|  | Metodologías Programación II  Trabajo Final  Metodologías Programación II  Comisión 1 | Osés Lucia |
| --- | --- | --- |

1. Introducción.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pág 02
2. Diagrama Entidad Relación Spotify.\_\_\_\_\_\_\_\_\_\_\_ pág 03
3. Diagrama Secuencia Spotify. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pág 04
4. Objetivo descripción del problema planteado. \_\_\_ pág 04
5. Especificación e implementación de las clases utilizadas.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pág 05
6. La implementación de la aplicación desarrollada.\_ pág 12
7. Metodología de trabajo y prácticas ágiles utilizada. pág 15
8. Frameworks que se podrían utilizar para realizar un desarrollo orientado a objetos. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pág 34
9. Uso de Patrones en el contexto del problema planteado.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pág 34
10. Descripción de los repositorios se podrían utilizar para el desarrollo de software colaborativo. \_\_\_\_\_\_\_\_\_\_\_ pág 36

**Introducción**

Se realizará el diseño e implementación de un negocio Spotify en el lenguaje Smalltalk

1.1 Propósito

El propósito del proyecto es el desarrollo de un negocio, donde debemos entender el problema y la comunicación con el cliente, así poder implementar los contenidos vistos en la cátedra metodologías programación II de la universidad Arturo Jauretche.

1.2 Ámbito del sistema

Spotify es un servicio de música y podcasts que te da acceso a millones de canciones y a otro contenido de creadores de todo el mundo. Este modelo de negocio proporciona un acceso básico gratuito respaldado con publicidad, y otro con una suscripción de pago.

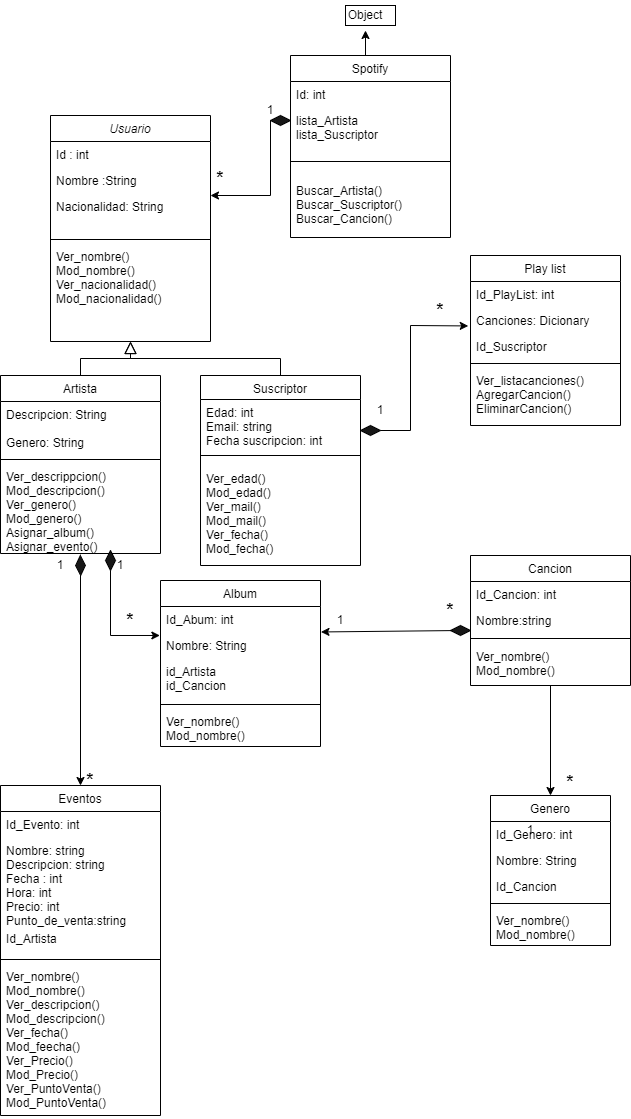
Para la distribución de archivos musicales por [internet](https://lovtechnology.com/la-historia-de-internet/), la compañía cuenta con varios servidores Streaming y transferencia de redes [peer-to-peer](https://lovtechnology.com/todo-sobre-la-red-peer-to-peer/) (P2P). Para hacer uso de la plataforma sólo es necesario conectar con una conexión a internet a 256 kbits/s, con una tasa de transferencia por archivos de 320 kbits/s. Estas pistas se reproducen bajo demanda, es decir, como si estuvieran físicas en el almacenamiento interno del dispositivo.

Otra de las características es el uso de la caché del [software](https://bigsoftware.es/que-es-un-software/). De esta manera las canciones se guardan en esta memoria y pueden ser reproducidas nuevamente sin necesidad de descargarlas varias veces y consumir así [ancho de banda.](https://lovtechnology.com/que-es-el-ancho-de-banda/)

En cuanto al acceso a la plataforma, es necesario crear una cuenta para que sea posible utilizar la aplicación. Con esta cuenta el usuario puede acceder desde cualquier dispositivo. No obstante, solo se puede escuchar música con la misma cuenta en un solo dispositivo.

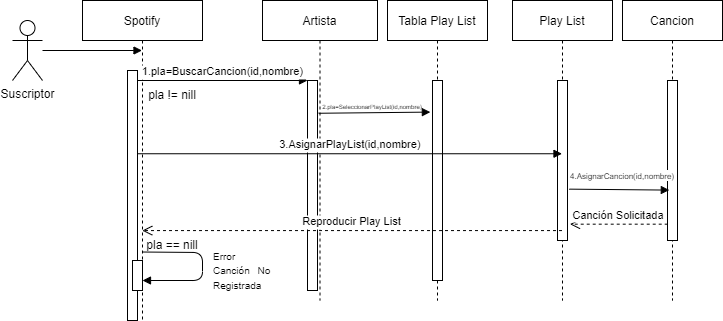
Spotify está disponible en una gran variedad de dispositivos, como ordenadores, teléfonos, tabletas, altavoces, televisores o coches.

**Diagrama Entidad Relación Spotify**

****

**Figura 01- UML Spotify**

**Diagrama Secuencia Spotify**



**Figura 02 - Diagrama Secuencia Spotify**

**Objetivo**

El objetivo general es la puesta en práctica de una metodología de programación implementado en el lenguaje Smalltalk de un programa que simule el funcionamiento de un negocio.

Para el desarrollo del negocio debemos entender el problema y la comunicación con el cliente, definir clases con sus respectivos atributos y métodos que implementan todos los recursos vistos, como select, collect, reject, detect y además hacer uso de colecciones como listas o diccionarios que almacenen información y la muestran de forma ordenada.

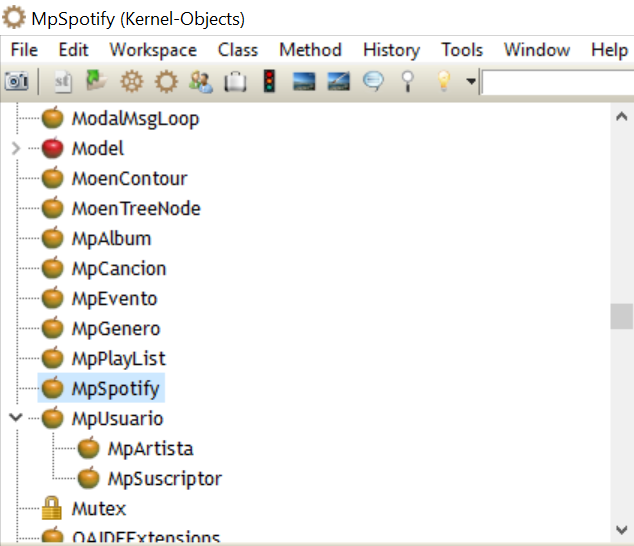
Para el desarrollo del trabajo final integrador se desarrollará la implementación de un negocio que funcione como plataforma de gestión de música Spotify. Este producto desarrollado en Smalltalk contará con las funciones de:

* Registrar, eliminar o modificar Usuarios.
* Registrar y buscar Playlist.
* Registrar y buscar Canciones.

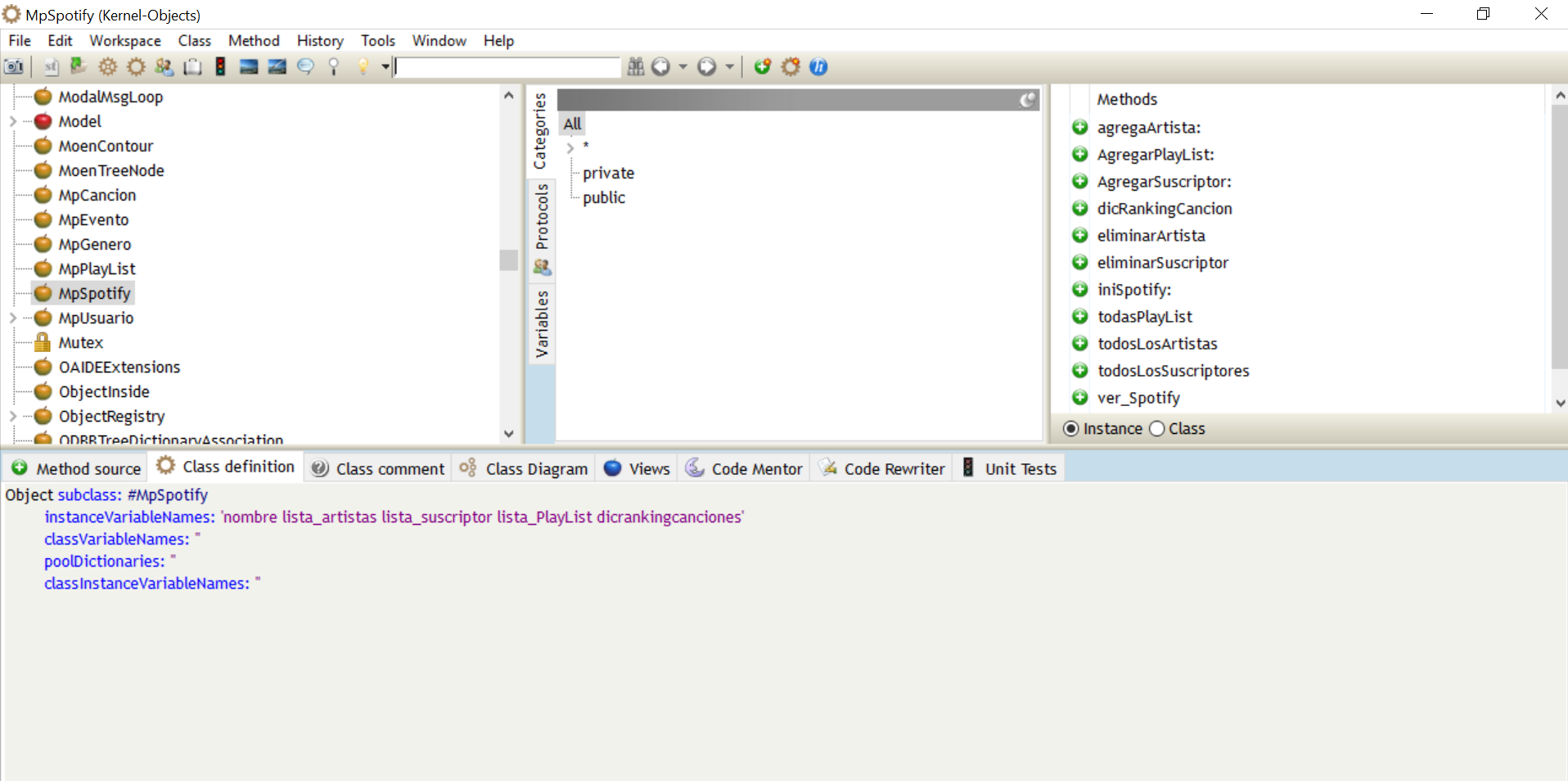
Añadir eventos, géneros musicales, álbumes, entre otras.

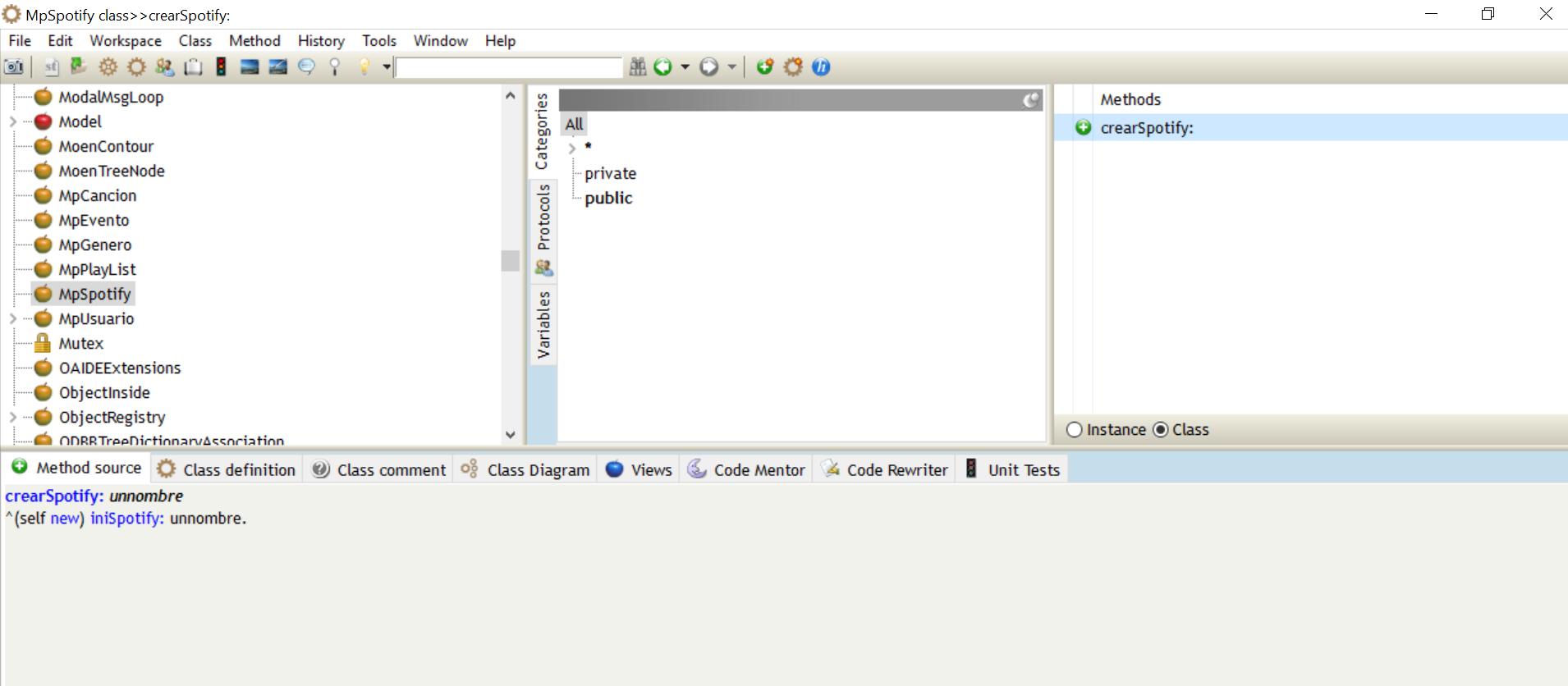
El diseño del negocio se plasmó en el Lenguaje Unificado de Modelado (UML) **Figura 01- UML Spotify ,** se observan todas las clases con sus respectivos atributos y métodos. Serán implementados sólo las clases más importantes para representar el funcionamiento del negocio

