

Universidad Pólitecnica Salesiana

Inteligencia artificial 1

Angel Jadan

In [149]: `import sqlite3`

```
#Conexion
conexion = sqlite3.connect('pedidosdb')

cursor = conexion.cursor()
#crear tabla personas
cursor.execute("CREATE TABLE IF NOT EXISTS CLIENTES("+
               "cli_id integer primary key autoincrement not null,"+
               "cli_nombre varchar(250),"+
               "cli_direccion varchar(250),"+
               "cli_telefono varchar(20),"+
               "cli_email varchar(250)"
               ");")

#Crear tabla cuentas
cursor.execute("CREATE TABLE IF NOT EXISTS CUENTAS("+
               "cue_id integer primary key autoincrement not null,"+
               "cue_numerotarjeta varchar(50),"+
               "cue_tarjetabanco varchar(250),"+
               "cue_saldodisponible numeric(10,2),"+
               "fk_cli_id integer,"+
               "foreign key(fk_cli_id) references CLIENTES(cli_id),"+
               "UNIQUE(cue_numerotarjeta)"
               ");")

#Crear tabla producto
cursor.execute("CREATE TABLE IF NOT EXISTS PRODUCTOS("+
               "pro_id integer primary key autoincrement not null,"+
               "pro_nombre varchar(250),"+
               "pro_stock integer,"+
               "pro_precio numeric(10,2)"
               ");")

#Crear tabla pedido
cursor.execute("CREATE TABLE IF NOT EXISTS PEDIDO("+
               "ped_id integer primary key autoincrement not null,"+
               "ped_cantidad int,"+
               "ped_precio numeric(10,2),"+
               "ped_total numeric(10,2),"+
               "ped_estado varchar(20),"+
               "ped_observaciones varchar(100),"+
               "fk_pro_id integer,"
```

```
        "fk_cli_id integer,"+
        "foreign key(fk_pro_id) references PRODUCTOS(pro_id)," +
        "foreign key(fk_cli_id) references CLIENTES(cli_id)" +
        ")")
conexion.commit()
#Cierre conexion
conexion.close()
```

```
In [382]: class Cliente:
    def __init__(self, id, nombre, direccion, telefono, email):
        self.id = id
        self.nombre = nombre
        self.direccion = direccion
        self.telefono = telefono
        self.email = email

class Cuenta:
    def __init__(self, id, tarjeta, banco, saldo, clienteid):
        self.id = id
        self.tarjeta = tarjeta
        self.banco = banco
        self.saldo = saldo
        self.clienteid = clienteid

class Producto:
    def __init__(self, id, nombre, cantidad, precio):
        self.id = id
        self.nombre = nombre
        self.stock = cantidad
        self.precio = precio

class Pedido:
    def __init__(self, id, productoid, cantidad, precio, total, estado, clienteid, observaciones):
        self.id = id
        self.productoid = productoid
        self.cantidad = cantidad
        self.precio = precio
        self.total = total
        self.estado = estado
        self.clienteid = clienteid
        self.observaciones = observaciones
```

```
In [383]: class crud_cliente:
    def insertCliente(self, Cliente):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("INSERT INTO CLIENTES VALUES (null, '"+
                        Cliente.nombre+
                        "', '"+
                        Cliente.direccion
                        +"', '"+
                        Cliente.telefono
                        +"', '"+
                        Cliente.email
                        +
                        "')")
        conexion.commit()
        conexion.close()

    def searchCliente(self, nombre):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("SELECT * FROM CLIENTES WHERE cli_nombre = '"+nombre+"';")
        Cliente = cursor.fetchall()
        conexion.commit()
        conexion.close()

        return Cliente
    def deleteCliente(self, nombre):
        try:
            conexion = sqlite3.connect('pedidosdb')
            cursor = conexion.cursor()
            cursor.execute("DELETE FROM CLIENTES WHERE CLI_NOMBRE='"+nombre+"'")
            conexion.commit()
            conexion.close()
            return True
        except:
            return False

'''cliente = Cliente(0, 'cliente1', 'Cuenca', '956565', 'correo1@gmail.com')
ccliente = crud_cliente()
ccliente.guardarCliente(cliente)'''
```

```
'''ccliente = crud_cliente()
clientes = ccliente.buscarCliente('cliente1')
for cliente in clientes:
    print(cliente[0])
    print(cliente[1])
    print(cliente[2])
    print(cliente[3])
    print(cliente[4])
...

'''ccliente = crud_cliente()
est = ccliente.eliminarCliente('Angel')
print(est)'''
```

