10             conexion = mysql.connector.connect(
11                 host='localhost',
12                 user='root',
13                 password='',
14                 database='proyecto_final'
15             )
16             if conexion.is_connected():
17                 print('conexión exitosa a la base de datos')
18                 return conexion
19             except Error as e:
20                 print(f'Error al conectar a la base de datos: {e}')
21                 return None
22
23      @staticmethod
24      def generar_numero_tarjeta():
25          """Genera un número de tarjeta de 12 dígitos aleatorios."""
26          numero_tarjeta = ''.join([str(random.randint(0, 9)) for _ in range(12)])
27          return numero_tarjeta
28

```

Ilustración 4 Primer modulo conexión

En la siguiente imagen podemos ver la construcción de la clase empleados con sus respectivos atributos y algunos de sus métodos de los cuales podemos apreciar son estáticos para poder verificar su funcionamiento

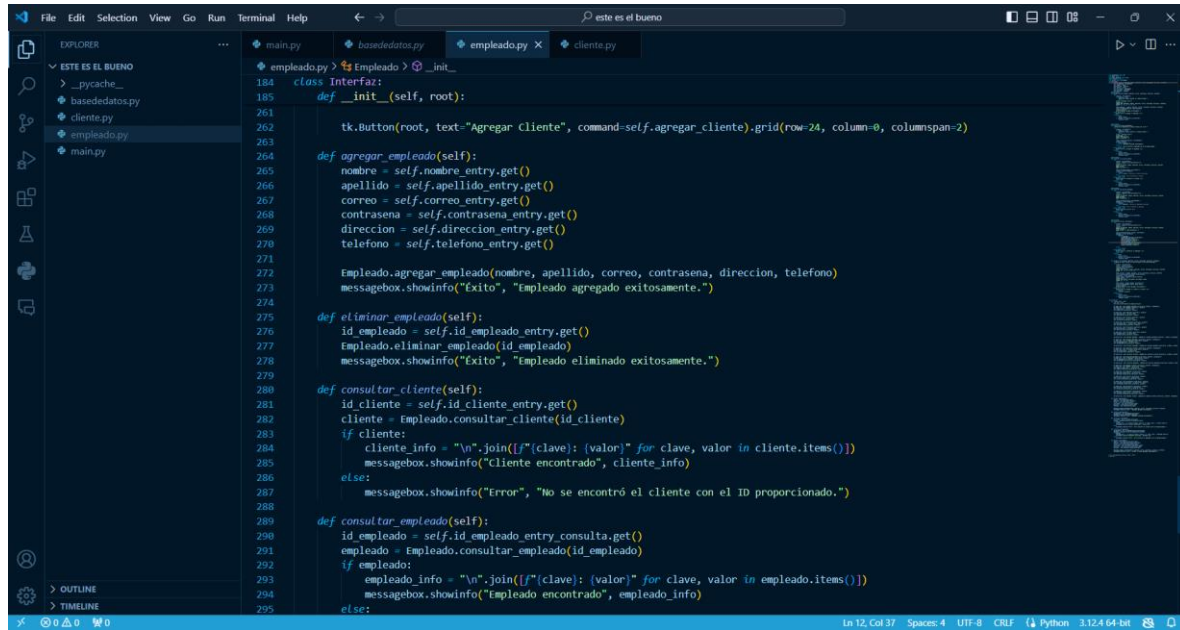
```

1  from basedatos import bd
2  from mysql import *
3  from mysql.connector import Error
4  import tkinter as tk
5  from tkinter import messagebox
6  class Empleado:
7      def __init__(self, id_empleado, nombre, apellido, correo, contraseña, direccion, telefono):
8          self.id_empleado = id_empleado
9          self.nombre = nombre
10         self.apellido = apellido
11         self.correo = correo
12         self.contrasena = contraseña
13         self.direccion = direccion
14         self.telefono = telefono
15     @staticmethod
16     def agregar_empleado(nombre, apellido, correo, contraseña, direccion, telefono):
17         try:
18             conexion = bd.conectar()
19             if conexion is None:
20                 print("No se pudo conectar a la base de datos.")
