|  |  |
| --- | --- |
| Proyecto VLC  Proyecto transversal | Descripción breve  Esta documentación corresponde al trabajo de VLC de generar una playlist random aplicando histos alcanzados de cada módulo.  Alberto Mañas y Mateu Massó  Desarrollo de aplicaciones WEB |

Índice

[Definición general del proyecto de software. 2](#_Toc26174325)

[Especificación de requerimientos del proyecto. 2](#_Toc26174326)

[Sistemas Informáticos. 2](#_Toc26174327)

[Bases de datos. 3](#_Toc26174328)

[Lenguaje de marcas. 3](#_Toc26174329)

[Entornos de Desarrollo. 3](#_Toc26174330)

[Programació. 4](#_Toc26174331)

[Procedimientos de desarrollo. 5](#_Toc26174332)

[Herramientas utilizadas: 5](#_Toc26174333)

[Planificación: 5](#_Toc26174334)

[Procedimientos de instalación y prueba. 5](#_Toc26174335)

[Requisitos no funcionales: 5](#_Toc26174336)

[Obtención e instalación: 5](#_Toc26174337)

[Definición gereal del proyecto de software. 6](#_Toc26174338)

[Arquitectura del sistema. 7](#_Toc26174339)

[Descripción jerárquica: 7](#_Toc26174340)

[Diagrama de módulos: 7](#_Toc26174341)

[Descripción individual de los módulos: 7](#_Toc26174342)

[Bibliografía. 8](#_Toc26174343)

## Definición general del proyecto de software.

En este proyecto, el primero que realizaremos en este curso, hemos construido un programa que genera y ejecuta en **VLC** una lista de reproducción **aleatoria** de las canciones que teniamos previamente descargadas en nuestra maquina local y definidas en un **XML** previamente creado por nosotros.

## Especificación de requerimientos del proyecto.

Los requerimientos principales del proyecto son lograr aplicar todos los requisitos que cada profesor ha establecido en su módulo:

### Sistemas Informáticos.

* Crea desde línea de comandos en tu máquina un directorio donde almacenar los ficheros con las canciones. Nos referiremos a este directorio como la biblioteca de canciones.
* Copia o mueve desde línea de comandos las canciones desde el soporte donde se encuentran a tu biblioteca.
* Formatea desde línea de comandos los títulos de las canciones para darles el formato adecuado.
* Instala desde línea de comandos el programa VLC.
* Invoca desde línea de comandos el programa VLC y averigua qué parámetros admite.
* Averigua qué errores genera una mala invocación de VLC.
* Ejecuta el programa VLC desde línea de comandos para reproducir una determinada canción.
* Ejecuta el programa VLC desde línea de comandos para reproducir una lista de canciones.
* Arranca desde línea de comandos el editor que te permita crear el fichero XML con la descripción de las canciones de tu biblioteca que diseñarás en el módulo de lenguaje de marcas.
* Arranca desde línea de comandos el editor que te permite crear un fichero Schema que valida la biblioteca XML y que diseñarás en el módulo de lenguaje de marcas.

### Bases de datos.

* Realiza el modelo E-R de la base de datos que almacenaría las canciones.
* Pasa el modelo E-R al modelo relacional.
* Extrae los atributos que describen las canciones de la biblioteca.
* Define el tipo de dato de cada atributo.
* Establece la clave primaria y la clave foránea.

### Lenguaje de marcas.

* Crea un fichero XML que describa la información contenida en la biblioteca de canciones.
* Escribe su Schema que describa la biblioteca de canciones con el fin de poder validarla.
* Valida el fichero XML. Comprueba que el fichero está bien formado (utilizando el Schema).
* Utiliza el modelo E-R y el modelo relacional que has propuesto en el módulo de bases de datos para definir los tipos de datos de los elementos del fichero XML.
* Utiliza el modelo E-R y el modelo relacional que has propuesto en el módulo de bases de datos para definir los tipos de los elementos del fichero XML.
* Define los espacios de nombres que sean necesarios.

