TECHNICAL MANUAL

DIANA ESTEFANIA BERDUCIDO DOMINGO

202000277

AUX. JACKELINE BENITEZ

ENGLISH

PARADIGM USED: POO

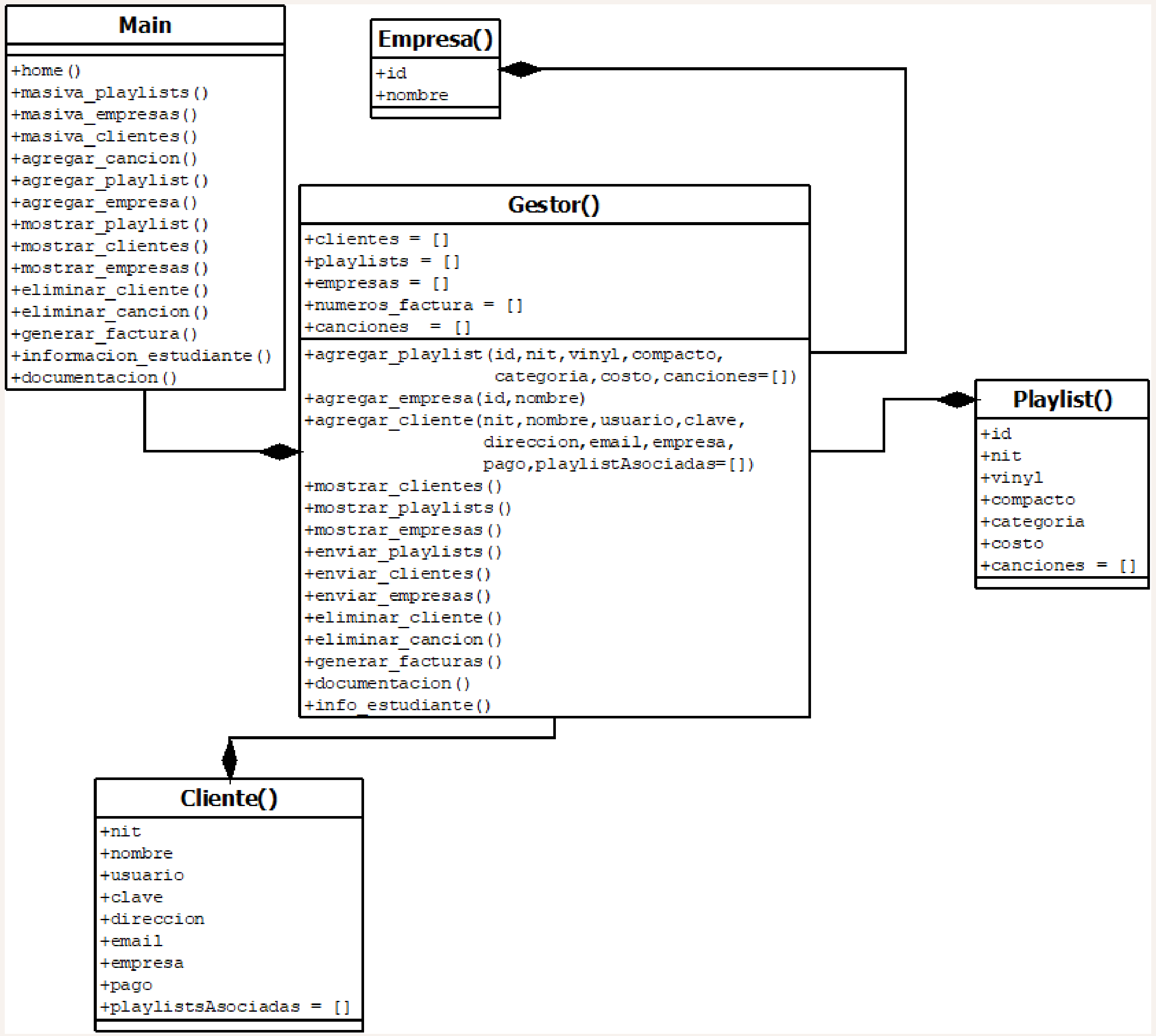
By creating objects, the information for each XML was stored.