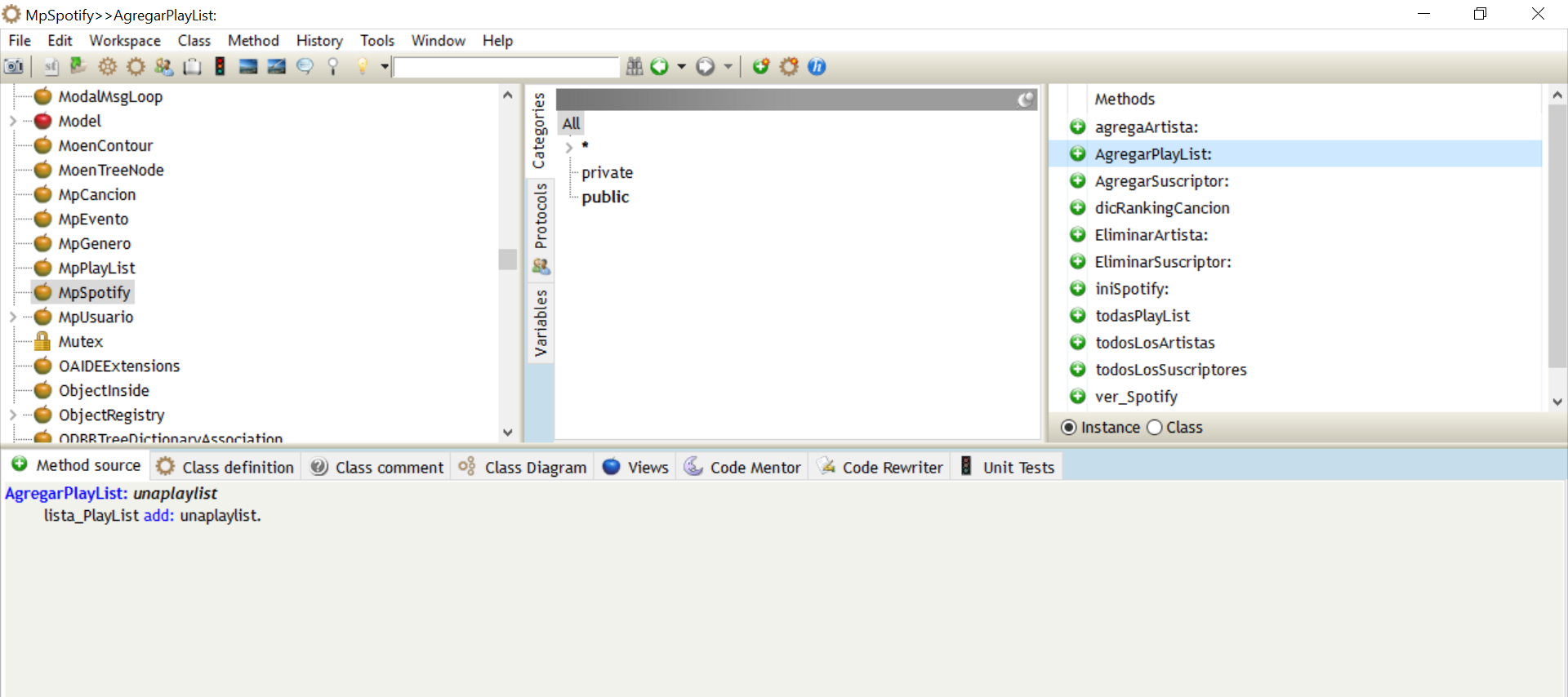
**Especificación e implementación de las clases utilizadas**

****

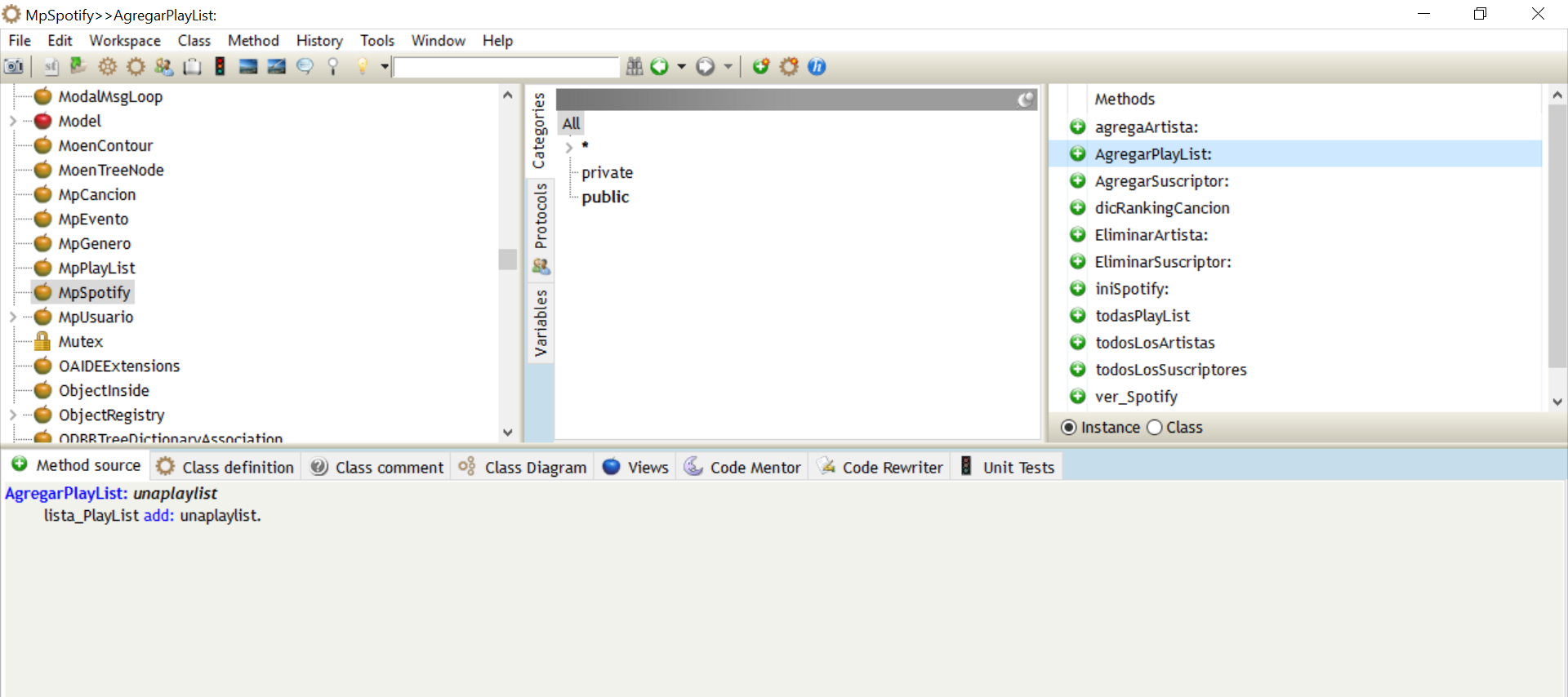
**Figura 03 - Clases negocio Spotify**

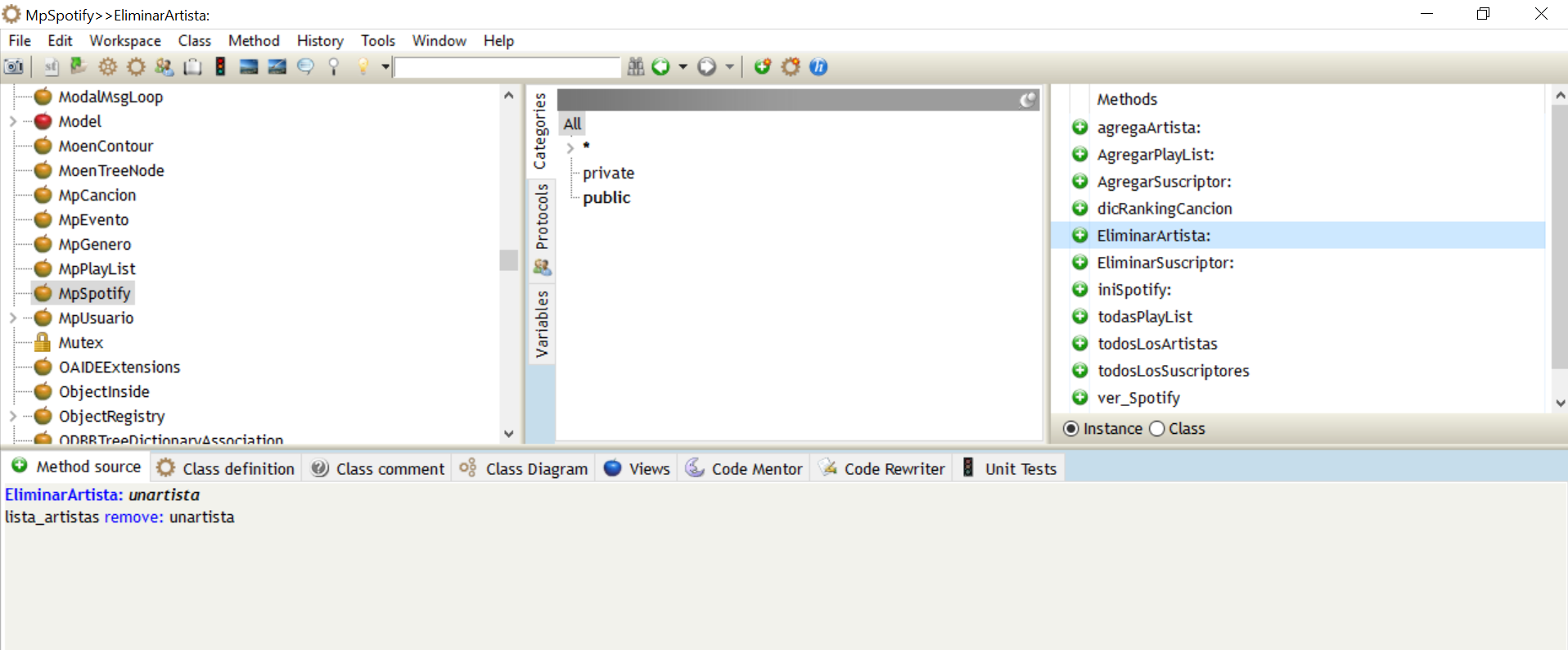
Clase Spotify - Métodos de instancia.

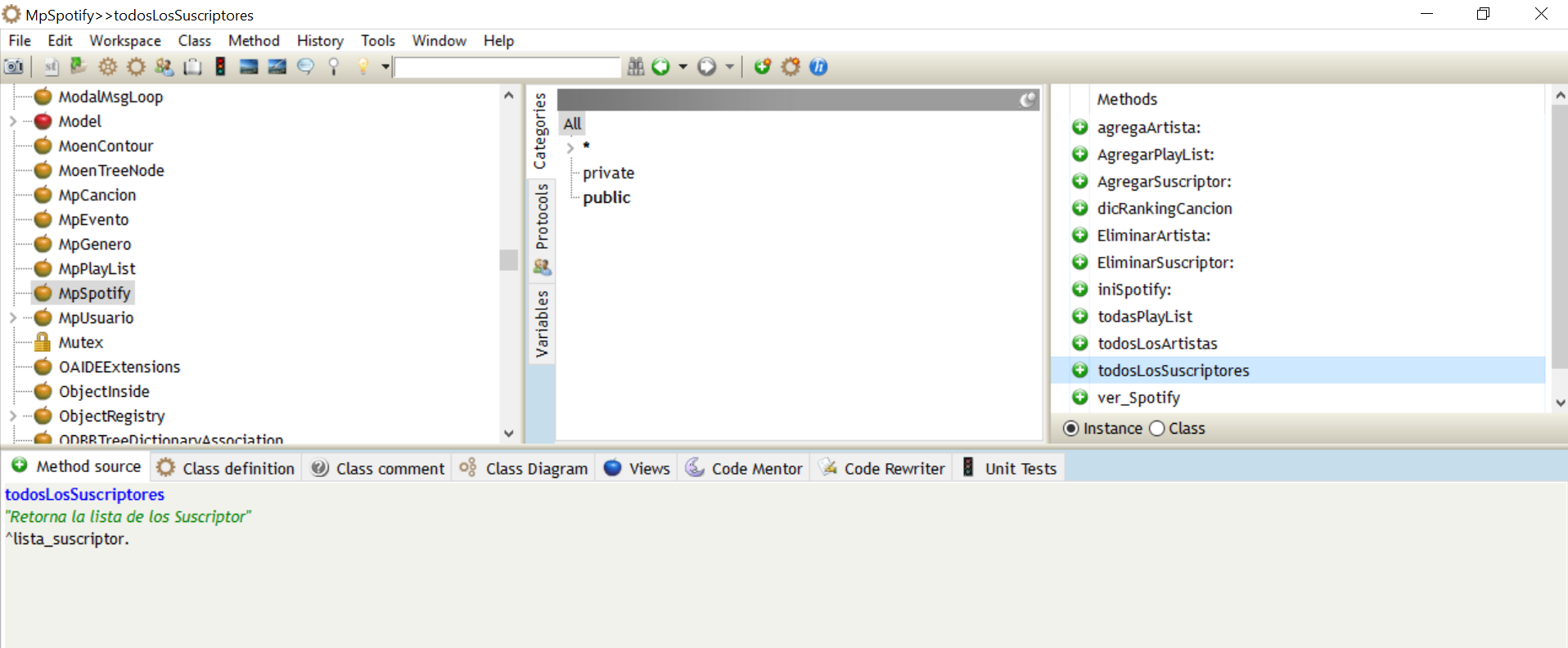
Clase Spotify - Métodos de clase.