### Entornos de Desarrollo.

* Crea un repositorio del proyecto en GitHub.
* Crea un fichero README.MD que describa adecuadamente el proyecto.
* Realiza el control de versiones y trabaja de manera colaborativa con tu pareja de programación.
* Utiliza ramas para realizar los evolutivos, testing y debugging, siguiendo el flujo de trabajo que especificaremos en las sesiones presenciales.
* Documenta todo el proceso.
* ¿Escribe las historias de usuario/a? = identifica y describe los requisitos funcionales de la aplicación.
* Elige el ciclo de desarrollo que consideres más adecuado a tu manera de trabajar (y a la de tu pareja de programación) y al proyecto.
* Utiliza clockify para llevar un seguimiento del tiempo de desarrollo empleado en cada una de las fases del ciclo de vida del proyecto. Sé riguroso porque los informes generados por este herramienta serán un producto a entregar.

### Programació.

* Divide el código en rutinas de modo que sea SRP y OCP.

• SRP (S) o Principio de Única Responsabilidad o Single Responsibility Principle. Un componente sólo debe tener un motivo para cambiar.

• OCP (O) o Open/Closed Principle. Las entidades de software (clases, módulos, funciones, etc.) deben estar “abiertas” a la extensión pero “cerradas” a la modificación.

* Emplea precondiciones y postcondiciones para chequear las invariantes (las estructuras de datos) que maneja el programa y que se comunican las distintas funciones.
* Utiliza las librerías XML que proporciona el lenguaje Python para extraer del fichero XML (la biblioteca de canciones) la información que necesitas de las canciones para construir la playlist (aleatoria) de canciones que le pasarás al programa VLC.
* Elige la estructura de datos adecuada para representar y manipular en memoria la lista de canciones.
* Utiliza bloques try /except para capturar las excepciones que se puedan producir durante la ejecución del programa.
* Crea una lista **aleatoria** con las canciones de tu biblioteca de música. Si la lista no es aleatoria, el proyecto no se considera resuelto satisfactoriamente.
* Invoca desde Python el programa VLC y pásale la lista de canciones importando las librerías adecuadas.

## Procedimientos de desarrollo.

### Herramientas utilizadas:

Hemos utilizado Visual Studio Code para desarrollar el código utilizando un linter de pycodestyle para que se resaltaran los errores de identación y de erratas humanas al hacer el código.

Por otro lado tras realizar algún cambio, hemos hecho commits utilizando una extensión de VSCode llamada commitizen para establecer un protocolo a la hora de subir archivos y trabajar de forma conjunta con la pareja del proyecto.

### Planificación:

Hemos utilizado una metodogía en cascada, esta consiste en hacer modulo por modulo de tal forma de que el modulo ya modificado queda completamente acabado antes de pasar al próximo.

En un principio nos hicimos un pequeño diagrama con todas las funcionabilidades que iba a tener cada módulo para poder desarrollarlo con mayor eficiencia posible sin tener que ir haciendo modificaciones constantemente. Para ello nos reunimos durante 1h o 2h y compartimos ideas de las funcionabilidades que aportariamos al proyecto.

Dicha tarea fue compleja por la poca experiencia que teniamos ambos en proyectos de este calible, pero tras invertir tiempo en la organización del proyecto fue más sencillo poder empezar a desarrollar y estudiar cada parte que había que implementar.

## Procedimientos de instalación y prueba.

### Requisitos no funcionales:

El inconveniente mayor es la condición de que el fichero XML donde se encuentran los datos de las canciones no siga la estructura que nosotros le hemos aportado, dado a que tiene que seguir la estructura lógica de ese fichero para una ejecución existosa.

### Obtención e instalación:

El programa debe ser ejecutado teniendo python descargado y podríamos ejecutar el archivo main.py desde la carpeta. Hay que especificar que para poder reproducir la lista que está en el xml habrá que tener dicha carpeta con el contenido en un directorio seleccionado en el código.

Si se desea, también se podrá ejecutar desde la misma terminal que Visual Studio Code nos ofrede o incluso desde la terminal de python.