Customer, playlist, and enterprise objects were created.

For the API it is recommended to start with the creation of playlists and companies, because the client carries information about each one.

OBJECTIVE OF THE PROGRAM

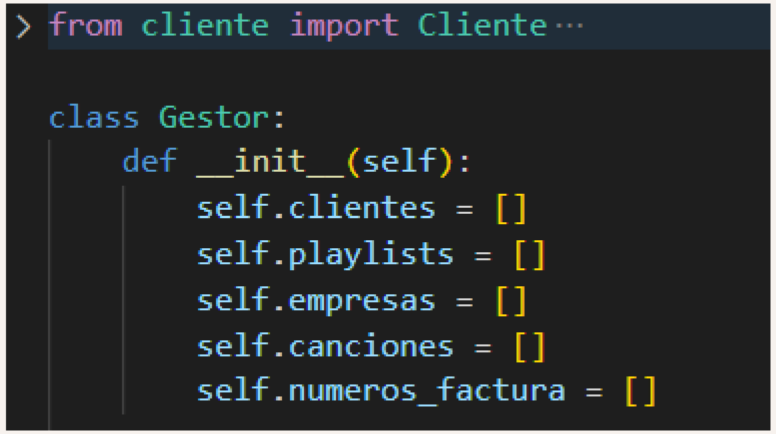
- An API was created, it behaves in different ways. The REST API aims to receive requests such as: GET, DELETE, POST.
- From the beginning, three XML files are received that have the objective of collecting information related to playlists, companies and customers. Initially it joins all the information to later act depending on the request that is executed.
- You can add more customers, playlists, and businesses using XML-long files. You can review what's stored in the system and even delete a client or song from a playlist.
- If necessary, help information, such as student data and program documentation, can be reviewed.