21                 return
22             cursor = conexion.cursor()
23             agregacion = """
24             INSERT INTO empleados (nombre, apellido, correo, contraseña, direccion, telefono)
25             VALUES (%s, %s, %s, %s, %s, %s)
26             """
27             datos_empleado = (nombre, apellido, correo, contraseña, direccion, telefono)
28             cursor.execute(agregacion, datos_empleado)
29             conexion.commit()
30             print("Empleado agregado exitosamente.")
31         except Error as e:
32             print(f"Error al agregar el empleado: {e}")
33         finally:
34             if cursor:
35                 cursor.close()
36             if conexion and conexion.is_connected():
37                 conexion.close()

```

Ilustración 5 Clase empleados

La siguiente imagen nos muestra la implementación de los métodos de la clase empleados con la interfaz creada con tkinter a través de una clase.

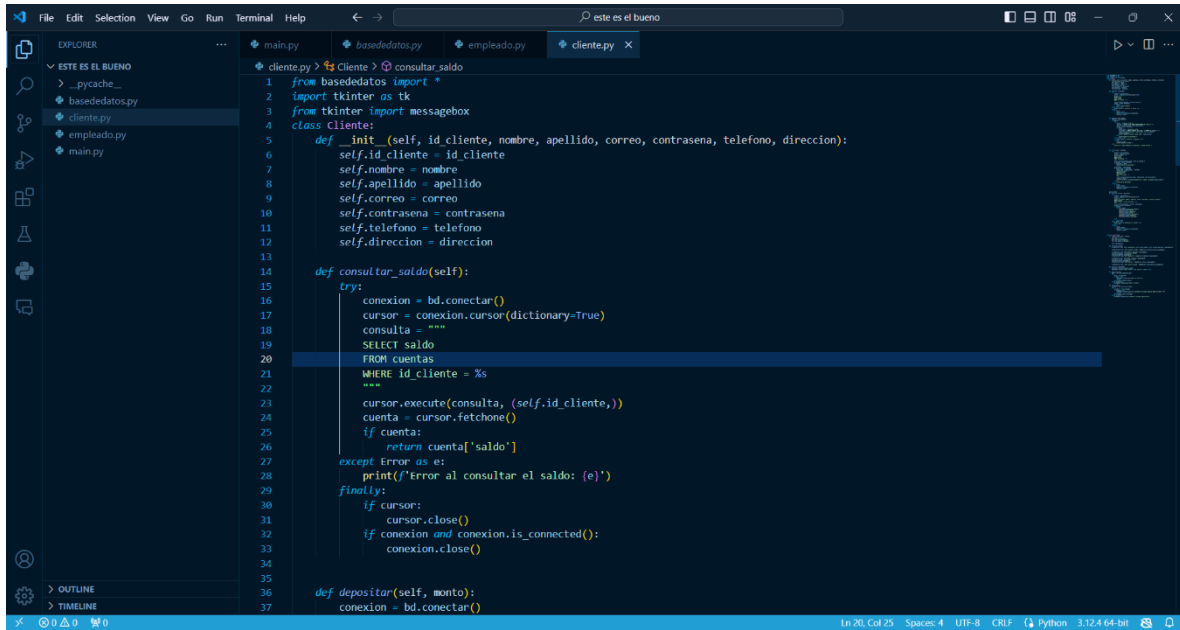


```

184 class Interfaz:
185     def __init__(self, root):
186
187         tk.Button(root, text="Agregar Cliente", command=self.agregar_cliente).grid(row=24, column=0, columnspan=2)
188
189     def agregar_empleado(self):
190         nombre = self.nombre_entry.get()
191         apellido = self.apellido_entry.get()
192         correo = self.correo_entry.get()
193         contrasena = self.contrasena_entry.get()
194         direccion = self.direccion_entry.get()
195         telefono = self.telefono_entry.get()
196
197         Empleado.agregar_empleado(nombre, apellido, correo, contrasena, direccion, telefono)
198         messagebox.showinfo("Exito", "Empleado agregado exitosamente.")