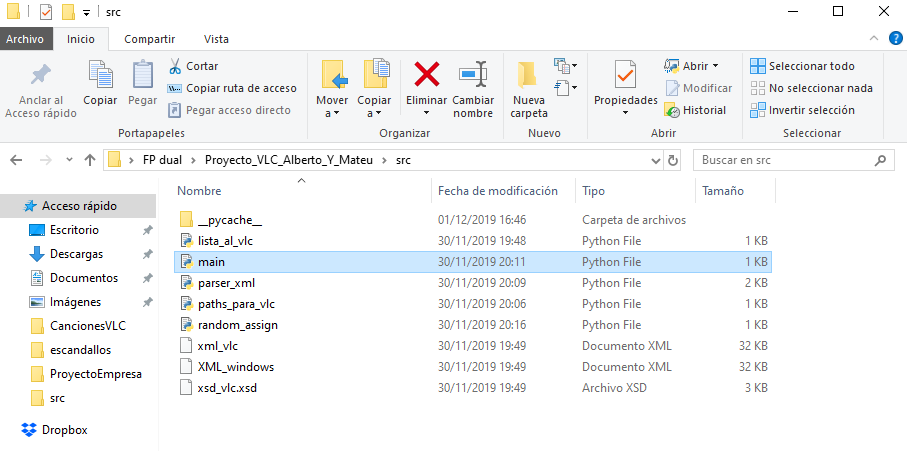


Ilustración 1: ejecución del programa desde carpeta.

### Definición gereal del proyecto de software.

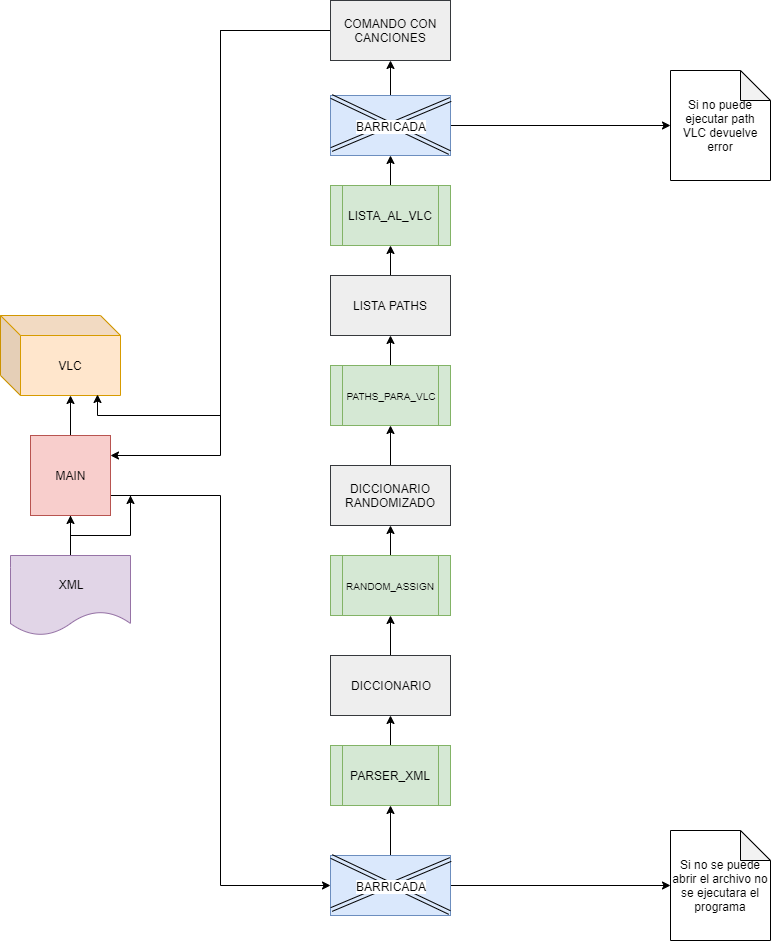
La funcionabilidad del proyecto es reproducir una lista de canciones de forma aleatoria utilizando el reproductor VLC. Para ello hemos tenido que generar distintos módulos con intención de poder hacer éste algorítmo con éxito y conectarlos entre si dependiendo de la funcionabilidad que queremos ofrecer.