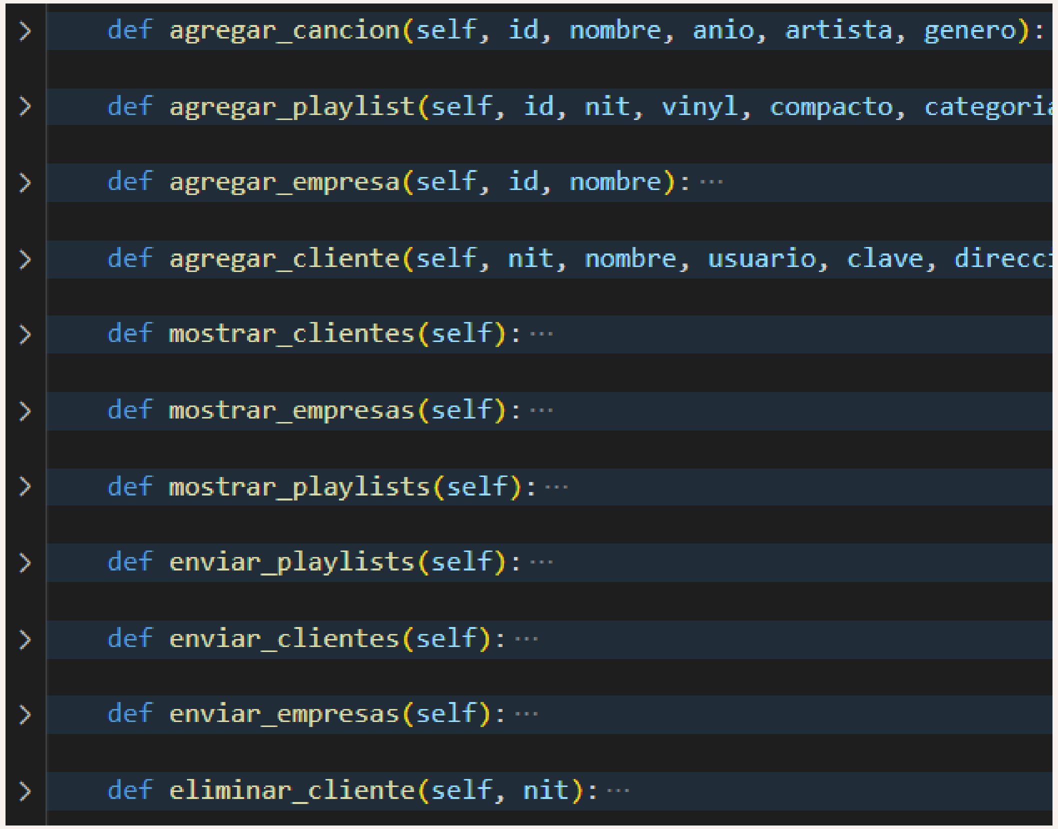
CLASS DIAGRAM

GESTOR.PY

- In this class all the methods that would be responsible for saving information and displaying it, also deleting it were created.



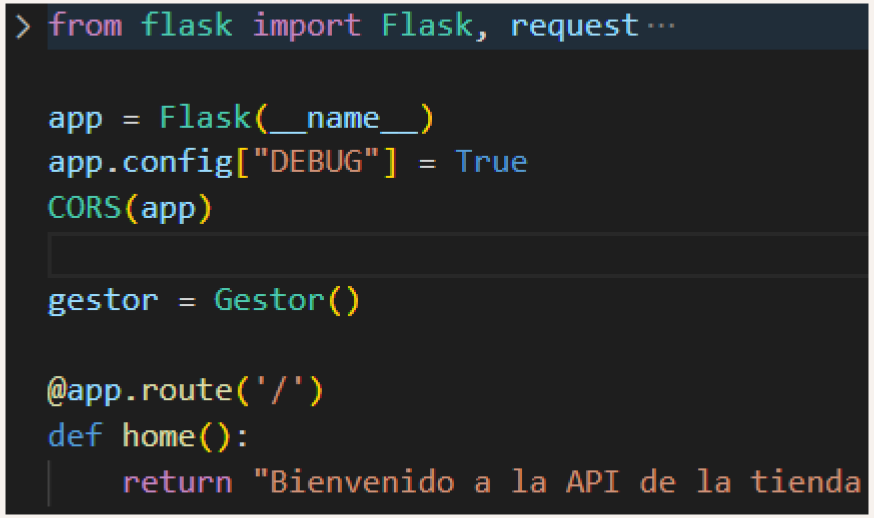
```
from cliente import Cliente···

class Gestor:
    def __init__(self):
        self.clientes = []
        self.playlists = []
        self.empresas = []
        self.canciones = []
        self.numeros_factura = []
```



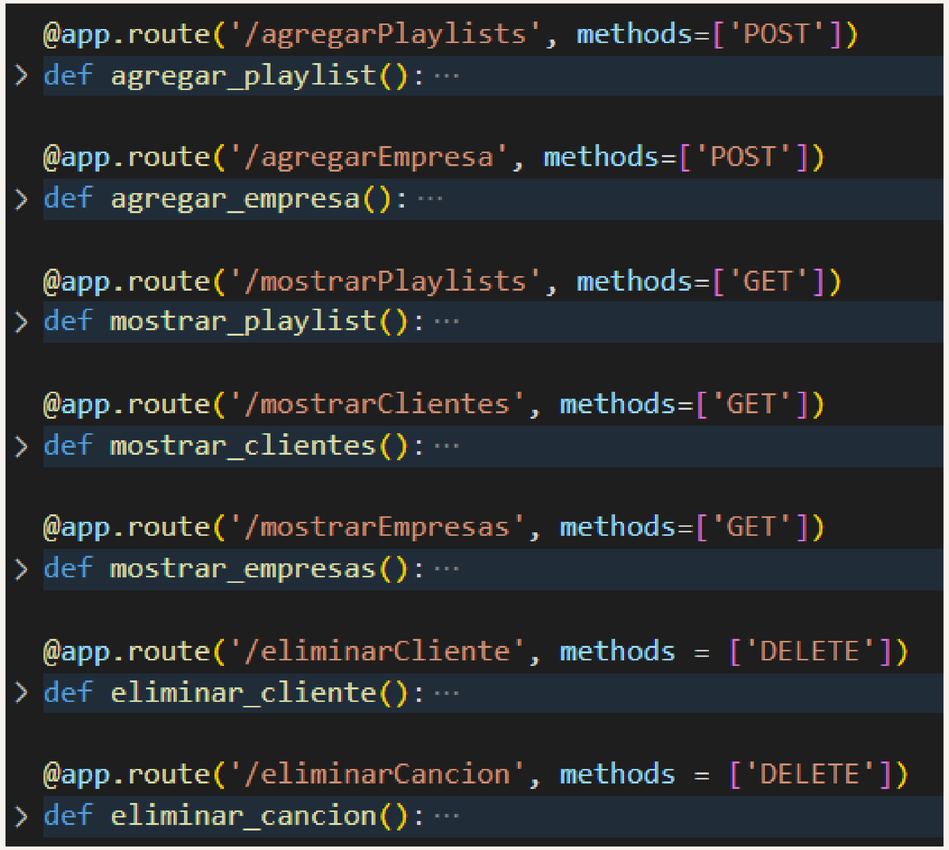
```
    def agregar_cancion(self, id, nombre, anio, artista, genero):

    def agregar_playlist(self, id, nit, vinyl, compacto, categoria

    def agregar_empresa(self, id, nombre):···

    def agregar_cliente(self, nit, nombre, usuario, clave, direcc

    def mostrar_clientes(self):···

    def mostrar_empresas(self):···

    def mostrar_playlists(self):···

    def enviar_playlists(self):···

    def enviar_clientes(self):···

    def enviar_empresas(self):···

    def eliminar_cliente(self, nit):···
```