Clase Spotify - Métodos Agregar- Eliminar- Buscar. Estos ejemplos son con Suscriptores, Artistas y Playlist se implementan para cada uno.

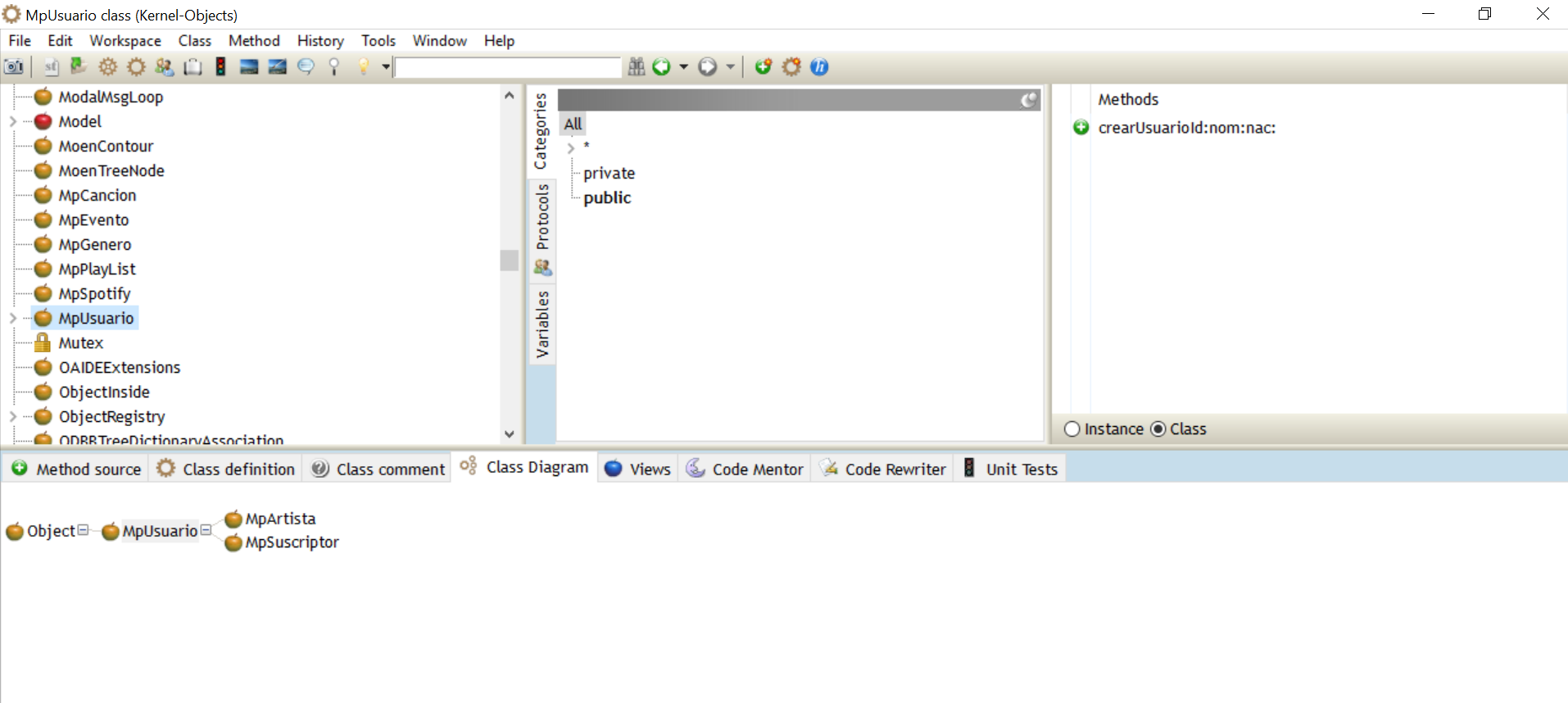




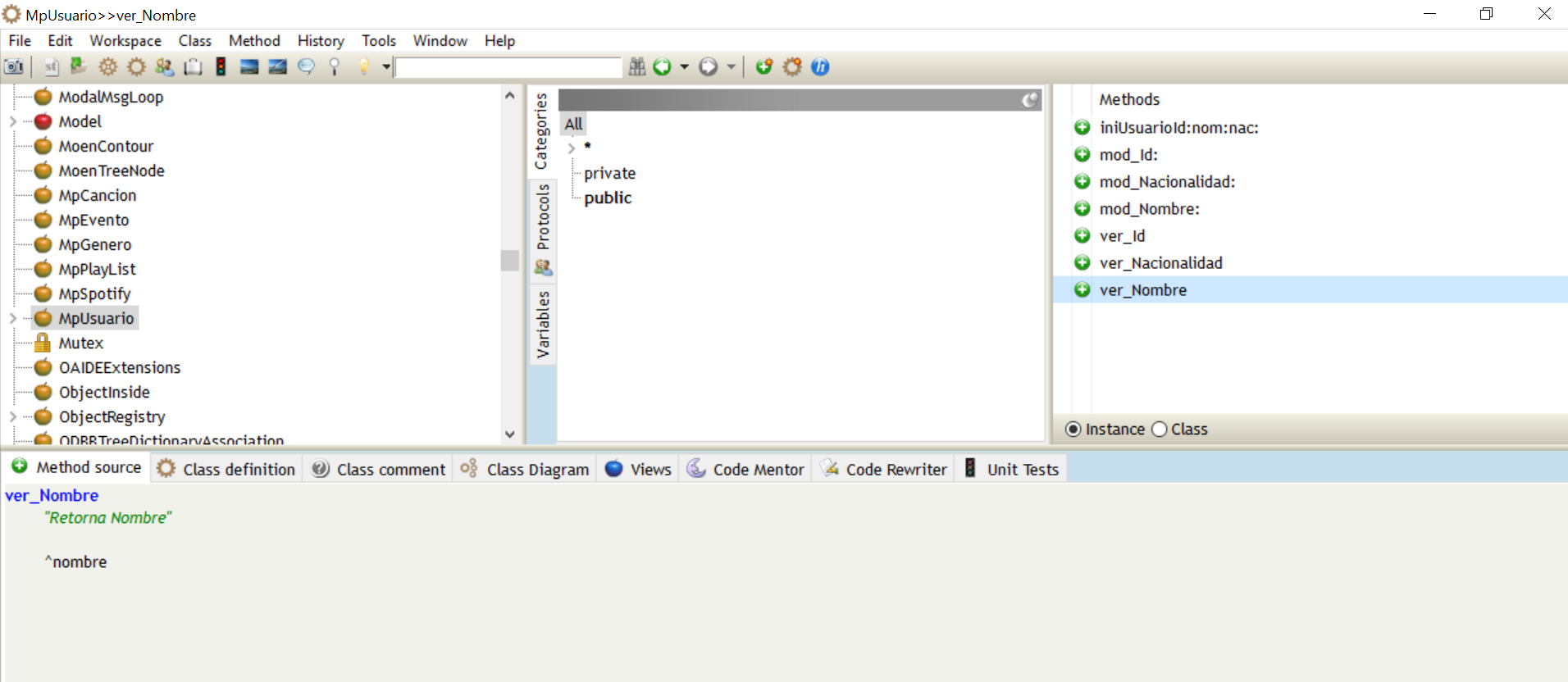


Clase Usuario - Se observa la herencia.

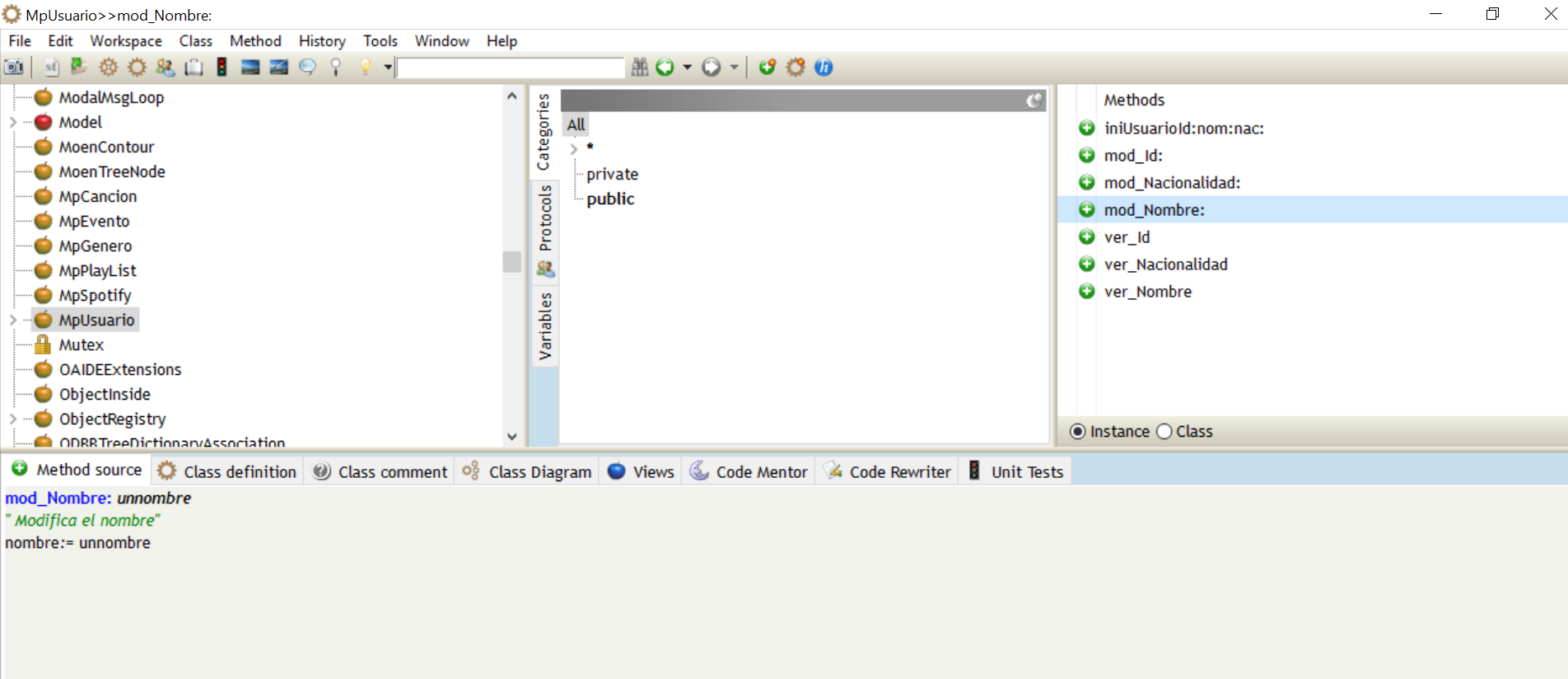
Ambos (suscriptor y artista) son subclases de Usuario, en otros ejemplos se puede encontrar con el nombre Persona. Esta condición se da porque cumplen con la relación “es-un” con las clases superiores en la jerarquía.



Clase Usuario- Método Ver\_ Nombre



Clase Usuario- Método Modificar Nombre.

Los métodos Ver y Modificar se implementan de manera similar en las otras clases del proyecto Spotify.

**La herencia en Smalltalk**

Herencia de estructura: en Smalltalk es total, las subclases heredan todos los atributos de la superclase. No hay forma de no heredar algún atributo.

Herencia de comportamiento: es parcial, se puede no heredar un método de una superclase, redefiniéndolo en la subclase.

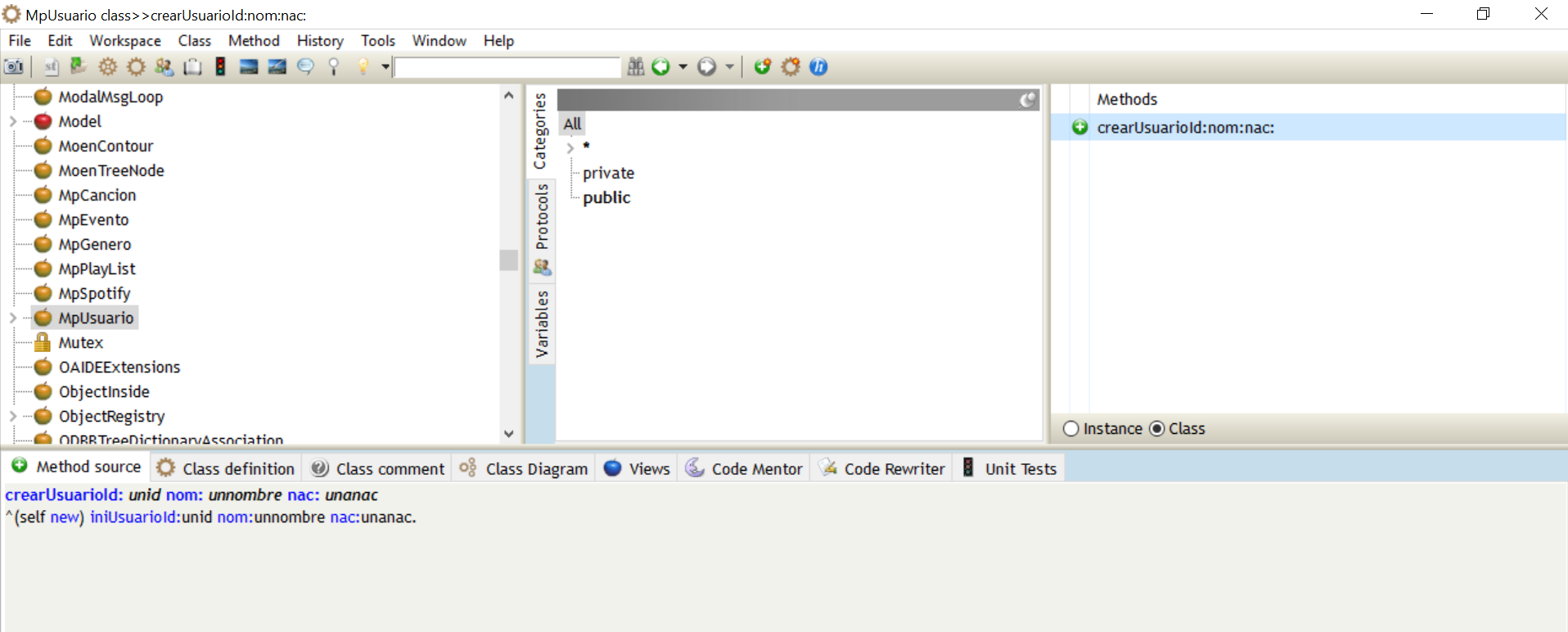
**Self y super en Smalltalk**

self y super son objetos cuyo significado depende del contexto donde se usen.