Out[383]: "ccliente = crud_cliente()\nest = ccliente.eliminarCliente('Angel')\nprint(est)"

```
In [411]: class crud_cuenta:
    def insertCuenta(self, Cuenta):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("INSERT INTO CUENTAS VALUES (null,'" +
                        Cuenta.tarjeta
                        + "','" +
                        Cuenta.banco
                        + "','" +
                        Cuenta.saldo
                        + "','" +
                        Cuenta.clienteid+
                        "')"
        conexion.commit()
        conexion.close()

    def searchCuenta(self, id):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("SELECT * FROM CUENTAS WHERE cue_id = '"+id+"'")
        Cuenta = cursor.fetchall()
        conexion.commit()
        conexion.close()

        return Cuenta

    def listCuenta(self, clienteid):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("SELECT * FROM CUENTAS WHERE fk_cli_id = '"+clienteid+"'")
        Cuenta = cursor.fetchall()
        conexion.commit()
        conexion.close()

        return Cuenta

    '''
    Recibe el saldo y el numero de cuenta
    ccuenta =crud_cuenta()
    res = ccuenta.updateCuenta('250','1')
    '''

    def updateCuenta(self, saldo, ctaid):
        try:
```

```
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("UPDATE CUENTAS "+
                        "SET CUE_SALDODISPONIBLE="+saldo+
                        " WHERE CUE_ID="+ctaid
                        +";")
        Cuenta = cursor.fetchall()
        conexion.commit()
        conexion.close()
        return True
    except:
        return False

def deleteCuenta(self, id):
    try:
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.update("DELETE FROM CUENTAS WHERE CLI_ID='"+id+"'")
        conexion.commit()
        conexion.close()
        return True
    except:
        return False
```



```
In [442]: class crud_pedido:
    def insertPedido(self, Pedido):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("INSERT INTO PEDIDO VALUES (null,"+
            str(Pedido.cantidad)
            +","+"
            str(Pedido.precio)
            +","+"
            str(Pedido.total)
            +","+"
            str(Pedido.estado)
            +","+"
            str(Pedido.observaciones)
            +","+"
            str(Pedido.productoid)
            +","+"
            str(Pedido.cllienteid)
            +")")
        conexion.commit()
        conexion.close()

    def searchPedido(self, id):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("SELECT * FROM PEDIDO WHERE ped_id = '"+id+"';")
        Pedido = cursor.fetchall()
        conexion.commit()
        conexion.close()

        return Pedido
    def deletePedido(self, id):
        try:
            conexion = sqlite3.connect('pedidosdb')
            cursor = conexion.cursor()
            cursor.execute("DELETE FROM PEDIDO WHERE PED_ID='"+id+"'")
            conexion.commit()
            conexion.close()
            return True
        except:
            return False
```

```
def filtradoPedidoEstado(self, estado):
    conexion = sqlite3.connect('pedidosdb')
    cursor = conexion.cursor()
    sql = "SELECT * FROM PEDIDO WHERE PED_ESTADO = '"+estado+"';"
    #print(sql)
    cursor.execute(sql)
    pedidos = cursor.fetchall()
    conexion.commit()
    conexion.close()
    return pedidos

cpedido = crud_pedido()
res = cpedido.filtradoPedidoEstado("PENDIENTE")
print(res)
```

```
[(1, 2, 30, 60, 'PENDIENTE', '', 1, 1)]
```

```
In [443]: class crud_producto:
    '''
    producto = Producto('0', 'medias', '20', '3.00')
    cproducto = crud_producto()
    res = cproducto.insertProducto(producto)
    '''

    def insertProducto(self, Producto):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("INSERT INTO PRODUCTOS VALUES (null, '"+
                        Producto.nombre
                        +"', "+
                        Producto.stock
                        +"', "+
                        Producto.precio+
                        ")")
        conexion.commit()
        conexion.close()

    '''
    cproducto = crud_producto()
    productos = cproducto.searchProducto('2')
    for producto in productos:
    print(producto[3])

    '''

    def searchProducto(self, id):
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("SELECT * FROM PRODUCTOS WHERE pro_id = '"+id+"'")
        Producto = cursor.fetchall()
        conexion.commit()
        conexion.close()

        return Producto
    '''

    cproducto = crud_producto()
    est = cproducto.updateStock('25', '1')
    '''

    def updateStock(self, nuevostock, productoid):
        conexion = sqlite3.connect('pedidosdb')
```

```
cursor = conexion.cursor()
cursor.execute("UPDATE PRODUCTOS "+
               "SET PRO_STOCK="+nuevostock+
               " WHERE PRO_ID="+productoid
               +";")
Producto = cursor.fetchall()
conexion.commit()
conexion.close()
return True

def deleteProducto(self, id):
    try:
        conexion = sqlite3.connect('pedidosdb')
        cursor = conexion.cursor()
        cursor.execute("DELETE FROM PRODUCTOS WHERE PRO_ID='"+id+"'")
        conexion.commit()
        conexion.close()
        return True
    except:
        return False
```