MAIN.PY (API FLASK)

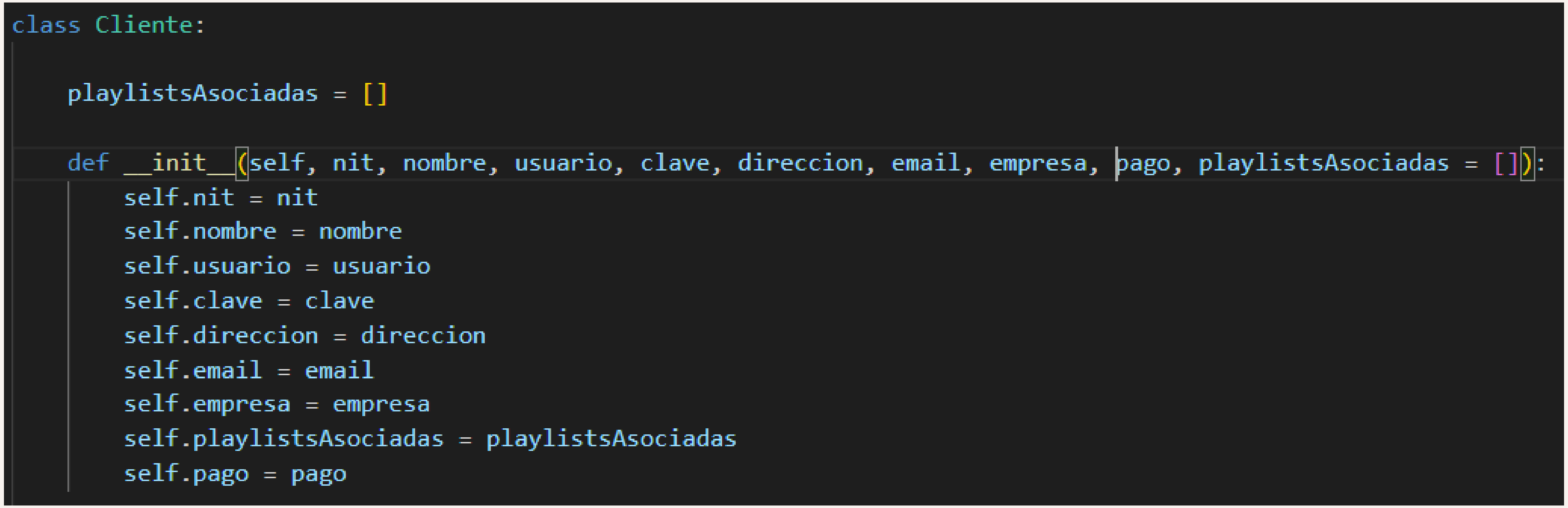
- After importing the necessary libraries, the home() path was created to verify proper operation (installation).



```
from flask import Flask, request···

app = Flask(__name__)
app.config["DEBUG"] = True
CORS(app)

gestor = Gestor()

@app.route('/')
def home():
    return "Bienvenido a la API de la tienda
```