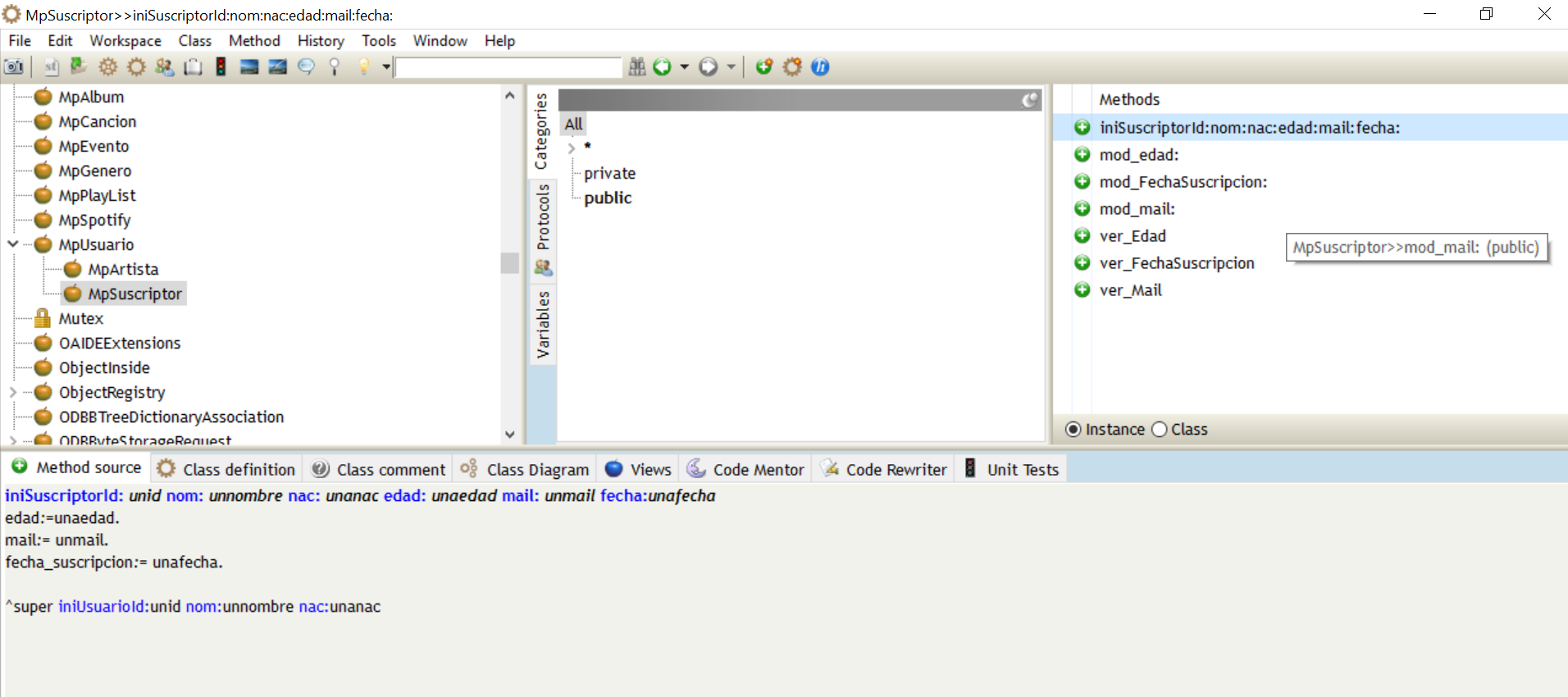
En la programación en Smalltalk de nuestro negocio para una mejor práctica se recomienda utilizar Self, porqué es usada en métodos; su valor es siempre el objeto que recibe el mensaje que causa que el método que contiene self sea ejecutado.

En cambio super es usado en métodos. Su valor es equivalente a self, el objeto que recibe el mensaje que causa que el método que contiene super sea ejecutado, pero la estrategia de búsqueda del método es diferente. La búsqueda comienza en la superclase inmediata de la clase que contiene el método en el cual súper aparece. Los mensajes a super son utilizados cuando se quiere utilizar un método de una superclase que es redefinido en una subclase.

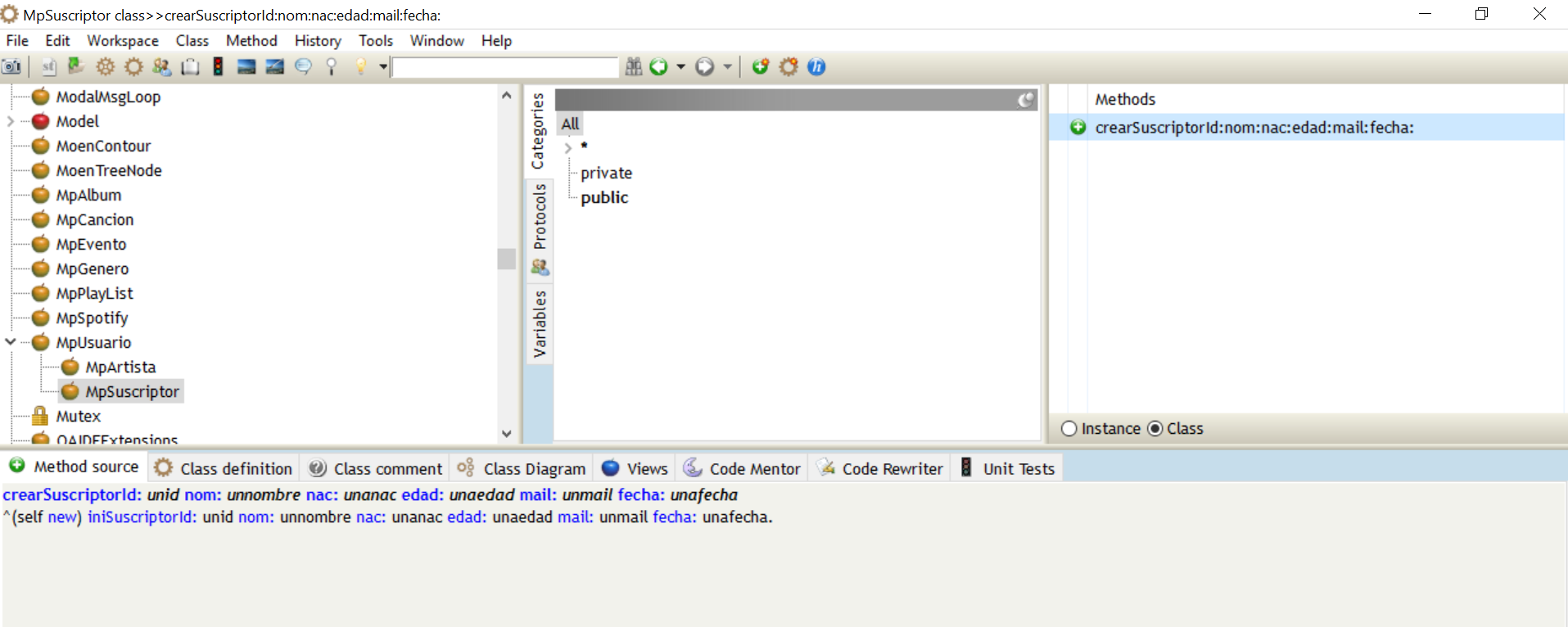
Clase Usuario- Metodo Crear implementando Self



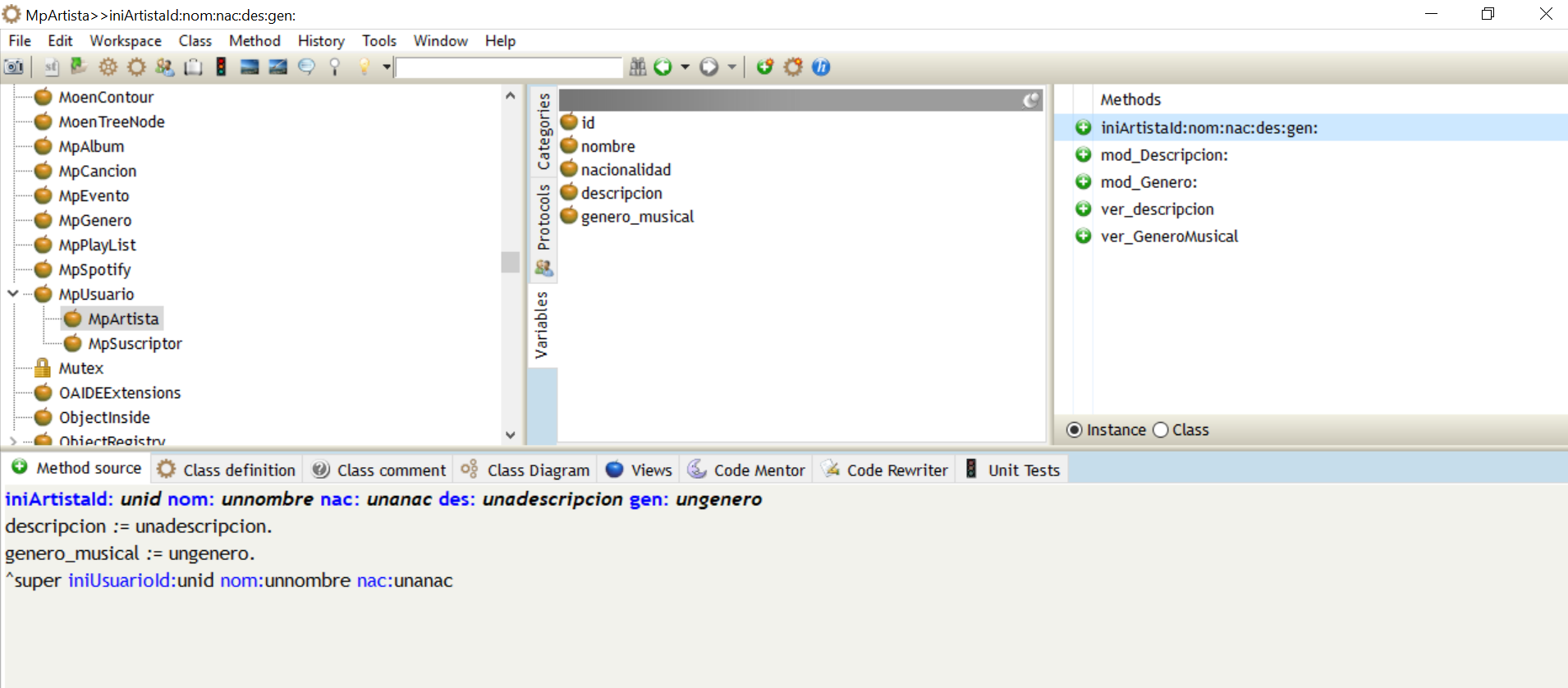
Clase Suscriptor- Método Inicializar implementando Super



Clase Suscriptor- Método Crear implementando Self



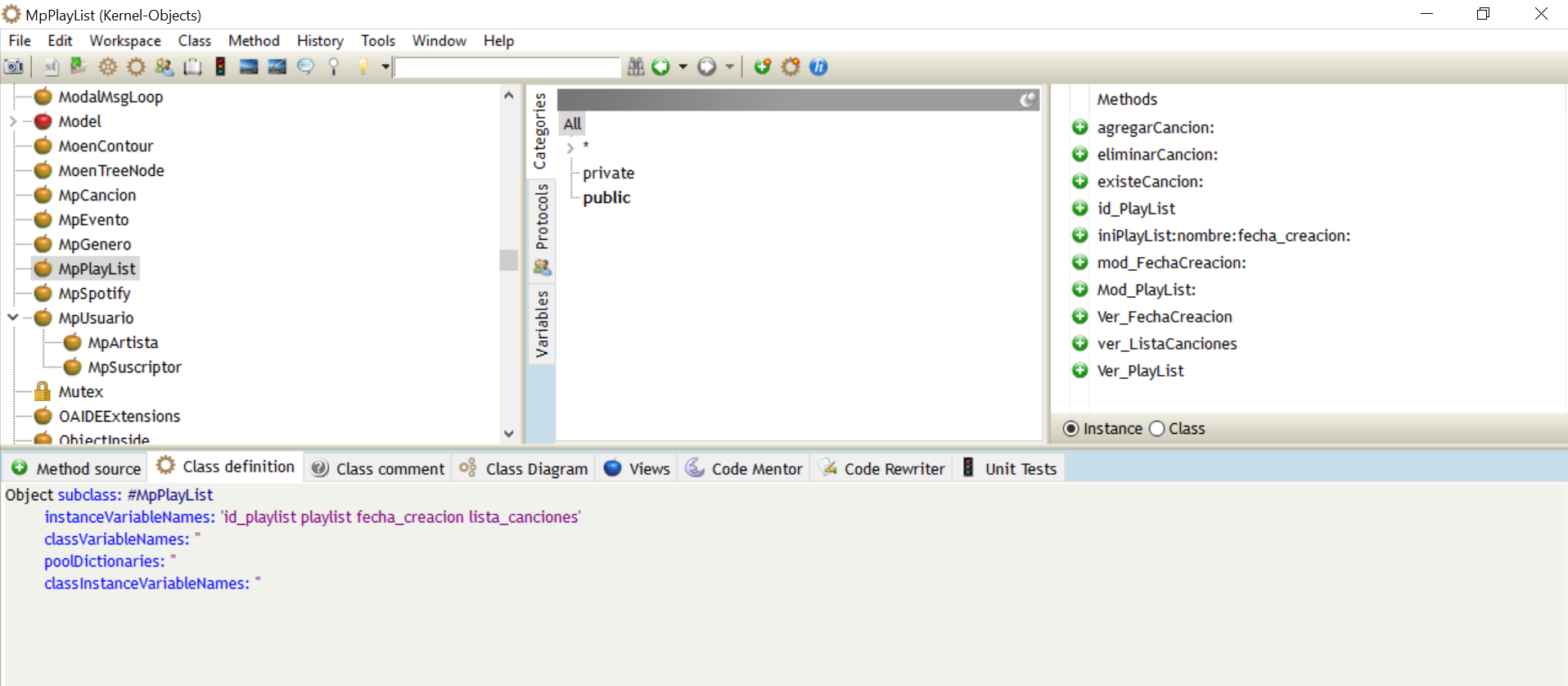
Clase Artista - Método inicializar implementando Super.



