MANUAL TÉCNICO

DIANA ESTEFANIA BERDUCIDO DOMINGO

202000277

AUX. JACKELINE BENITEZ

ESPAÑOL

Mediante la creación de objetos se almacenó la información de cada XML.

Se crearon objetos relacionados con cliente, playlist y empresa.

Para la API se recomienda iniciar con la creación de playlists y empresas, porque el cliente lleva información de cada uno.

OBJETIVO DEL PROGRAMA

- Se creó una API, esta se comporta de diferentes maneras. La API REST tiene el objetivo de recibir peticiones del estilo: GET, DELETE, POST.
- De inicio se reciben tres archivos XML que tienen el objetivo de recolectar información relacionada con playlists, empresas y clientes. Inicialmente une toda la información para posteriormente actuar dependiendo de la request que se ejecute.
- Puede agregar más clientes, playlists y empresas mediante archivos con extensión XML. Puede revisar lo que está almacenado en el sistema e incluso borrar un cliente o una canción de una playlist.
- De ser necesario se puede revisar información de ayuda, como datos del estudiante y documentación del programa.

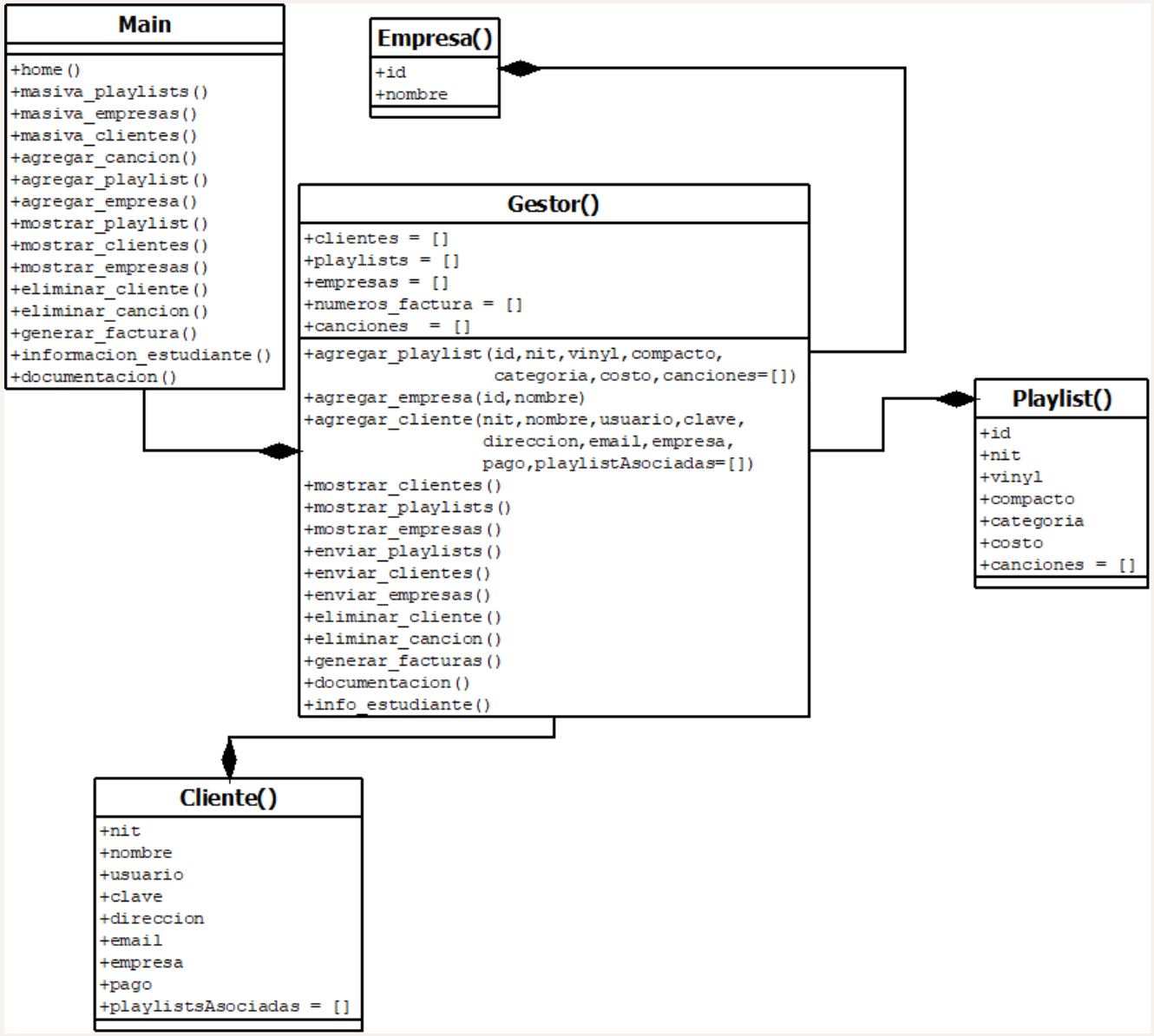
Main
+home()
+masiva_playlists()
+masiva_empresas()
+masiva_clientes()
+agregar_cancion()
+agregar_playlist()
+agregar_empresa()
+mostrar_playlist()
+mostrar_clientes()
+mostrar_empresas()
+eliminar_cliente()
+eliminar_cancion()
+generar_factura()
+informacion_estudiante()
+documentacion()
Empresa()
+id
+nombre
Gestor()
+clientes = []
+playlists = []
+empresas = []
+numeros_factura = []
+canciones = []
+agregar_playlist(id,nit,vinyl,compacto,
categoria,costo,canciones=[])
+agregar_empresa(id,nombre)
+agregar_cliente(nit,nombre,usuario,clave,
direccion,email,empresa,
pago,playlistAsociadas=[])
+mostrar_clientes()
+mostrar_playlists()
+mostrar_empresas()
+enviar_playlists()
+enviar_clientes()
+enviar_empresas()
+eliminar_cliente()
+eliminar_cancion()
+generar_facturas()
+documentacion()
+info_estudiante()
Playlist()
+id
+nit
+vinyl
+compacto
+categoria
+costo
+canciones = []
Cliente()
+nit
+nombre
+usuario
+clave
+direccion
+email
+empresa
+pago
+playlistsAsociadas = []


DIAGRAMA
DE CLASES

GESTOR.PY

- En esta clase se crearon todos los métodos que se encargarían de guardar información y mostrarla, también eliminar.

```python
from cliente import Cliente···

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

```python
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, direcci
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)

- Posterior a importar las librerías necesarias, se creó la ruta de home(), para verificar un correcto funcionamiento (instalación).

```python
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
```

```python
@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():···
```

```python
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
```

- Esta es la clase encargada de la creación del objeto cliente. La cual recibe nit, usuario, clave (contraseña), nombre del cliente, id de la empresa, dirección, email, pago y playlists asociadas.

```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
```

- Esta es la clase encargada de la creación del objeto playlist. La cual recibe id, nit, vinyl, compacto, categoria, costo y una lista de canciones. Pretende conectarse mediante id y nit que se asocian al cliente.

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

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

- Esta es la clase encargada de la creación del objeto Empresa. La cual recibe id y nombre de la empresa. Cada cliente recorre sus elementos y se asocian con la empresa. Esta asociación es mayoritariamente importante en la generación de factura y reporte.