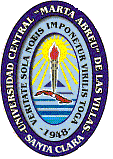
******

UNIVERSIDAD CENTRAL “MARTA ABREU” DE LAS VILLAS

FACULTAD DE MATEMÁTICA FÍSICA Y COMPUTACIÓN

CARRERA CIENCIA DE LA COMPUTACIÓN

**TÍTULO:**

**Proyecto “Sistema digital para Tienda de Música”**

**Programación en C/C++**

**Autora:**

Adriana María Garay Chávez

**Curso 2024**

## Breve introducción y presentación de la problemática

La idea principal de este proyecto es programar un sistema para una tienda digital de música. En él están presentes elementos como los usuarios del sistema (las personas con acceso a él que serían los trabajadores de la tienda y sus clientes), los artistas con los que la tienda tiene contratos y sus respectivos álbumes y canciones. Este sistema debe permitir a los clientes explorar una amplia variedad de canciones de diferentes artistas, buscar canciones específicas, y comprar las que gusten, añadiéndolas a su biblioteca personal.

Para ello se ha aplicado el paradigma de programación orientada a objetos (POO) y el lenguaje de programación C++. Se ha creado una pequeña base de datos en un archivo de texto que se corresponde con el banco de canciones con el que cuenta la tienda en primera instancia. Se ha diseñado una interfaz por consola amigable al usuario objetivo de modo que puedan ejecutar las operaciones posibles de forma fluida y eficiente.

## ¿Cómo funciona el sistema? Descripción de los casos de uso

El programa cuando se inicia muestra dos opciones numeradas: una para iniciar sesión como Cliente y otra para iniciar sesión como Administrador (también se muestra la opción salir).

**Menú de usuario:**

1. Cliente

1- Consultar las canciones en su biblioteca

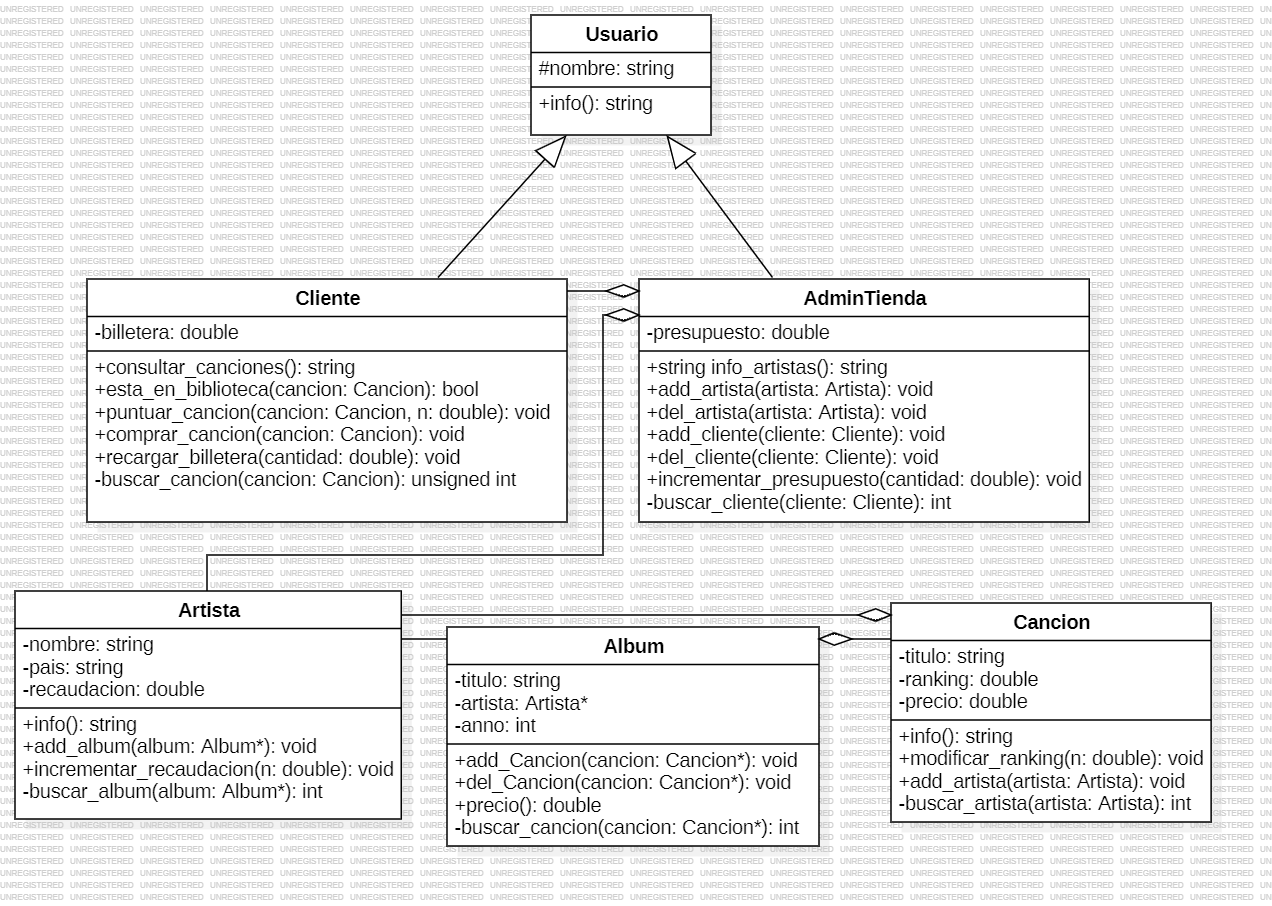
2- Puntuar una cancion

3- Comprar cancion

4- Recargar billetera

1. Salir
2. Administrador
3. Registrar un nuevo artista
4. Registrar un nuevo cliente
5. Salir
6. Salir

## Análisis y diseño de la aplicación

Para resolver la problemática descrita, siguiendo el paradigma de POO y teniendo en cuenta la creación de la interfaz por consola en C++, fueron creadas las clases y funciones que se observan en el diagrama a continuación.