Gestión Cliente

In [444]: **class** GestionCliente:

```
#res = GestionCliente.guardarCliente('cliente2','cuenca','262611','mail@mail.com')
def guardarCliente(nombre, direccion, telefono, correo):
    cliente = Cliente(0,nombre,direccion, telefono, correo)
    ccliente = crud_cliente()
    ccliente.insertCliente(cliente)
    return True

'''
cliente = GestionCliente.buscarCliente('Juan')
'''
def buscarCliente(nombre):
    ccliente = crud_cliente()
    clientes = ccliente.searchCliente(str(nombre))
    for cliente in clientes:
        cliEncontrado = Cliente(cliente[0],
                                   cliente[1],
                                   cliente[2],
                                   cliente[3],
                                   cliente[4])

    return cliEncontrado

#res = GestionCliente.guardarCuenta('1165161651','solidario',250.00,2)
def guardarCuenta(tarjeta, banco, saldo, clienteid):
    cuenta = Cuenta(0,tarjeta,banco,str(saldo),str(clienteid))
    ccuenta = crud_cuenta()
    est = ccuenta.insertCuenta(cuenta)
    return est

'''
cuentas = GestionCliente.listarCuentas(2)
for cuenta in cuentas:
    print(cuenta[0])
    print(cuenta[1])
'''
def listarCuentas(clienteid):
    ccuenta = crud_cuenta()
    return ccuenta.listCuenta(str(clienteid))
'''
```

Solo devuelve una cuenta, recibe el id de la cuenta y devuelve esa cuenta

```

cta = GestionCliente.buscarCuenta(1)
'''
def buscarCuenta(cuentaaid):
    ccuenta = crud_cuenta()
    cuentas = ccuenta.searchCuenta(str(cuentaaid))
    for cuenta in cuentas:
        cta = Cuenta(
            cuenta[0],
            cuenta[1],
            cuenta[2],
            cuenta[3],
            cuenta[4]
        )
    return cta

'''
Realiza el debito de una cuenta indicada, recibe el numero de cuenta, el valor del debito y el id del cliente al que pertenece la cuenta.
Devuelve un True, se se realizo el debito, y False, si no se realiza el debito.

cta = GestionCliente.debitarCuenta(1,20,2)
'''
def debitarCuenta(cuentaaid, valordebito, clienteid):
    cta = GestionCliente.buscarCuenta(str(cuentaaid))
    saldoanterior = float(cta.saldo)
    nuevosaldo = float(saldoanterior)-float(valordebito)

    ccuenta = crud_cuenta()
    return ccuenta.updateCuenta(str(nuevosaldo), str(cuentaaid))

```

Gestión producto

```
In [445]: class GestionProducto:
    def guardarProducto(nombre, stock, precio):
        producto = Producto(0,nombre,stock, precio)
        cproducto = crud_producto()
        res = cproducto.insertProducto(producto)

    """
    Recibe el id del producto y devuelve el producto encontrado
    producto = GestionProducto.buscarProducto(1)
    """

    def buscarProducto(id):
        cproducto = crud_producto()
        productos = cproducto.searchProducto(str(id))
        for producto in productos:
            pro = Producto(producto[0],
                            producto[1],
                            producto[2],
                            producto[3])

            return pro
    def actualizarStock(productoid, nuevostock):
        cproducto = crud_producto()
        estado = cproducto.updateStrock(nuevostock, productoid)
        return estado
```