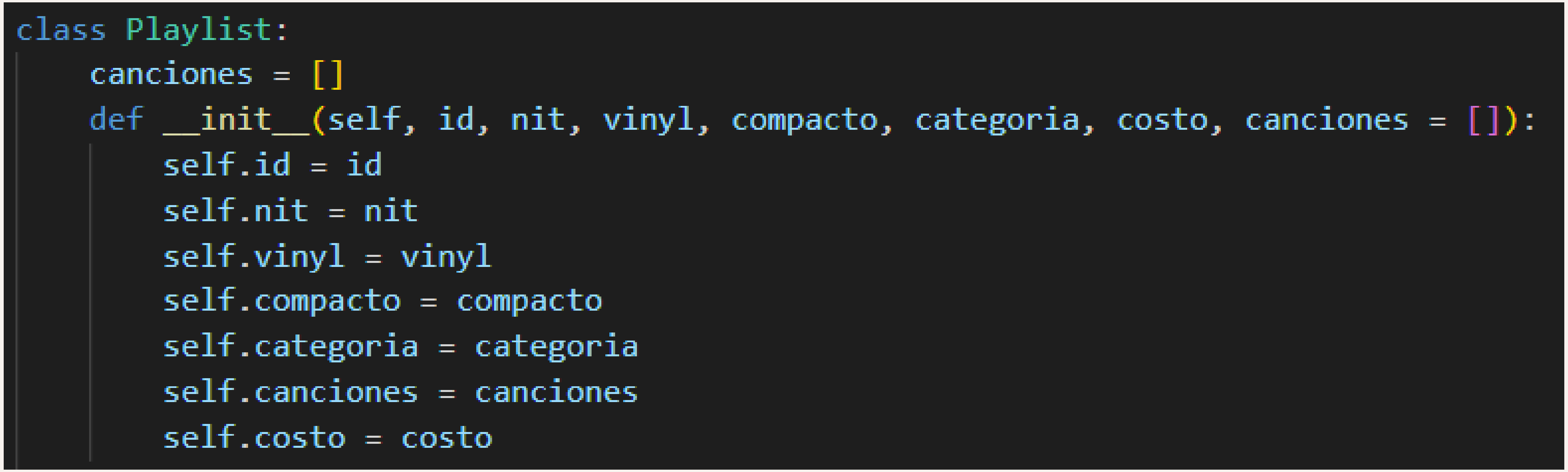
```
@app.route('/agregarPlaylists', methods=['POST'])
def agregar_playlist():···

@app.route('/agregarEmpresa', methods=['POST'])
def agregar_empresa():···

@app.route('/mostrarPlaylists', methods=['GET'])
def mostrar_playlist():···

@app.route('/mostrarClientes', methods=['GET'])
def mostrar_clientes():···

@app.route('/mostrarEmpresas', methods=['GET'])
def mostrar_empresas():···

@app.route('/eliminarCliente', methods = ['DELETE'])
def eliminar_cliente():···

@app.route('/eliminarCancion', methods = ['DELETE'])
def eliminar_cancion():···
```



```
class Cliente:

    playlistsAsociadas = []

    def __init__(self, nit, nombre, usuario, clave, direccion, email, empresa, pago, playlistsAsociadas = []):
        self.nit = nit
        self.nombre = nombre
        self.usuario = usuario
        self.clave = clave
        self.direccion = direccion
        self.email = email
        self.empresa = empresa
        self.playlistsAsociadas = playlistsAsociadas
        self.pago = pago
```