El usuario podrá reproducir las canciones que desee de forma aleatoria de tal forma que no habrán autores, compositores ni álbunes del mismo autor consecutivos para ofrecer una experiencia de aleatoriedad.

## Arquitectura del sistema.

La arquitectura del sistema es el corazón de un programa, es la forma en la que está están estructurados los módulos que contiene dicho programa. En nuestro caso hemos aplicado una arquitectura modular donde el main llamará a los distintos módulos para ejecutar el programa.

### Diagrama de módulos:



### Descripción individual de los módulos:

En éste apartado se explicará detalladamente la descripción general y propósito, responsabilidad y restricciones, dependencias e implementación de cada uno de los módulos.

#### parser\_xml.py

**DESCRIPCIÓN GENERAL Y PROPÓSITO 🡪** Éste módulo parsea el xml que hemos generado para el proyecto, su función es sacar los elementos seleccionados del xml y generar un diccionario con los datos deseados.

**RESPONSABILIDAD Y RESTRICCIONES 🡪** La función específica de éste es parsear el xml\_vlc.xml y extraer la información de las etiquetas iterando entre ellas.

Éste, puede encontrar el nombre del artista, el título del álbum, el nombre de la canción, su ruta absoluta, el género de la canción y el compositor de esta.

Sus restricciones son que para poder iterar con éxito y parsear el xml para poder aplicarlo a otro xml deberá contener la misma estructura que el xml utilizado.

**DEPENDENDIAS** 🡪 El módulo para funcionar requiere la importanción de la librería xml.etree.ElemenTree as ET. Esta librería hacer que el parser se pueda realizar con éxito.

Además de que el archivo xml que parseamos contenta la misma estructura en la que nosotros iteramos para extraer la información que anteriormente se ha mencionado.

**IMPLEMENTACIÓN** 🡪 La implementación del parser se encuentra en el main.py

**URL REPOSITORIO** 🡪 <https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/src/parser_xml.py>

#### random\_assigns.py

**DESCRIPCIÓN GENERAL Y PROPÓSITO 🡪** Éste módulo coje un diccionario, hace una copia, añade un contador (En un inicio desde 0) e introduce en un diccionario auxiliar vacío que posteriormente usaremos para guardar el valor la canción anterior.

Elegimos hacer un while porque no sabemos exactamente las cantidades de repetición que tenemos que hacer. Pero sí que sabemos que len del diccionario tiene que ser la misma que la del diccionario original.

Importamos la función random.choice creada con las keys del diccionario y elije un valor de estos aleatorios y lo que miramos el valro dentro de la key cuya key sea artista, album, genero y compositor. Si ningún valor coincide en los valores del diccionario auxiliar,

Para asegurarnos que no vuelva a salir la misma canción, utilizamos diccionario.pop para que dropee la canción y que no contenga el valor de esta en el diccionaro.

Cuando suceda el caso solo haya una canción por pasar por el diccionario y tenga el mismo artista, album, generero o compositor, hemos puesto la condición de que si sucede cuando es la última canción, pase sí o sí

**RESPONSABILIDAD Y RESTRICCIONES 🡪** La función específica dentro del sistema es randomizar el diccionario que generamos con el xml de tal forma que no pueda contener más de un artista, album, genero y compositor consecutivos.

Cuando esto sucede en las últimas posiciones del diccionario, no se aplica.

**DEPENDENDIAS** 🡪 Los requisitos del modulo es que el diccionario\_randomizado no sea una un diccionario vacío.

Éste modulo depende de una librería externa llamada random e importamos choice y

**IMPLEMENTACIÓN** 🡪 Dicha implementación se encuentra en main.py

**URL REPOSITORIO** 🡪 <https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/src/random_assign.py>

#### paths\_para\_vlc.py

**DESCRIPCIÓN GENERAL Y PROPÓSITO 🡪** Coje el diccionario y crea una lista de tuplas con el diccionario que dentro de cada tupla tendrá el order y la localización de la canción. Luego hacemos un sort para que esté ordenada la lista numericamente. A continuación hacemos un for para que en cada valor de la lista coja la segunda posición de la tupla que correnderá a la localización del archivo determinado.