199
200     def eliminar_empleado(self):
201         id_empleado = self.id_empleado_entry.get()
202         Empleado.eliminar_empleado(id_empleado)
203         messagebox.showinfo("Exito", "Empleado eliminado exitosamente.")
204
205     def consultar_cliente(self):
206         id_cliente = self.id_cliente_entry.get()
207         cliente = Empleado.consultar_cliente(id_cliente)
208         if cliente:
209             cliente_info = "\n".join([f"{clave}: {valor}" for clave, valor in cliente.items()])
210             messagebox.showinfo("Cliente encontrado", cliente_info)
211         else:
212             messagebox.showinfo("Error", "No se encontró el cliente con el ID proporcionado.")
213
214     def consultar_empleado(self):
215         id_empleado = self.id_empleado_entry_consulta.get()
216         empleado = Empleado.consultar_empleado(id_empleado)
217         if empleado:
218             empleado_info = "\n".join([f"{clave}: {valor}" for clave, valor in empleado.items()])
219             messagebox.showinfo("Empleado encontrado", empleado_info)
220         else:
  
```

Ilustración 6 Clase interfaz

La siguiente imagen nos muestra la construcción de la clase Cliente en la cual podemos ver sus atributos y algunos de sus métodos.



```

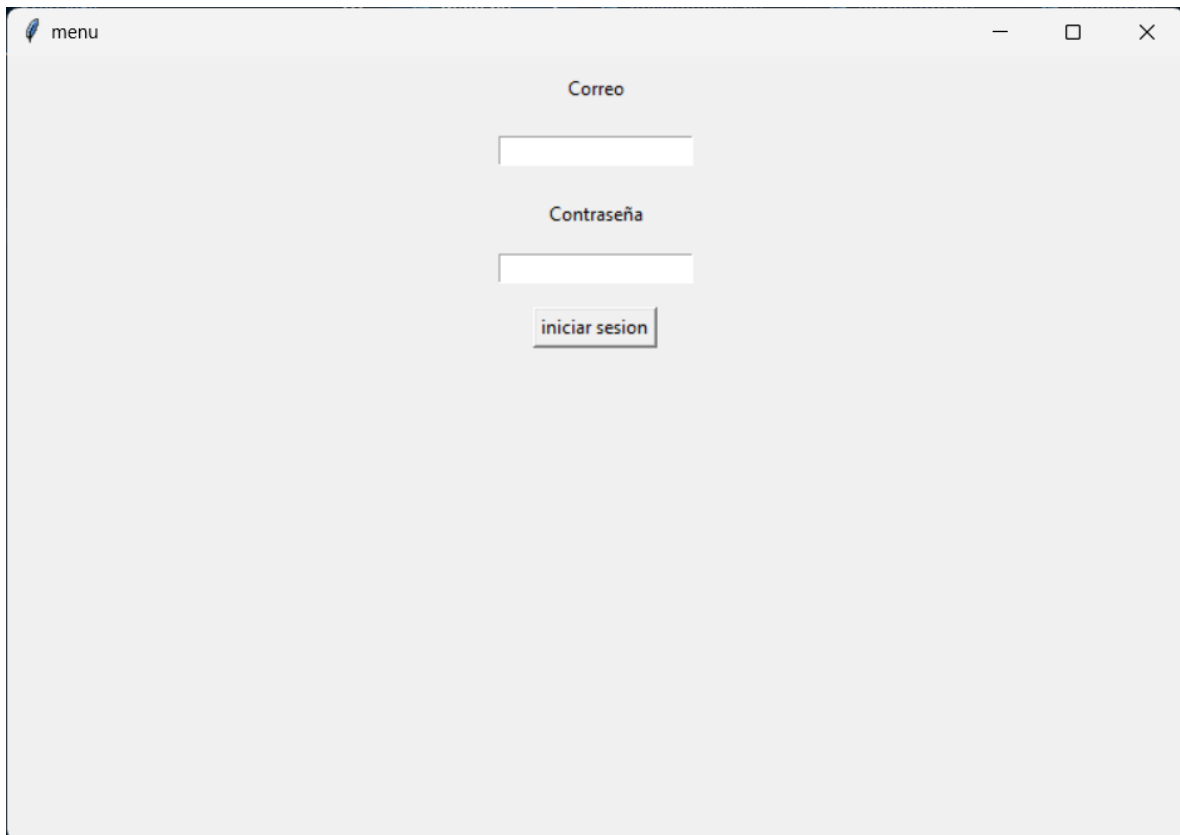
1 from basedatos import *
2 import tkinter as tk
3 from tkinter import messagebox
4 class Cliente:
5     def __init__(self, id_cliente, nombre, apellido, correo, contrasena, telefono, direccion):
6         self.id_cliente = id_cliente
7         self.nombre = nombre
8         self.apellido = apellido
9         self.correo = correo
10        self.contrasena = contrasena
11        self.telefono = telefono
12        self.direccion = direccion
13
14    def consultar_saldo(self):
15        try:
16            conexion = bd.conectar()
17            cursor = conexion.cursor(dictionary=True)
18            consulta = """
19            SELECT saldo