CLIENTE.PY

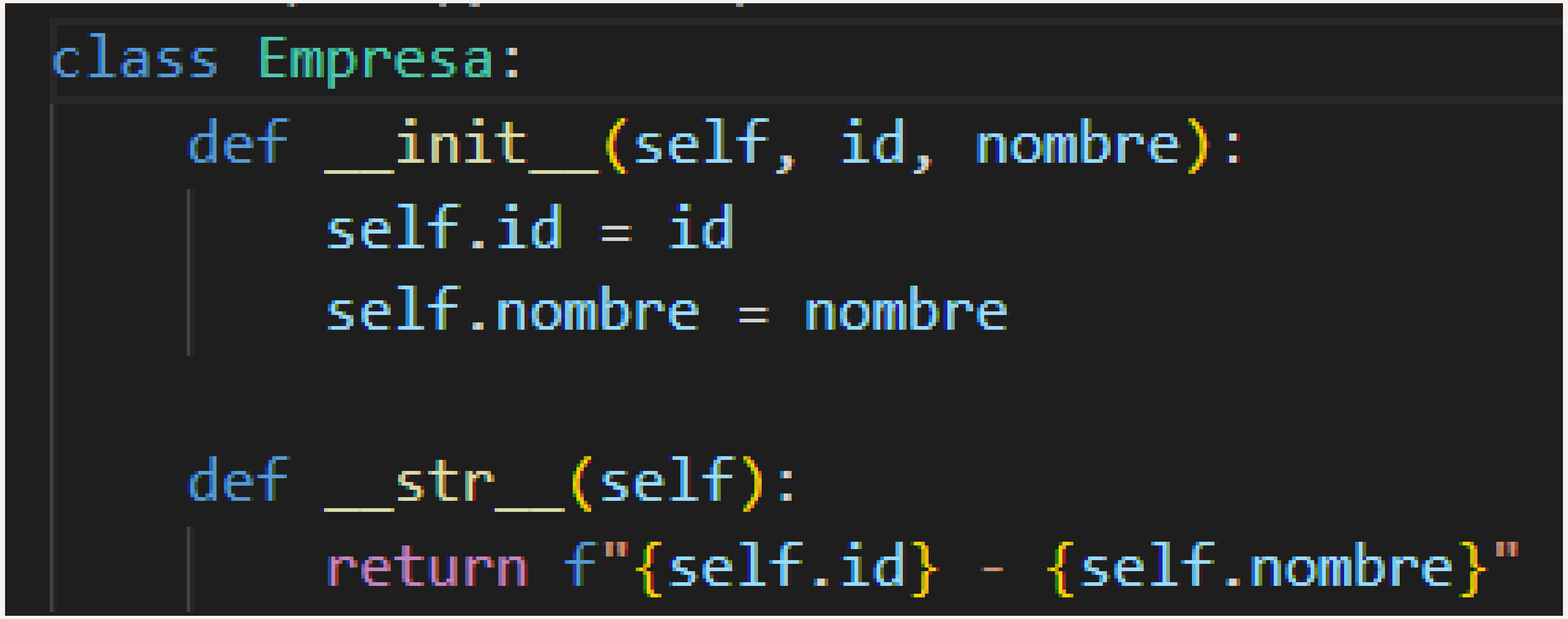
- This is the class responsible for creating the client object. Which receives nit, username, password (password), customer name, company ID, address, email, payment and associated playlists.



```python
class Playlist:
    canciones = []
    def __init__(self, id, nit, vinyl, compacto, categoria, costo, canciones = []):
        self.id = id
        self.nit = nit
        self.vinyl = vinyl
        self.compacto = compacto
        self.categoria = categoria
        self.canciones = canciones
        self.costo = costo
```

PLAYLIST.PY

- This is the class responsible for creating the playlist object. Which receives id, nit, vinyl, compact, category, cost and a list of songs. It intends to connect through ids and nits that are associated with the client.



```python
class Empresa:
    def __init__(self, id, nombre):
        self.id = id
        self.nombre = nombre

    def __str__(self):
        return f"{self.id} - {self.nombre}"
```

EMPRESA.PY

- This is the class responsible for creating the Enterprise object. Which receives id and name of the company. Each client goes through their elements and they associate with the company. This partnership is mostly important in invoice generation and reporting.