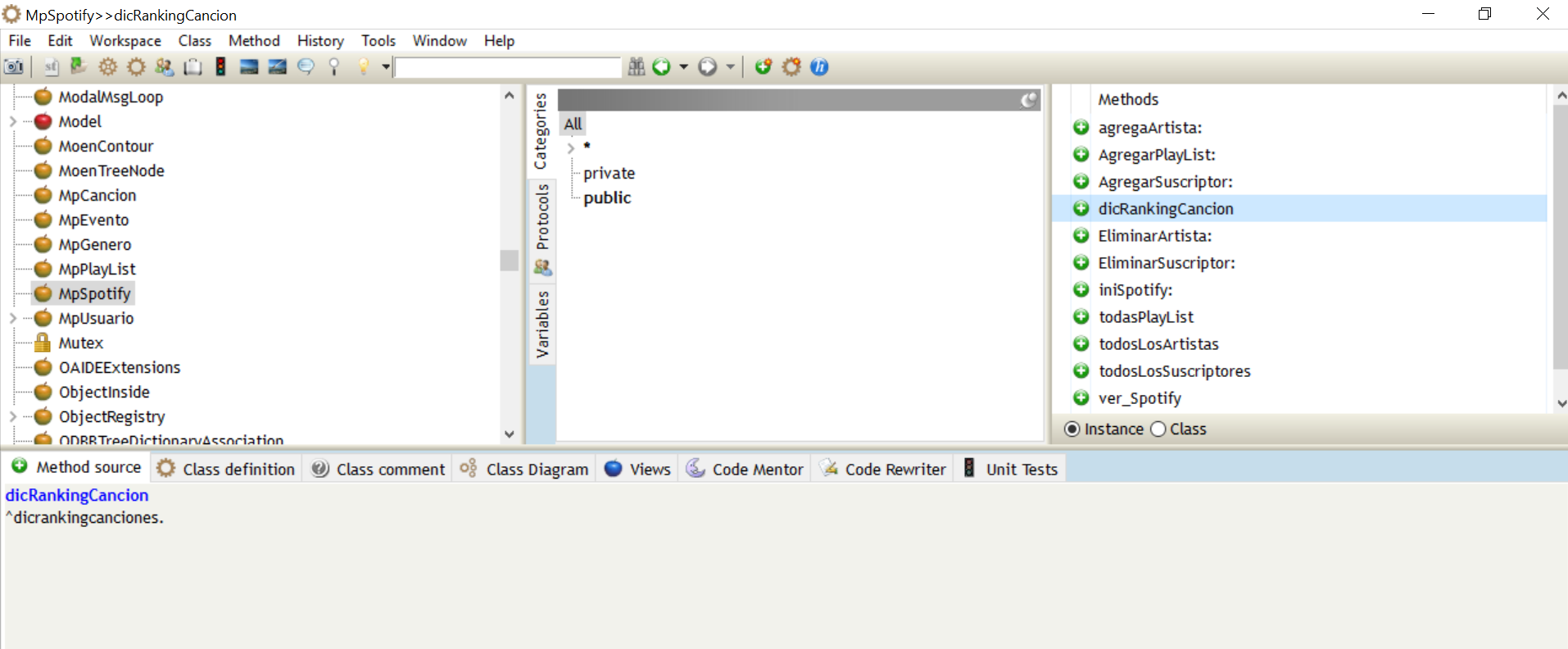
En los métodos crear, como por ejemplo cuando creamos el Suscriptor o Artista se usa ^self, por que se ejecuta el método inicializar Suscriptor o Artista.

En el método inicializar , como por ejemplo cuando inicializamos el Suscriptor o Artista se utiliza ^super por que utilizaremos los métodos de la superclase Usuario.

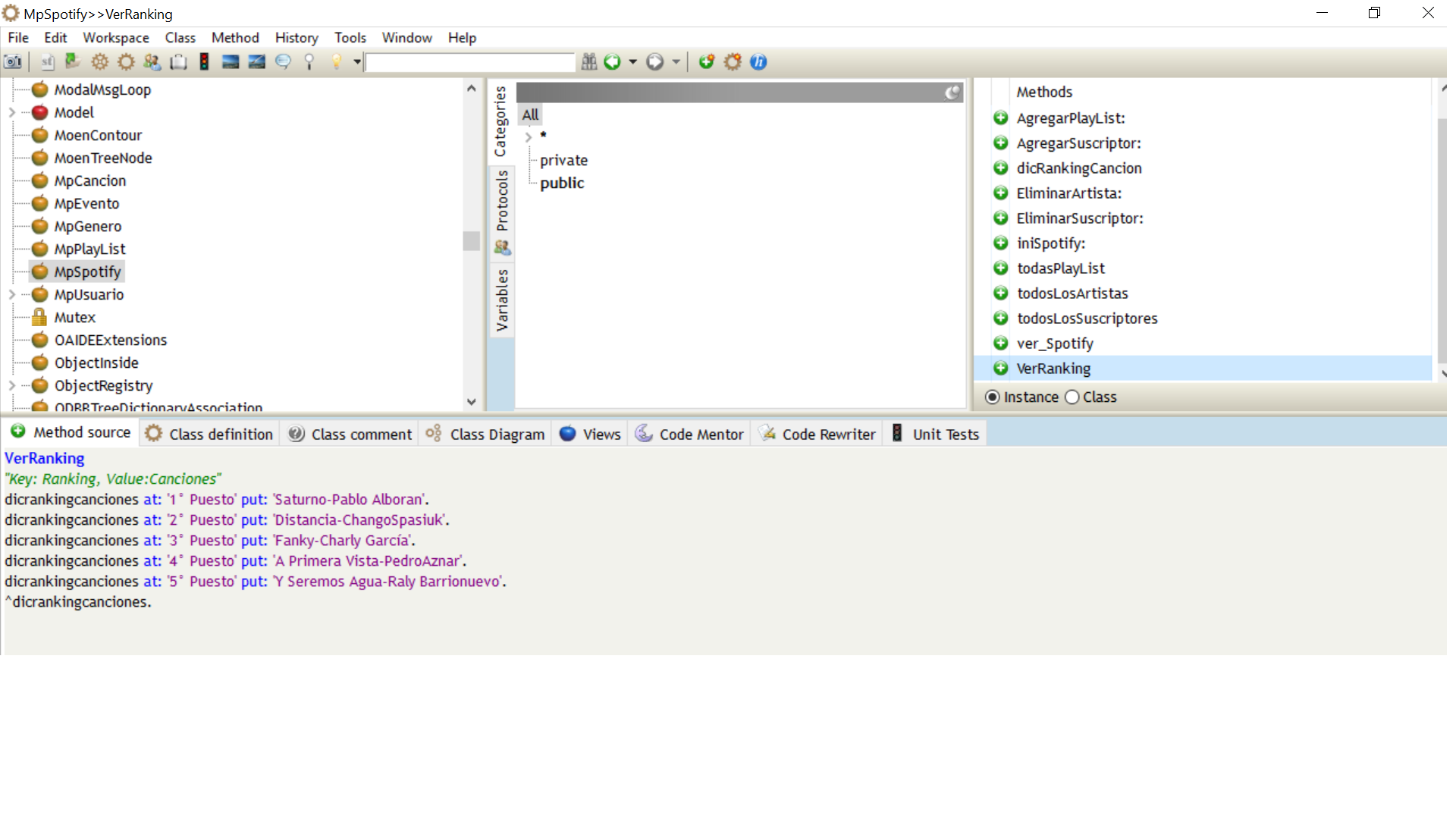
Clase PlayList



Spotify - Diccionario de canciones.

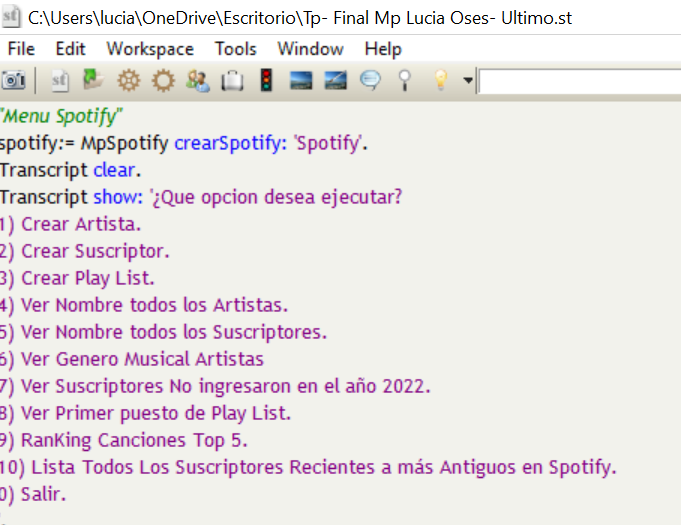


El Diccionario admite el at: put:, con ellos podemos manejar la colección en forma indexada, es decir una asociación está formada por una clave y un valor, una clave no pueden repetirse y un valor puede repetirse.



**La implementación de la aplicación desarrollada.**

Menú de Spotify

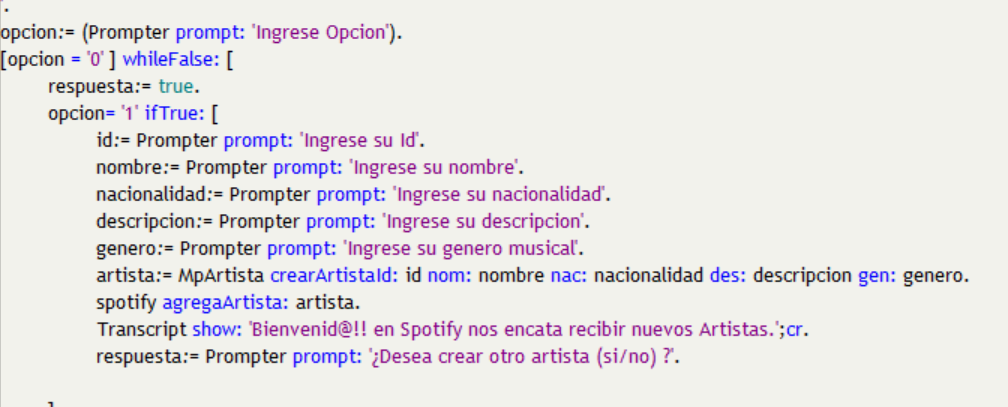
****

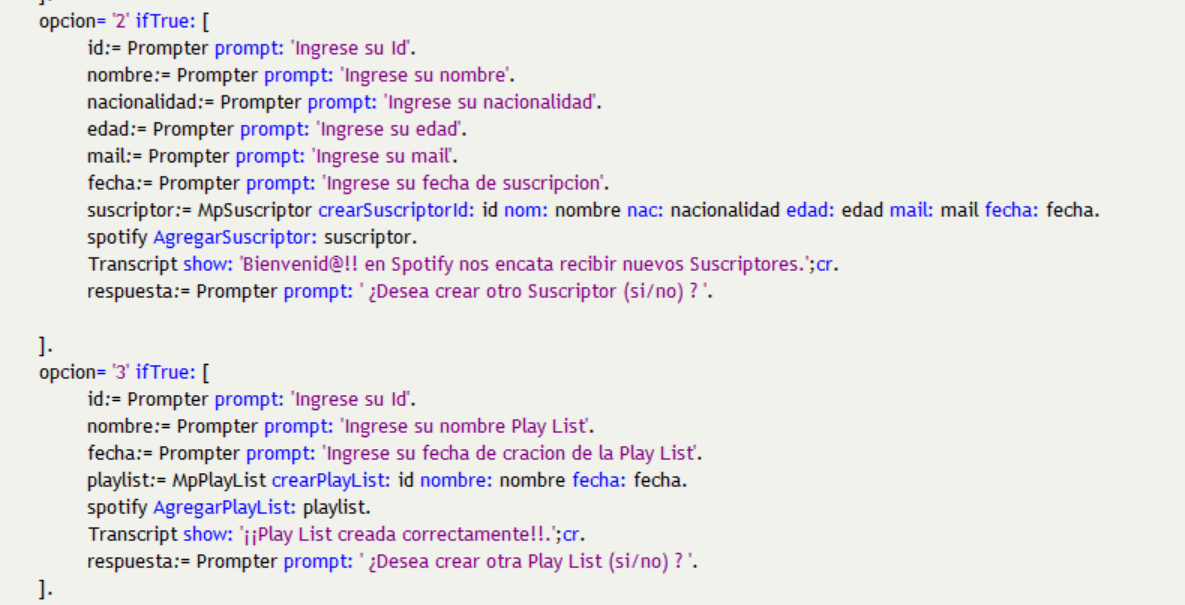
Para desarrollar nuestro menú implementaremos un bucle While que itera o repite un bloque de código mientras la condición sea verdadera, es decir True.

La condición que se evalúa True o False se hace en cada repetición del bucle. La instrucción que representa la línea del código se ejecuta si la condición es True, pero si el resultado es falso las instrucciones no serán ejecutadas.