Gestión Pedido

In [446]: **class** GestionPedido:

```
def realizarPedido( productoid, cantidad,precio, estado, clienteid, observaciones):  
    total = float(cantidad) * float(precio)  
    pedido = Pedido(id, productoid, cantidad,precio,total, estado, clienteid, observaciones)  
    cpedido = crud_pedido()  
    res = cpedido.insertPedido(pedido)  
    return res  
'''
```

*Recibe el id del cliente,
y devuelve un True si es que tiene cuenta o false si no tiene cuenta.*

```
num = GestionPedido.verificarCuenta(2)  
print(num)
```

```
'''  
def verificarCuenta(clienteid):  
    n = 0  
    cuentas = GestionCliente.listarCuentas(clienteid)  
    for cuenta in cuentas:  
        n=n + 1  
  
    return n  
'''
```

*Recibe el id del cliente
y devuelve la cantidad de saldo disponible*

```
num = GestionPedido.verificarSaldo(2)  
print(num)  
'''
```

```
def verificarSaldo(clienteid):  
    saldo = 0.00  
    cuentas = GestionCliente.listarCuentas(clienteid)  
    for cuenta in cuentas:  
        saldo = saldo + cuenta[3]  
    return saldo
```

Recibe el id del producto y devuelve el stock

```
num = GestionPedido.verificarStock(1)
```



```

    print(num)
    '''
    def verificarStock(idproducto):
        stock = 0
        cproducto = crud_producto()
        cproducto.searchProducto(str(idproducto))
        for producto in productos:
            stock = producto[2]
        return stock

    def filtrarEstado(estado):
        cpedido = crud_pedido()
        pedidos = cpedido.filtradoPedidoEstado(str(estado))

        return pedidos

#realizarPedido( productoid, cantidad,precio, estado, clienteid, observaciones)
#res = GestionPedido.realizarPedido('1','2','30.00','PENDIENTE','1','')
pedidos = GestionPedido.filtrarEstado('PENDIENTE')
print(pedidos)

```

```
[(1, 2, 30, 60, 'PENDIENTE', '', 1, 1)]
```

Instancia de cobro

```

In [454]: class Cobro:

    def pedidosPendientes(estado):
        pedidos = GestionPedido.filtrarEstado(estado)
        return pedidos

```

Ventana registro cliente

```
In [455]: #Interfaz grafica
from tkinter import *
from tkinter import messagebox

#Crear la ventana raiz
ventana = Tk()

#Cambio den el tamaño de La ventana
ventana.geometry("750x450")

ventana.configure(background="white")
#Bloquear el tamaño de La ventana
ventana.resizable(0,0)

#Etiqueta de texto
lblnombre=Label(ventana,text="Nombre",background="white").place(x=0,y=10)
lbldireccion=Label(ventana,text="Direccion",background="white").place(x=0,y=30)
lbltelefono=Label(ventana,text="Telefono",background="white").place(x=0,y=50)
lblcorreo=Label(ventana,text="Correo",background="white").place(x=0,y=70)

nombre=StringVar()
txtnombre=Entry(ventana,textvariable=nombre).place(x=100,y=10)
#name=nombre.get()

direccion=StringVar()
txtdireccion=Entry(ventana,textvariable=direccion).place(x=100,y=30)

telefono=StringVar()
txttelefono=Entry(ventana,textvariable=telefono).place(x=100,y=50)

correo=StringVar()
txtcorreo=Entry(ventana,textvariable=correo).place(x=100,y=70)

def guardar():
    nomb = nombre.get()
    dir = direccion.get()
    tel = telefono.get()
    corr = correo.get()

    res = GestionCliente.guardarCliente(nomb,dir,tel,corr)
```

```
if res == True:
    messagebox.showinfo(message="Cliente guardado", title="Sms")
    nombre.set("")
    direccion.set("")
    telefono.set("")
    correo.set("")
else:
    messagebox.showinfo(message="No se ha podido guardar revise por favor", title="Sms")

#Boton de comando
cFuncion=Button(ventana, command = guardar , text="Guardar",width=10,height=2).place(x=110, y=100)

ventana.mainloop()
```