20            FROM cuentas
21            WHERE id_cliente = %s
22            """
23            cursor.execute(consulta, (self.id_cliente,))
24            cuenta = cursor.fetchone()
25            if cuenta:
26                return cuenta['saldo']
27        except Error as e:
28            print("Error al consultar el saldo: (e)")
29        finally:
30            if cursor:
31                cursor.close()
32            if conexion and conexion.is_connected():
33                conexion.close()
34
35    def depositar(self, monto):
36        conexion = bd.conectar()
37

```

Ilustración 7 Clase Clientes

Ejecución

En la siguiente imagen nos muestra el inicio de sesión el cual verifica en las tablas de empleados y clientes para poder crear una instancia de la clase y acceder a sus métodos.



The image shows a graphical user interface for a login system. It features a window with a title bar containing a 'menu' icon and standard window controls (minimize, maximize, close). The main area of the window is light gray and contains the following elements:

- A label 'Correo' above a text input field.
- A label 'Contraseña' above a text input field.
- A button labeled 'iniciar sesion' located below the password field.

Ilustración 8 Inicio de sesión

La siguiente imagen nos muestra el menú de una instancia de la clase empleados.

The screenshot shows a menu window titled 'Int...' with the following structure:

- Agregar Empleado**
 - Nombre
 - Apellido
 - Correo
 - Contraseña
 - Dirección
 - Teléfono
 - Agregar Empleado
- Eliminar Empleado**
 - ID Empleado
 - Eliminar Empleado
- Consultar Cliente**
 - ID Cliente
 - Consultar Cliente
- Consultar Empleado**
 - ID Empleado
 - Consultar Empleado
- Agregar Cliente**
 - Nombre
 - Apellido
 - Correo
 - Contraseña
 - Dirección
 - Teléfono
 - Agregar Cliente

Ilustración 9 Menu Empleados

La siguiente imagen nos muestra el resultado de la ejecución de uno de los métodos de empleado.