**Usuario**: Esta es la clase base para todos los usuarios del sistema. Tiene un atributo protected nombre que representa el nombre del usuario. La clase proporciona métodos para obtener y establecer el nombre del usuario.

**AdminTienda**: Esta clase hereda de la clase Usuario y representa al administrador de la tienda. Tiene atributos privados para almacenar la lista de artistas contratados (art\_contratados), la lista de clientes (clientes) y el presupuesto de la tienda (presupuesto). Proporciona métodos para buscar, agregar y eliminar artistas y clientes, incrementar el presupuesto, obtener información de los artistas, y obtener y establecer el presupuesto, la lista de artistas contratados y la lista de clientes.

**Cliente**: Esta clase también hereda de la clase Usuario y representa a un cliente de la tienda. Tiene atributos privados para almacenar la biblioteca de canciones del cliente (biblioteca) y la cantidad de dinero en la billetera del cliente (billetera). Proporciona métodos para consultar canciones, verificar si una canción está en la biblioteca, puntuar una canción, comprar una canción, recargar la billetera, y obtener y establecer la billetera y la biblioteca.

**Artista**: Esta clase representa a un artista en la tienda de música. Tiene atributos privados para almacenar el nombre del artista (nombre), el país del artista (pais), la lista de álbumes del artista (albumes) y la recaudación del artista (recaudacion). Proporciona métodos para agregar un álbum, incrementar la recaudación, obtener y establecer su nombre, su país, su recaudación y los álbumes.

**Album**: Esta clase representa un álbum en la tienda de música. Tiene atributos privados para almacenar el título del álbum (titulo), el artista del álbum (artista), la lista de canciones en el álbum (canciones) y el año del álbum (anno). Proporciona métodos para agregar y eliminar canciones, calcular el precio del álbum, y obtener y establecer el título, el artista y el año.

**Cancion**: Esta clase representa una canción en la tienda de música. Tiene atributos privados para almacenar el título de la canción (titulo), el ranking de la canción (ranking), el precio de la canción (precio) y la lista de artistas de la canción (artistas). Proporciona métodos para modificar el ranking, agregar un artista, y obtener y establecer el título, el ranking, el precio y los artistas.

## Funcionalidades (se mencionan solo algunas)

La sobrecarga de operadores es una característica muy útil en C++ que permite definir el comportamiento de los operadores para tipos de datos personalizados. Esta facilidad que brinda el lenguaje ha sido muy útil en lo que respecta a este proyecto, especialmente para los métodos de búsqueda. De forma específica se sobrecargaron los operadores == y != para los tipos de datos Artista, Album y Cancion, así los métodos de búsqueda filtran a partir del valor booleano que devuelven las comparaciones.

Debido a la existencia de algunas dependencias circulares entre una clase y otra, por ejemplo la que existe entre las clases Artista, Album y Cancion (entre las que hay relaciones de asociación y agregación) el uso de punteros fue imprescindible, lo cual también contribuyó con la eficiencia de memoria ya que solo se necesitaba referenciar a un objeto existente y no crear una copia del objeto (en este caso, esto último hubiese sido muy costoso e ineficiente ya que se está trabajando con listas compuestas por varios objetos).

Se aprovecharon también las funcionalidades de la clase vector < > para el trabajo con todas estas listas de objetos, que, de lo contrario, haberlo hecho con arreglos (ya sea dinámicos o no) hubiese sido muy trabajoso.

Se hizo uso de una variable global nombrada admin: la variable controladora de todo el sistema, es decir, el equivalente a la propia tienda de música. Se decidió crearla porque era necesaria una comunicación “en tiempo real” entre las clases que iba algo más allá de una asociación y porque, del modo en que fue concebido, no tenía sentido que se instanciara más de un objeto de la clase AdminTienda.

## Excepciones

En lo que respecta al manejo de excepciones interesaba mucho validar los datos introducidos por consola, para que el programa pudiera seguir su curso normal y se evitara una detención abrupta.

Para lograr esto se utilizaron las herramientas pertinentes del lenguaje: la cláusula throw y los bloques try-catch; pero además se hizo uso de las funciones lambda. Era necesario que los datos pudieran convertirse a los tipos de datos int y double, por ejemplo, pero también que cumplieran con cierta condición, como estar en un rango determinado. Para ello fueron implementadas en main.cpp las funciones int convertir\_a\_int(const string& cadena, const function<bool(int)>& validar) y double convertir\_a\_double(const string& cadena, const function<bool(double)>& validar). Puede observarse que el segundo parámetro es la función lambda mencionada, de modo que, cada vez que era necesario validar un dato se llamaba a esta función y como segundo parámetro se pasaba la condición que debía cumplir el número.

## Determinaciones finales

A partir de la problemática presentada se cumplieron los objetivos de utilizar componentes de la Standard Template Library (STL), implementar una aplicación utilizando POO y la utilización de sobrecarga de operadores, todas cuestiones que resultaron muy ventajosas y casi indispensables para un tipo de proyecto como este.

Quedaron implementadas funciones que pueden resultar útiles para el mantenimiento del código y para complejizarlo en algún futuro según las necesidades del cliente.

A partir de los conocimientos adquiridos en la asignatura se logró diseñar e implementar esta aplicación en la que se pusieron en práctica varios elementos del lenguaje tratando de obtener el resultado más óptimo posible.