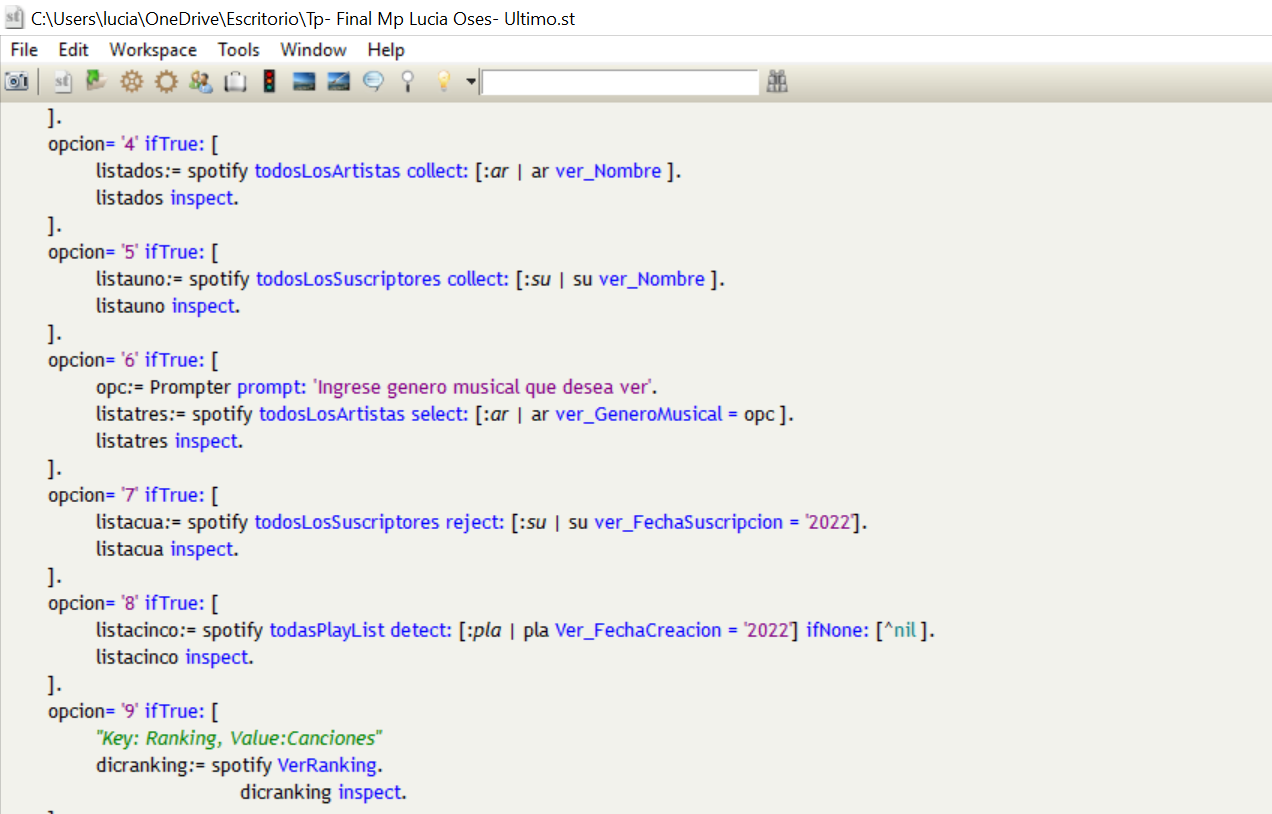
En nuestro menú no se ejecutan las instrucciones cuando se ingresa la opción 0 (cero) y el resultado es falso. Mientras este no sea cero se ejecutarán los diferentes bloques de instrucciones mostrando la funcionalidad de nuestro negocio Spotify.

Crear Artista - Suscriptor y Playlist.

****

****

Implementación Collect - Select - Reject - Detect y Diccionario.



Collect: para cada uno de los elementos del receptor se evalúa el bloque con ese elemento como parámetro, y devuelve una nueva colección formada por el resultado de esas evaluaciones. En nuestro programa devolvemos todos los nombres de los Artistas y luego de los Suscriptores (opción 5).

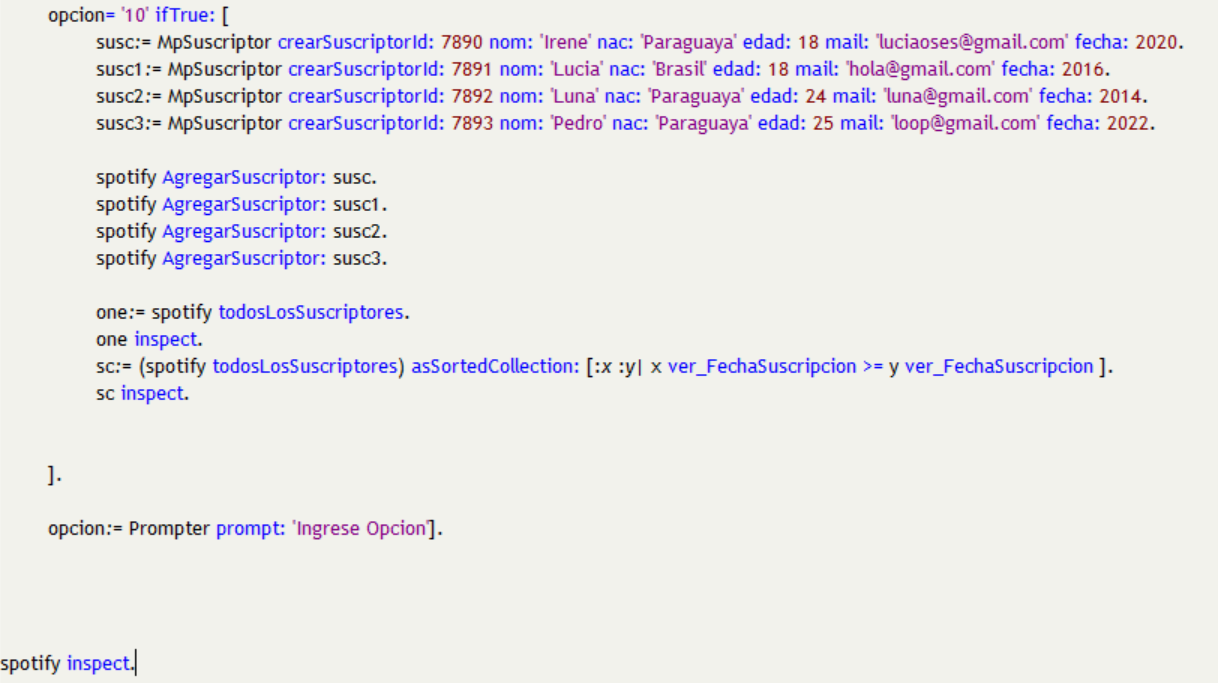
Select: crea y retorna una subcolección de los elementos del receptor. En nuestro programa seleccionamos una lista de artistas pertenecientes al Género Musical que ingresemos por consola.

Reject: crea y retorna una subcolección de los elementos del receptor que no verifiquen la expresión booleana final del bloque. En nuestro programa seleccionamos una lista de suscriptores no pertenecientes al año que pasemos como parámetro, en este ejemplo 2022.

Detect: devuelve el primer elemento de la lista que verifique la expresión booleana final del bloque. En nuestro programa va a devolver el primer puesto de las Play List.

Diccionario: En nuestro programa va a devolver el top cinco de canciones.

Lista de Suscriptores ordenados de mayor a menor por fecha de suscripción a Spotify



En la clase SortedCollection permite ordenar una colección ya cargada bajo otro criterio, es subclase de OrderedCollection. En nuestro programa ordenaremos los suscriptores por fecha de suscripción a Spotify.

**Metodología de trabajo y prácticas ágiles utilizada.**

En nuestro programa Spotify tenemos que atender a las necesidades de rapidez, flexibilidad y variantes externas, donde es de gran importancia aumentar la productividad y satisfacer las necesidades del cliente en el menor tiempo posible para proporcionar mayor valor al negocio.

La implementación de una metodología adecuada, nos da un marco de trabajo seguro para el análisis, diseño, implementación y pruebas del software y sistemas. Además, abarca una amplia colección de métodos y técnicas de gestión de proyectos para asegurarnos un producto de calidad.

**Descripción General**

Spotify es una plataforma de reproducción musical donde debemos desarrollar nuestros servicios, negocio donde esperan obtener grandes beneficios y ganancias.

El modelo de Spotify es un enfoque autónomo basado en las personas para escalar la metodología ágil que destaca la importancia de la cultura y las redes. Ha ayudado a Spotify y a otras organizaciones a aumentar los niveles de innovación y productividad al centrarse en la autonomía, la comunicación, la responsabilidad y la calidad.

7.1 Perspectiva del producto

En el proyecto se exponen los productos en perspectiva con otros productos relacionados. Para ello se realiza el Estudio técnica y operativo.

7.2 Funciones del Producto

En el proyecto se detalla las funciones de cada uno de los productos que se requiere, es decir que métodos implementaremos en nuestra aplicación Smalltalk.

7.3 Caracteristicas de los usuarios

En el proyecto se describen las características generales de los usuarios, este informe se encuentra desarrollado en el detalle de los requerimientos funcionales.

7.4 Restricciones

Detallaremos las problemáticas a las que se enfrenta nuestro producto. Es importante realizar un relevamiento y detallar el funcionamiento de cada componente de nuestro proyecto.

Nos enfrentamos a problemas como:

* Dispositivos telefónicos o computadores que no puedan acceder a nuestro producto.
* Caídas temporales del servicio.
* Aplicación compleja y de difícil acceso.
* Limitación de tiempo

7.5 Suposiciones y Dependencias

Teniendo en cuenta las problemáticas mencionadas en el punto 2.4 Restricciones, el enfoque de las mejoras deben orientarse siempre en las problemáticas actuales que sufre el proyecto, así rediseñar, mejorar y fortalecer el producto final.

7.6 Requisitos Futuros

Los requisitos que se tardaran en nuestros proyecto es el diseño a medida de nuestra página web o la aplicación móvil, estos dependen de la cantidad de usuarios que debemos prestar servicio.

**Metodología de desarrollo ágil Scrum**

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

**Proyecto de negocio Spotify se estructuró mediante los Sprint**

**Visión General del producto.**

El proyecto se va a desarrollar y organizar en etapas (Sprint) donde se realizarán tareas específicas y se evaluará su desempeño.

**Primer Sprint**

En este primer Sprint se verá detallado cada etapa del proyecto, teniendo ciclos o iteraciones de trabajo para llegar a desarrollar nuestro negocio.

El primer Sprint dará comienzo el 3/05 y finalizará el 23/05, donde se realizará el desarrollo del esquema IEEE 830-1998, donde figuran los requerimientos funcionales, el diagrama de Entidad Relación (DER).

**Segundo Sprint**

En este segundo Sprint dará comienzo el 1/06 y finaliza 14/06, donde se realizará el desarrollo de las clases del negocio con sus atributos y métodos en Smalltalk.

**Tercer Sprint**

En el Tercer y último Sprint de este proyecto dará comienzo 15/06 hasta 12/07, donde se realizará la implementación de los métodos definidos en las clases a través de un menú, donde se imprimirá la información, además se simulara una parte del negocio a través de un diagrama de secuencia.

**Proyecto de negocio Spotify - Reuniones**

**Reunión de Planificación del Sprint**

En esta reunión se decide que se va a hacer y cómo se va a llegar a los objetivos, se apuntan las ideas de todos los integrantes, para que las decisiones no sean una imposición “del jefe” y se logre un común acuerdo, también se determinan el número de tareas del primer sprint.

Es imprescindible la asistencia de todo el equipo y su longitud es variable dependiendo el tiempo estimado del proyecto, para este sprint con duración de tres semanas, se estima una reunión de 4 horas.

Al final del encuentro cada miembro del equipo sabe en que trabajara, que se espera que haga y la forma que lo hará posible.

**Reuniones Scrum Diario**

En estas reuniones, de poca duración, se realizan con los miembros del equipo que están llevando adelante las tareas.

El Scrummaster o facilitador es la persona que se encarga de eliminar las dificultades que se interpusieron.

El objetivo de esta reunión es mejorar la comunicación, desempeño, detectar errores y tomar decisiones rápidamente.

**Reuniones de Retrospectiva**

En esta reunión vamos a evaluar cómo ha trabajado el equipo y qué problemas ha tenido durante el desarrollo de las tareas del sprint y como se pueden solucionar.

El equipo Scrum y el cliente colaboran durante la revisión que se hizo durante el primer Sprint. Esta es una reunión informal, donde se ajusta el Product Backlog, si es necesario.

El project owner se encarga de llevar adelante la revisión verificando que los cambios no pongan en peligro el proyecto.