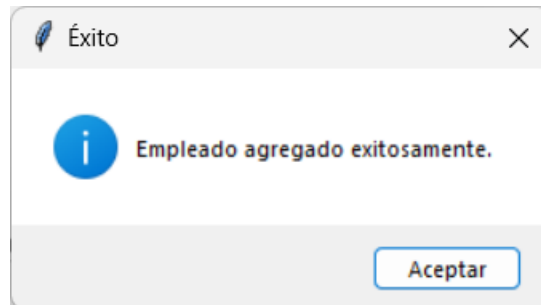


Ilustración 10 Ejecucion

La siguiente imagen nos muestra la correcta ejecución al insertar los datos en una base de datos

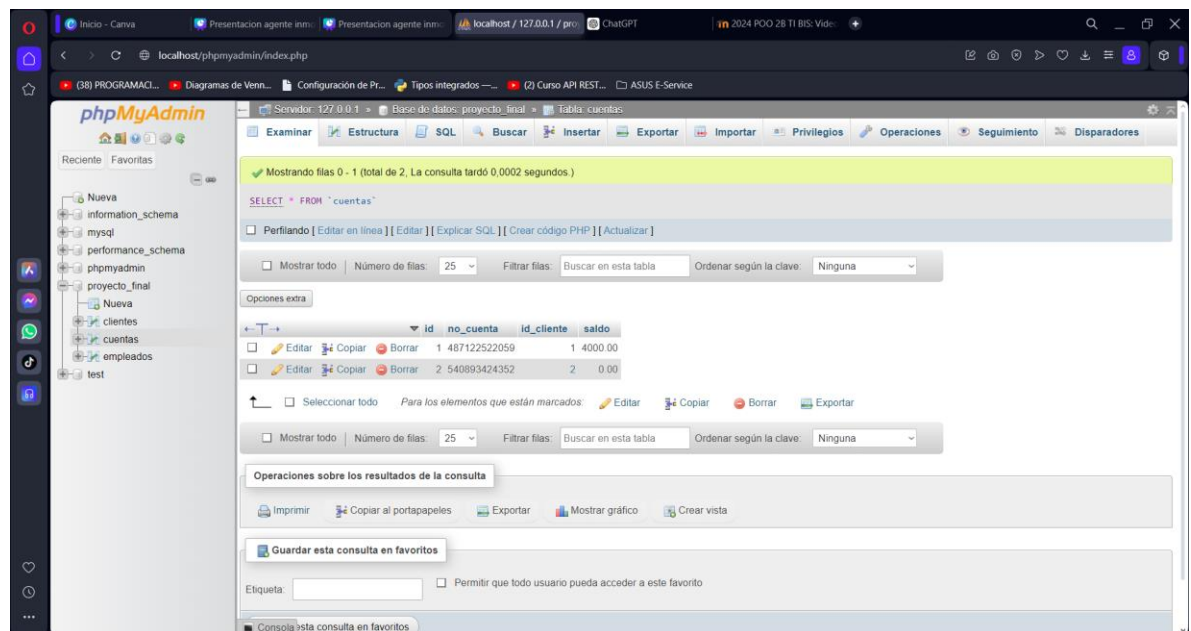


Ilustración 11 Base de datos

Programación Orientada a objetos.

Edmundo Cardoza Reveles
Ángel de Jesús Avitia

La siguiente imagen nos muestra el menú de una instancia de la clase Clientes

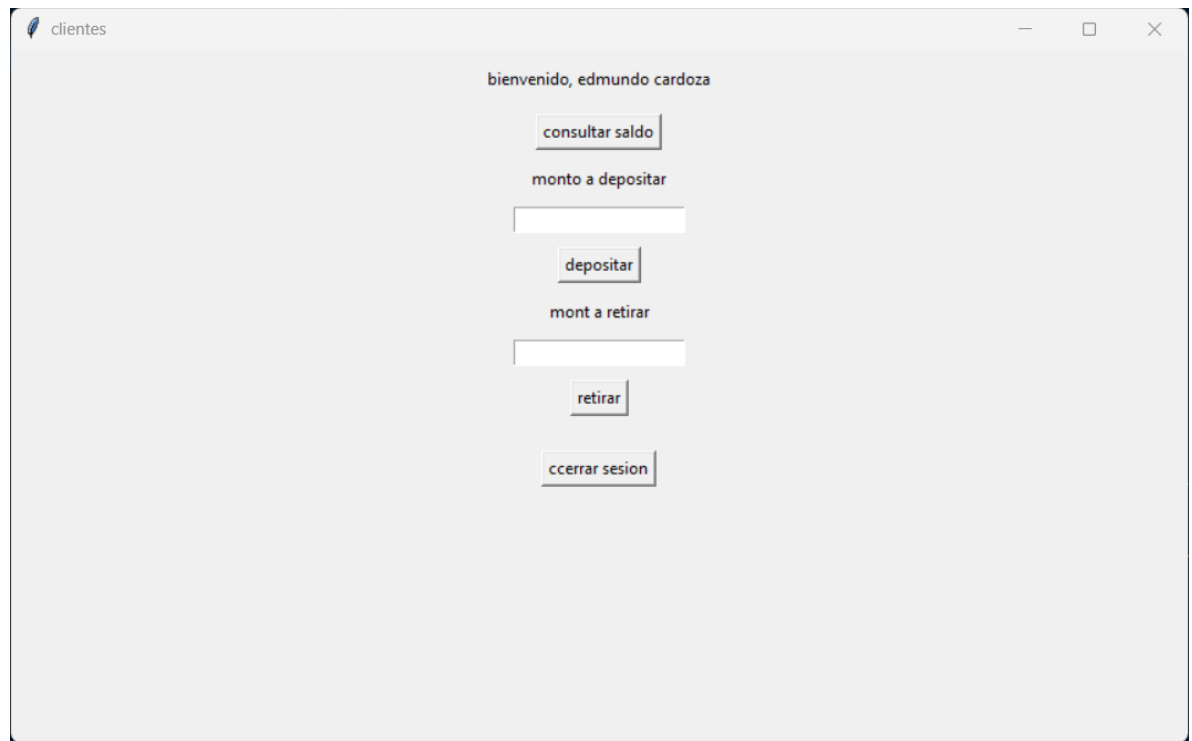


Ilustración 12 Menú Clientes

Conclusiones

Este proyecto nos ha ayudado a afianzar los conocimientos obtenidos durante el cuatrimestre, permitiéndonos integrar y aplicar de manera práctica los conceptos aprendidos en las materias más relevantes para las tecnologías de la información. La implementación de este proyecto ha sido fundamental para consolidar nuestra comprensión y habilidad en áreas clave como la programación, la gestión de bases de datos y el desarrollo de interfaces gráficas.

Queremos expresar nuestro sincero agradecimiento al profesor por ser una parte esencial de este trayecto, especialmente en una etapa tan crucial como el inicio de nuestra carrera. La guía y el apoyo brindado durante el curso han sido invaluable, ya que