tk



Nombre

Andres Espinoza

Direccion

Bolivar

Telefono

1654648

Correo

aespinoza@gmail.com

Guardar



Nombre

Andres Espinoza

Direccion

Bolivar

Telefono

1654648

Correo

aespinoza@gmail.com

Guardar



Sms



Cliente guardado

Aceptar

Vista pedidos

```
In [392]: #Interfaz grafica
from tkinter import *
from tkinter import messagebox

#Crear la ventana raiz
ventana = Tk()

#Cambio den el tamaño de la ventana
ventana.geometry("750x450")

ventana.configure(background="white")
#Bloquear el tamaño de la ventana
ventana.resizable(0,0)

id = 0

#Etiqueta de texto de datos cliente
lblnombre=Label(ventana,text="Cliente",background="white").place(x=0,y=10)
lbldireccion=Label(ventana,text="Direccion",background="white").place(x=0,y=30)
lbltelefono=Label(ventana,text="Telefono",background="white").place(x=0,y=50)
lblcorreo=Label(ventana,text="Correo",background="white").place(x=0,y=70)

nombre=StringVar()
txtnombre=Entry(ventana,textvariable=nombre).place(x=100,y=10)
#name=nombre.get()

direccion=StringVar()
txtdireccion=Entry(ventana,textvariable=direccion,state='disabled').place(x=100,y=30)

telefono=StringVar()
txttelefono=Entry(ventana,textvariable=telefono,state='disabled').place(x=100,y=50)

correo=StringVar()
txtcorreo=Entry(ventana,textvariable=correo,state='disabled').place(x=100,y=70)

def buscar():
    nomb = nombre.get()
    dir = direccion.get()
    tel = telefono.get()
    corr = correo.get()
```



```

cliente = Cliente(0,"","","","")
cliente = GestionCliente.buscarCliente(nomb)

if cliente.id>0:
    id = cliente.id
    #print(id)

    direccion.set(cliente.direccion)
    telefono.set(cliente.telefono)
    correo.set(cliente.email)

else:
    messagebox.showinfo(message="No se ha encontrado el cliente", title="Sms")

def verificarCuenta():
    GestionPedido.verificarCuenta(2)

#Boton de buscar cliente
cFuncion=Button(ventana, command = buscar , text="Buscar cliente",width=15,height=2).place(x=110, y=100)

#Etiquetas de datos para buscar productos
lblcodigo=Label(ventana,text="Codigo",background="white").place(x=350,y=10)
lblproducto=Label(ventana,text="Producto",background="white").place(x=350,y=30)
lblstock=Label(ventana,text="Stock",background="white").place(x=350,y=50)
lblcosto=Label(ventana,text="Costo",background="white").place(x=350,y=70)

codigo=StringVar()
txtnombre=Entry(ventana,textvariable=codigo).place(x=450,y=10)

prod=StringVar()
txtprod=Entry(ventana,textvariable=prod, state='disabled').place(x=450,y=30)

stock=StringVar()
txttelefono=Entry(ventana,textvariable=stock,state='disabled').place(x=450,y=50)

costo=StringVar()

```

```
txtcosto=Entry(ventana,textvariable=costo,state='disabled').place(x=450,y=70)

def buscarProd():
    cod = codigo.get()

    producto = GestionProducto.buscarProducto(cod)
    #print(producto.nombre)
    prod.set(producto.nombre)
    stock.set(producto.stock)
    costo.set(producto.precio)

#Boton de buscar producto
cFuncion=Button(ventana, command = buscarProd , text="Buscar producto",width=15,height=2).place(x=450, y=100)


#Etiquetas de datos para hacer pedido
lblpedido=Label(ventana,text="Pedido",background="white").place(x=100,y=160)
lblcantidad=Label(ventana,text="Cantidad",background="white").place(x=0,y=190)
lbltotal=Label(ventana,text="Total",background="white").place(x=0,y=210)

cantidad=StringVar()
txtcantidad=Entry(ventana,textvariable=stock).place(x=150,y=190)

total=StringVar()
txtcosto=Entry(ventana,textvariable=costo,state='disabled').place(x=150,y=210)

ventana.mainloop()
```



tk



Cliente

Direccion

Telefono

Correo

Buscar cliente

Codigo

Producto

Stock

Costo

Buscar producto

Pedido

Cantidad

Total