**Sprint Backlog - Primer Sprint**

Requerimientos que el equipo se compromete a realizar durante la duración del primer Sprint.

| **ID** | **Enunciado** | **Estado** | **Sprint** | **Prioridad** |
| --- | --- | --- | --- | --- |
| **1** | Reunion Planificacion Sprint | Iniciado | 0 | Alta |
| **2** | Reuniones Scrum Diario | Iniciado | 0 | Alta |
| **3** | **Esquema IEEE 830-1998** | **Iniciado** | **1** | **Alta** |
| **4** | **Requerimientos Funcionales** | **Iniciado** | **1** | **Alta** |
| **5** | **DER** | **Iniciado** | **1** | **Alta** |
| **6** | Reuniones de Retrospectiva | Iniciado | Al finalizar Primer Sprint. | Alta |

| **Tareas** | **L** | **M** | **M** | **J** | **V** |
| --- | --- | --- | --- | --- | --- |
| **Reunion Planificacion Sprint** | - | - | 2hs | - | - |
| **Reuniones Scrum Diario** | 15min | 15min | 15min | 15min | 15min |
| **Esquema IEEE 830-1998** | 1hs | - | - | - | - |
| **Requerimientos Funcionales** | - | - | - | 2hs | - |
| **DER** | - | - | - | - | 2hs |
| **Reuniones de Retrospectiva** | - | 2hs | - | - |  |

**Sprint Backlog - Segundo Sprint**

Requerimientos que el equipo se compromete a realizar durante la duración del segundo Sprint.

| **ID** | **Enunciado** | **Estado** | **Sprint** | **Prioridad** |
| --- | --- | --- | --- | --- |
| **1** | Reunion Planificacion Sprint | Iniciado | 0 | Alta |
| **2** | Reuniones Scrum Diario | Iniciado | 0 | Alta |
| **3** | **Clase Usuario/Artista/ Suscriptor(métodos y atributos)** | **Iniciado** | **2** | **Alta** |
| **4** | **Clase Canción (métodos y atributos)** | **Iniciado** | **2** | **Alta** |
| **5** | **Clase Playlist (métodos y atributos)** | **Iniciado** | **2** | **Alta** |
| **6** | **Clase Album (métodos y atributos)** | **Iniciado** | **2** | **Alta** |
| **7** | **Clase Evento (métodos y atributos)** | **Iniciado** | **2** | **Alta** |
| **8** | **Clase Género (métodos y atributos)** | **Iniciado** | **2** | **Alta** |
| **9** | **Clase Spotify (métodos y atributos)** | **Iniciado** | **2** | **Alta** |
| **10** | Reuniones de Retrospectiva | Iniciado | Al finalizar Segundo Sprint. | Alta |

| **Tareas** | **L** | **M** | **M** | **J** | **V** |
| --- | --- | --- | --- | --- | --- |
| **Reunion Planificacion Sprint** | 2hs | - | - | - | - |
| **Reuniones Scrum Diario** | 15min | 15min | 15min | 15min | 15min |
| **Clase Usuario/Artista/ Suscriptor(métodos y atributos)** | 30min | - | - | - | - |
| **Clase Canción (métodos y atributos)** | 30min | - | - | - | - |
| **Clase Playlist (métodos y atributos)** | - | 30min | - | - | - |
| **Clase Album (métodos y atributos)** | - | 30min | - | - | - |
| **Clase Evento (métodos y atributos)** | - | - | 30min | - | - |
| **Clase Género (métodos y atributos)** | - | - | 30min | - | - |
| **Clase Spotify (métodos y atributos)** | - | - | 30min | - | - |
| **Reuniones de Retrospectiva** | - | 2hs | - | - | - |

**Sprint Backlog - Tercer Sprint**

Requerimientos que el equipo se compromete a realizar durante la duración del tercer Sprint.

| **ID** | **Enunciado** | **Estado** | **Sprint** | **Prioridad** |
| --- | --- | --- | --- | --- |
| **1** | Reunion Planificacion Sprint | Iniciado | 0 | Alta |
| **2** | Reuniones Scrum Diario | Iniciado | 0 | Alta |
| **3** | **Menú** | **Iniciado** | **3** | **Alta** |
| **4** | **Agregar Artistas**  **suscriptores**  **Playlist.** | **Iniciado** | **3** | **Alta** |
| **5** | **Implementar**  **Select**  **Collect**  **Reject**  **Detect** | **Iniciado** | **3** | **Alta** |
| **6** | **Implementer**  **Diccionario** | **Iniciado** | **3** | **Alta** |
| **7** | **Implementar Lista Ordenada** | **Iniciado** | **3** | **Alta** |
| **8** | **Diagrama Secuencia** | **Iniciado** | **3** | **Alta** |
| **9** | Reuniones de Retrospectiva | Iniciado | Al finalizar Tercer Sprint. | Alta |

| **Tareas** | **L** | **M** | **M** | **J** | **V** |
| --- | --- | --- | --- | --- | --- |
| **Reunion Planificacion Sprint** | 2hs | - | - | - | - |
| **Reuniones Scrum Diario** | 15min | 15min | 15min | 15min | 15min |
| **Menú** | 2hs | - | - | - | - |
| **Agregar Artistas**  **suscriptores**  **Playlist.** | - | - | - | 2hs | 2hs |
| **Implementar**  **Select**  **Collect**  **Reject**  **Detect** | 3hs | - | - | - | - |
| **Implementer**  **Diccionario** | - | - | - | - | 3hs |
| **Implementar Lista Ordenada** | 2hs | - | - | - | - |
| **Diagrama Secuencia** | - | - | - | - | 3hs |
| **Reuniones de Retrospectiva** | - | 2hs | - | - | - |

**Product Backlog - Primer Sprint**

Descripciones genéricas de los requerimientos. Ninguna tarea debe superar las 16hs, no son asignadas se reparte en el equipo según corresponda.

| **Backlog Items** | **Estimación (hs)** |
| --- | --- |
| **Esquema IEEE 830-1998** | 1hs |
| **Requerimientos Funcionales** | 1hs |
| **DER - herencia de clases, entidades y relaciones.** | 1hs |

**Product Backlog - Segundo Sprint**

Descripciones genéricas de los requerimientos. Ninguna tarea debe superar las 16hs, no son asignadas se reparte en el equipo según corresponda.

| **Backlog Items** | **Estimación (hs)** |
| --- | --- |
| **Clase Usuario/Artista/ Suscriptor**  **(métodos ver - modificar- listas - inicialización y creación)** | 1hs |
| **Clase Canción**  **(métodos ver - modificar- listas - inicialización y creación)** | 1hs |
| **Clase Playlist**  **(métodos ver - modificar- listas - inicialización y creación)** | 1hs |
| **Clase Álbum**  **(métodos ver - modificar- inicialización y creación)** | 1hs |
| **Clase Evento**  **(métodos ver - modificar- inicialización y creación)** | 1hs |
| **Clase Género**  **(métodos ver - modificar- inicialización y creación)** | 1hs |
| **Clase Spotify**  **(métodos ver - modificar- listas - inicialización y creación)** | 1hs |

**Product Backlog - Tercer Sprint**

Descripciones genéricas de los requerimientos. Ninguna tarea debe superar las 16hs, no son asignadas se reparte en el equipo según corresponda.

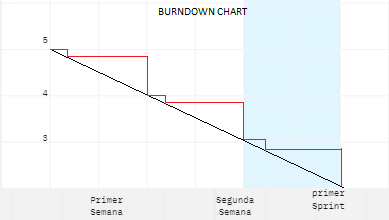
| **Backlog Items** | **Estimación (hs)** |
| --- | --- |
| **Crear Artista.** | 1hs |
| **Crear Suscriptor.** | 1hs |
| **Crear Playlist.** | 1hs |
| **Ver Nombre de todos los Artistas.** | 1hs |
| **Ver Nombre de todos los Suscriptores.** | 1hs |
| **Ver Genero Musical Artistas** | 1hs |
| **Ver Suscriptores No ingresaron en el año 2022.** | 1hs |
| **Ver Primer puesto de Play List.** | 1hs |
| **RanKing Canciones Top 5.** | 1hs |
| **Lista Todos Los Suscriptores Recientes a más Antiguos en Spotify.** | 1hs |
| **Diagrama Secuencia - Buscar Cancion** | 1hs |

**Burndown Chart - Primer Sprint**

ID tareas por Semanas.

Registro en negro semanal es el Progreso esperado de los requerimientos.

Registro en rojo semanal es el Progreso real de los requerimientos.

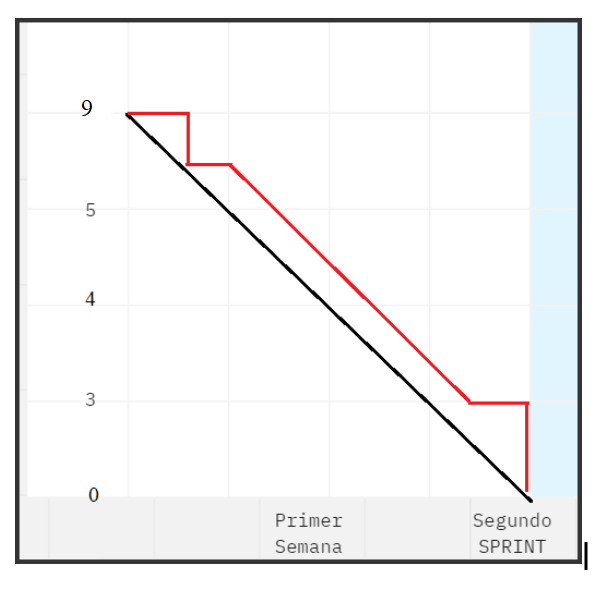
****

**Burndown Chart - Segundo Sprint**

ID tareas por Semanas.

Registro en negro semanal es el Progreso esperado de los requerimientos.

Registro en rojo semanal es el Progreso real de los requerimientos.

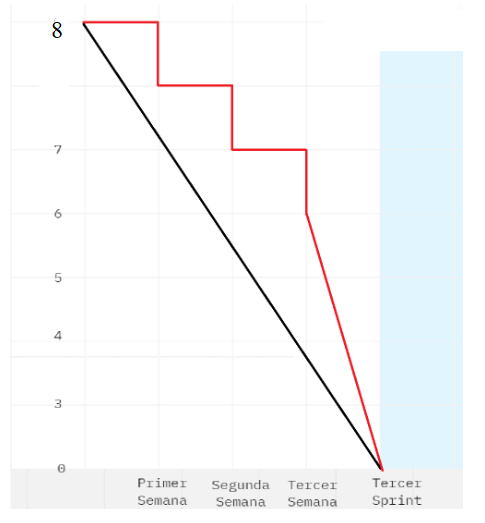


**Burndown Chart - Tercer Sprint**

ID tareas por Semanas.

Registro en negro semanal es el Progreso esperado de los requerimientos.

Registro en rojo semanal es el Progreso real de los requerimientos.



**Proyecto de negocio Spotify - Requerimientos Funcionales**

**Artista**

| ID | RF01 |
| --- | --- |
| Nombre | Registrar Artista |
| Característica | El administrador podrá registrar los datos del artista en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF02 |
| Nombre | Consultar Artista |
| Característica | El administrador podrá ingresar al sistema y consultar los datos del Artista. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF03 |
| Nombre | Modificar Artista |
| Característica | El administrador podrá ingresar al sistema y modificar los datos del Cliente. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF04 |
| Nombre | Listar Artista |
| Característica | El administrador podrá listar los artistas ingresados en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF05 |
| Nombre | Dar de Baja Artista |
| Característica | El administrador podrá eliminar artistas del sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |

**Suscriptor**

| ID | RF01 |
| --- | --- |
| Nombre | Registrar Suscriptor |
| Característica | El administrador podrá registrar los datos del Suscriptor en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF02 |
| Nombre | Consultar Suscriptor |
| Característica | El administrador podrá ingresar al sistema y consultar los datos del Suscriptor. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF03 |
| Nombre | Modificar Suscriptor |
| Característica | El administrador podrá ingresar al sistema y modificar los datos del Suscriptor. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF04 |
| Nombre | Listar Suscriptor |
| Característica | El administrador podrá listar los suscriptores ingresados en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF05 |
| Nombre | Dar de Baja suscriptor |
| Característica | El administrador podrá eliminar suscriptores del sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |

**PlayList**

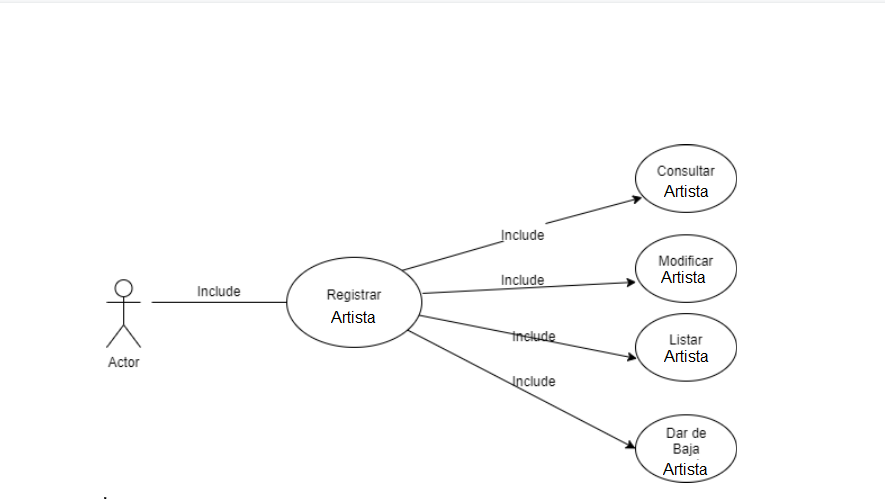
| ID | RF01 |
| --- | --- |
| Nombre | Registrar Playlist |
| Característica | El administrador podrá registrar los datos en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF02 |
| Nombre | Consultar Playlist |
| Característica | El administrador podrá ingresar al sistema y consultar los datos de la play list. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF03 |
| Nombre | Modificar Playlist |
| Característica | El administrador podrá ingresar al sistema y modificar los datos. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF04 |
| Nombre | Listar playlist. |
| Característica | El administrador podrá listar las play list ingresados en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF05 |
| Nombre | Dar de baja la playlist. |
| Característica | El administrador podrá eliminar la playlist. del sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |

**Canción**

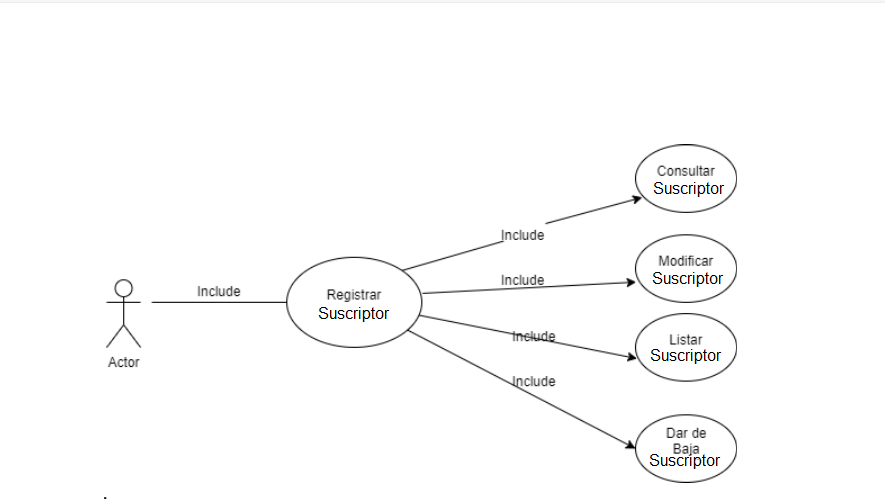
| ID | RF01 |
| --- | --- |
| Nombre | Registrar canción. |
| Característica | El administrador podrá registrar los datos en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF02 |
| Nombre | Consultar canción |
| Característica | El administrador podrá ingresar al sistema y consultar los datos de la canción. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF03 |
| Nombre | Modificar canción |
| Característica | El administrador podrá ingresar al sistema y modificar los datos. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF04 |
| Nombre | Listar canción |
| Característica | El administrador podrá listar las canciónes ingresados en el sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |
| ID | RF05 |
| Nombre | Dar de baja la canción. |
| Característica | El administrador podrá eliminar la canción. del sistema. |
| Detalle | Muestra en pantalla descripción. |
| Prioridad | Alta |