los temas iniciales constituyen la base sobre la cual construiremos nuestro conocimiento futuro. Su dedicación y compromiso han sido fundamentales para nuestro aprendizaje y desarrollo en el campo de las tecnologías de la información.

Este proyecto no solo ha reforzado nuestra habilidad técnica, sino que también nos ha preparado para enfrentar desafíos futuros en nuestra carrera profesional. Agradecemos profundamente el esfuerzo y la pasión que el profesor ha invertido en nuestra formación, y esperamos poder aplicar y expandir estos conocimientos a medida que avanzamos en nuestra carrera.

Bibliografía

W3Schools. (n.d.). W3Schools online web tutorials. Retrieved August 10, 2024, from <https://www.w3schools.com>

Codex Exempla. (n.d.). Codex Exempla. <http://www.codexexempla.org>

Mozilla Developer Network. (n.d.). MDN web docs. Retrieved August 10, 2024, from <https://developer.mozilla.org>

GeeksforGeeks. (n.d.). GeeksforGeeks. Retrieved August 10, 2024, from <https://www.geeksforgeeks.org>

(n.d.). TutorialsPoint. Retrieved August 10, 2024, from <https://www.tutorialspoint.com>

FreeCodeCamp. (n.d.). FreeCodeCamp. Retrieved August 10, 2024, from <https://www.freecodecamp.org>