**RESPONSABILIDAD Y RESTRICCIONES 🡪** Su función es coger la localización de cada canción

**DEPENDENDIAS** 🡪 Los requisitos del módulo es generar con éxito las canciones con su correspondiente ubicación.

Éste depende del diccionario generado en el parser\_xml.py.

**IMPLEMENTACIÓN** 🡪 La implementación de éste se encuentra el en main.py

**URL REPOSITORIO** 🡪 <https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/src/paths_para_vlc.py>

#### lista\_al\_vlc

**DESCRIPCIÓN GENERAL Y PROPÓSITO 🡪** Éste módulo asigna una variable a la ruta de VLC para poder llamarlo desde python, si la ruta no coincide con la asignada en ruta\_vlc no se podrá ejecutar por la barricada creada. En el caso que se cumpla, convertimos la lista en una string dado a que la librería os no permite pasar una lista. Gracias a la librería OS, importa os.popen y añadimos el comando que será la suma de la ruta\_vlc +” ”+ canciones(“ “.join(lista)).

**RESPONSABILIDAD Y RESTRICCIONES 🡪** Su principal función es llamar a VLC desde python

**DEPENDENDIAS** 🡪 Depende de que la ruta VLC sea la correcta y que lista sea una lista al igual que la variable declarada comando = ruta\_vlc + “ “ + canciones (canciones = “ “.join(lista)) sea una string.

Hemos importado la librería externa llamada OS para poder aplicar popen y ejecutar VLC con éxito.

**IMPLEMENTACIÓN** 🡪 Se encuentra la implementación en el main.py

**URL REPOSITORIO 🡪** <https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/src/lista_al_vlc.py>

#### main.py

**DESCRIPCIÓN GENERAL Y PROPÓSITO 🡪** Éste modulo es el main, dicho módulo llama a todos los otros para poder ejecutar con efectividad el programa creado.

Comprueba que la ruta\_xml

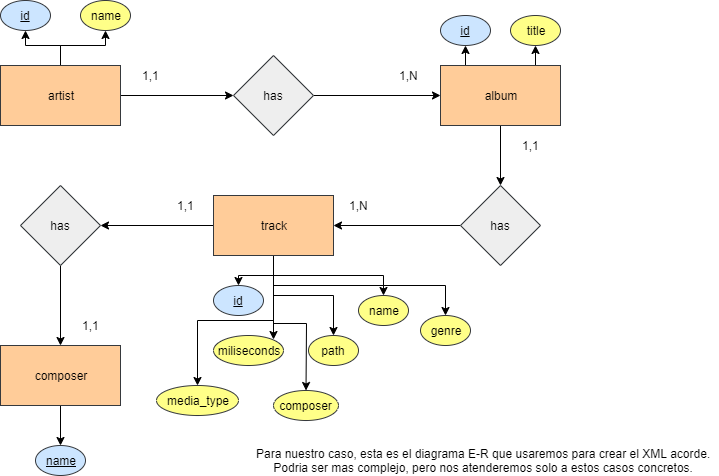
**RESPONSABILIDAD Y RESTRICCIONES 🡪** CUAL ES SU FUNCION ESPECÍFICA DENTRO DEL SISTEMA? QUE COSAS PUEDE Y NO PUEDE HACER?

**DEPENDENDIAS** 🡪 REQUISITOS DEL MODULO, QUE NECESITA O REQUIERE EL MODULO PARA FUNCIONAR? NECESITA DE SERVICIOS BRINDADOS POR OTROS MÓDULOS O POR LIBRERIAS EXTERNAS?

**IMPLEMENTACIÓN** 🡪 EN QUE MODULO SE ENCUENTRA LA IMPLEMENTACIÓN

**URL REPOSITORIO 🡪** <https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/src/main.py>

## Diagrama E-R.



**URL REPOSITORIO 🡪**

## Bibliografía.

* CIFP Repte Alumne\_a VLC random playlist
* <https://docs.python.org/3.6/library/xml.etree.elementtree.html>
* <https://www.python.org/doc/>
* Archivos pasados por Slack, Clasroom y GoogleDrive.