**Proyecto de negocio Spotify - Casos de Uso**

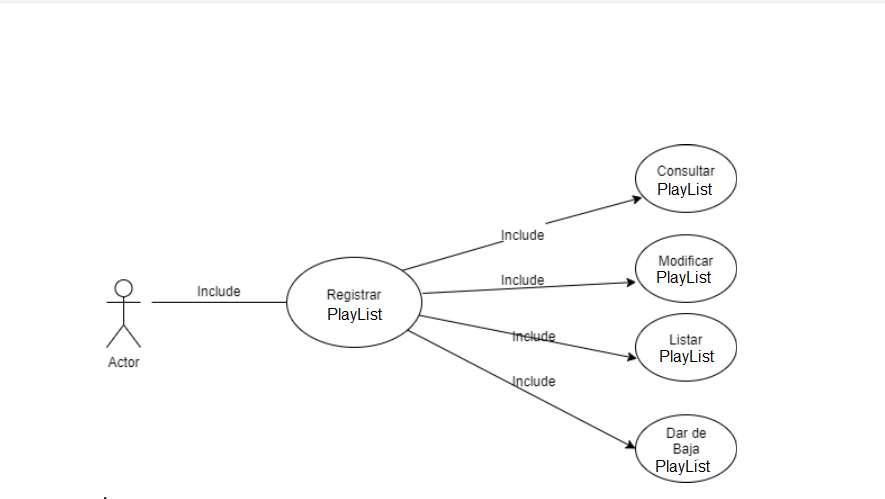
**Artista**

****

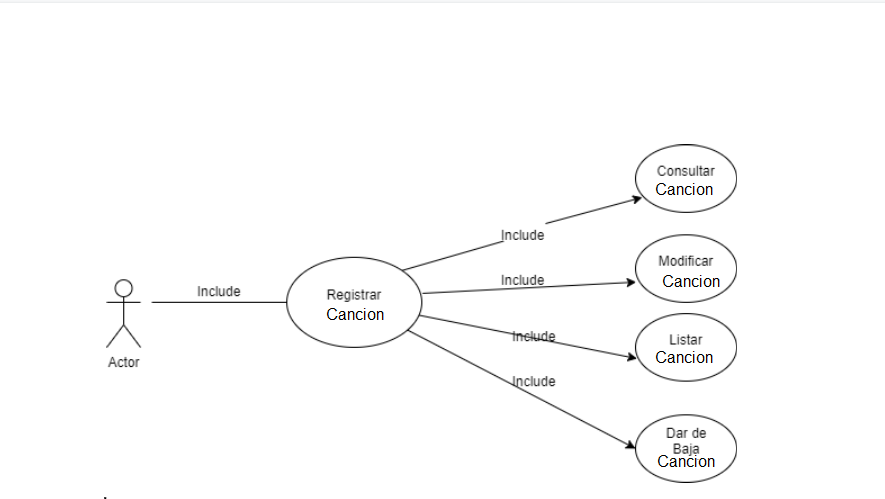
**Suscriptor**

****

**PlayList**

****

**Canciones**

****

**Frameworks que se podrían utilizar para realizar un desarrollo orientado a objetos.**

Los Sprint nos permite desarrollar y evolucionar de forma consciente los requisitos del negocio, describe las distintas fases que componen el ciclo de vida. El objetivo es obtener la mejor calidad posible, las características de un sistema software que satisfaga las necesidades de negocio de clientes y usuarios y que se integre con éxito en el entorno en el que se explote.

Net: Esta plataforma fue construida aplicando el paradigma de Programación Orientada a Objetos (POO). El núcleo de lenguajes como C# están basados en los principios OO. Recurso **C Sharp** ( C#) La programación orientada a objetos es una rama de la informática que usa como su propio nombre indica los objetos y las interacciones de estos para diseñar aplicaciones y programas informáticos. C Sharp presenta los atributos positivos de C++, Java y Visual Basic y los mejora otorgando un lenguaje fuerte y actualizado.

Ruby: Es un lenguaje de programación de propósito general, es decir, con Ruby se pueden desarrollar todo tipo de aplicaciones diferentes: aplicaciones de servicio web, clientes de correo electrónico, procesamiento de datos en Backend, aplicaciones de red, etc. Es 100% orientado a objetos.

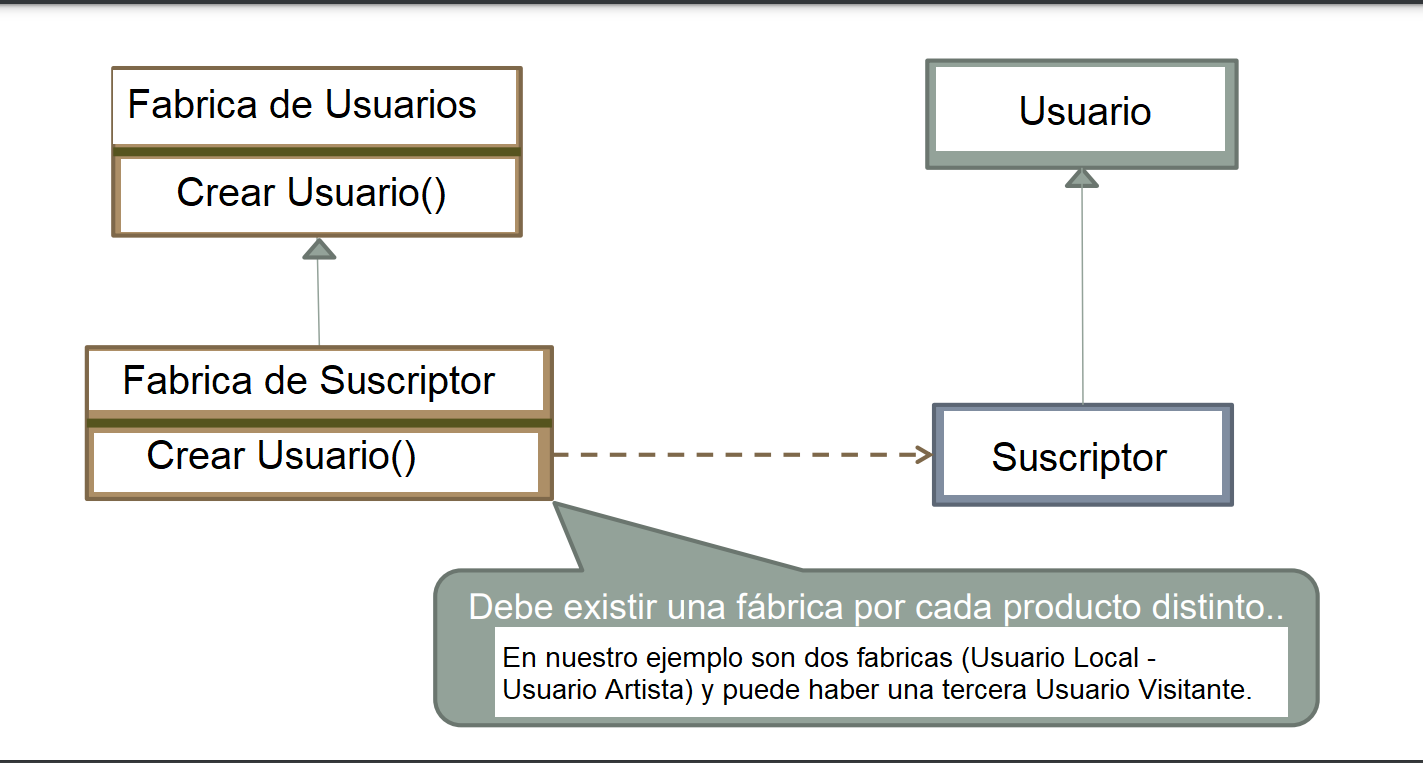
**Uso de Patrones en el contexto del problema planteado**

**Patrón creacional Factory Method**

Propósito: define una interfaz para crear un objeto, pero dejan que sean las subclases quienes decidan qué clase de objetos instanciar.

En Spotify tenemos distintos usuarios pueden ser oyentes suscriptores locales o suscriptores artistas, también podríamos tener oyentes suscriptores visitantes, es decir que no crean una cuenta o solo utilizan un porcentaje de los beneficios de nuestro negocio.

Factory Method nos permite un mecanismo donde se encapsula la creación de los objetos, de modo tal que el cliente solo pida la creación de ellos, sin especificar su clase.



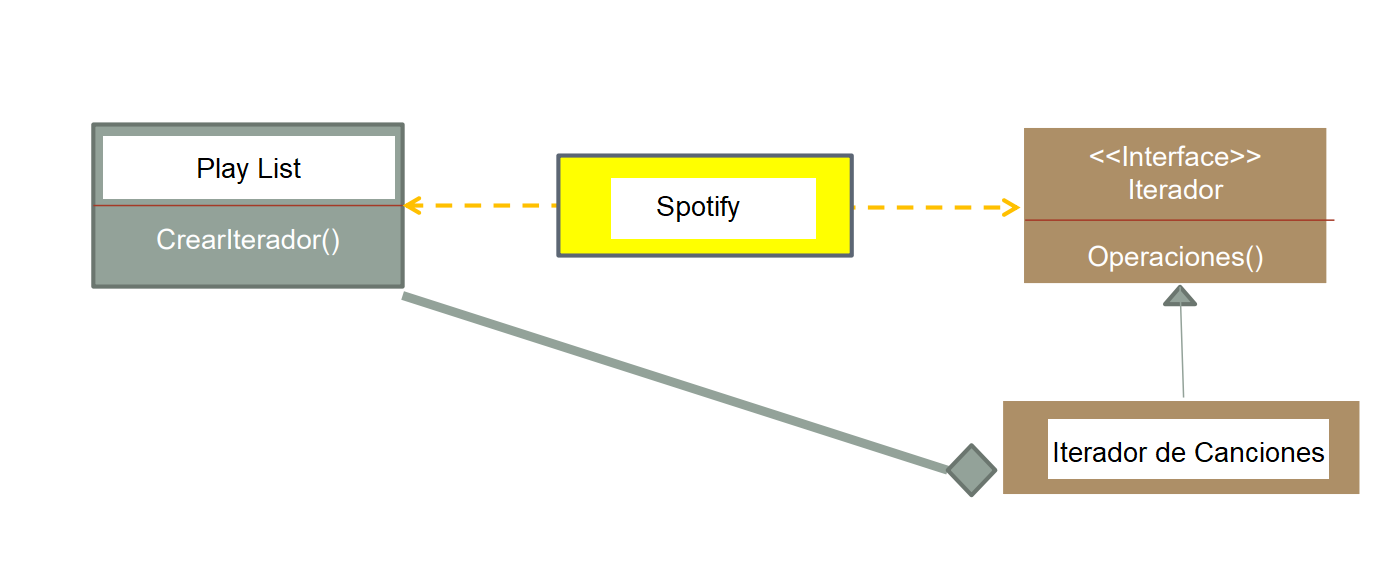
Las ventajas de usar este patrón son:

* elimina la necesidad de ligar clases específicas de la aplicación a nuestro código.
* El código solo trata con la interfaz del Producto.
* Crear objetos dentro de una clase con un método de fabricación es siempre más flexible que hacerlo directamente.

**Patrón de comportamiento Iterator**

Propósito: Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.

En Spotify buscamos un diseño que permita "recorrer" todas las canciones de la Playlist (secuencial, aleatorio, cada dos, etc) independientemente de cómo estén almacenadas.



Las ventajas de usar este patrón son:

* Permite variaciones en el recorrido de un agregado. Los agregados complejos se pueden recorrer de diferentes maneras.
* Los iteradores simplifican la interfaz del agregado. La interfaz de recorrido elimina la necesidad de tener una interfaz similar en el agregado.
* Es posible hacer más de un recorrido a la vez. Como el estado del recorrido lo tiene el iterador, es posible tener más de un iterador sobre la misma colección al mismo tiempo.

**Repositorio se podrían utilizar para el desarrollo de software colaborativo**

**Modelo de Repositorio**

El repositorio que utilizaremos es GitHub para que nuestro negocio funcione de la forma más óptima posible.

Los beneficios de utilizar esta red son:

* Seguimiento de errores. Estructura en árbol esta característica hace posible que los desarrolladores puedan trabajar en diferentes ramas de un proyecto, pero sin modificar en el código base principal, facilitando así que puedan probar nuevas funcionalidades sin miedo a cometer equivocaciones, ya que siempre pueden dar marcha atrás y volver a versiones anteriores.
* Búsqueda rápida. La velocidad es otro de los puntos fuertes de Git frente a otros sistemas controladores de versiones, ya que necesita menos capacidad de procesamiento y gestión al poder realizar las operaciones en local.
* Cuenta con una potente comunidad de desarrolladores en todo el mundo. Trabajo en equipo, permite que varios desarrolladores trabajen al mismo tiempo y en paralelo en un proyecto con un acceso compartido, pero sin estar físicamente cerca, así como identificar qué usuario y cuándo ha realizado cada modificación.
* Permite descargar como archivo el código fuente. Mayor autonomía, cada desarrollador cuenta con una copia local de todo el proyecto y de los cambios generados, lo que le permite trabajar de forma individual y a su propio ritmo, en cualquier momento y lugar.
* Posibilita la importación en Git, SVN o TFS. Sin conexión a la red, el desarrollador puede trabajar o enviar cambios de código al sistema sin necesidad de estar conectado a Internet o a cualquier otra red, al disponer de un repositorio local.
* Puedes personalizar cualquier servicio host en la nube.

Los repositorios en GitHub son gratuitos y se pueden configurar públicos o privados. Por lo que nos permite tener proyectos *open source,* lo que la convierte en una de las plataformas más utilizadas en el mundo. Es multiplataforma por lo que se puede usar para crear repositorios locales en todos los sistemas operativos más comunes, como Windows, Linux o Mac.

Esta plataforma facilita la gestión del proyecto, nos permite trabajar no solo estando dentro de la misma ciudad, sino desde distintas partes del mundo.

La dirección del repositorio del proyecto es: https://github.com/LuciaOses/Spotify-Smalltalk-Lucia-Oses