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, IES FB MOLL

Comisión Europea

Índice

Índice de ilustraciones

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.

- 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.
- Averigua cómo ejecutar el programa VLC desde línea de comandos para reproducir una lista de canciones.

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.
- 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.
- Documenta todo el proceso.
- 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 indentación y de erratas humanas al hacer el código.

Para realizar los commits hemos utilizado commitizen para establecer un protocolo adecuado a la hora de realizar los commits y trabajar de forma conjunta con la pareja del proyecto.

Planificación:

Hemos utilizado una metología en cascada, esta consiste en hacer modulo por módulo de tal forma de que el módulo ya modificado quede completamente acabado antes de pasar al próximo. Ya que es la primera vez que utilizamos una metodogía por completo, no hemos aplicado en su totalidad la de cascada, debido a que hemos ido refactorizando código posteriormente de otros módulos.

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 2 horas y compartimos ideas de las funcionabilidades que implementariamos al proyecto.

Dicha tarea fue compleja debido a la poca experiencia que teniamos ambos en proyectos, 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 mayor inconveniente es la condición de que el fichero XML donde se encuentran los datos de las canciones no siga la estructura que nosotros le hemos marcado, dado a que tiene que seguir la estructura lógica determinada por el modelo E-R de la base de datos y el xml.

Obtención e instalación:

El programa debe ser ejecutado teniendo python descargado ejecutando el archivo el fichero

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 ofrece o incluso desde la terminal de python.

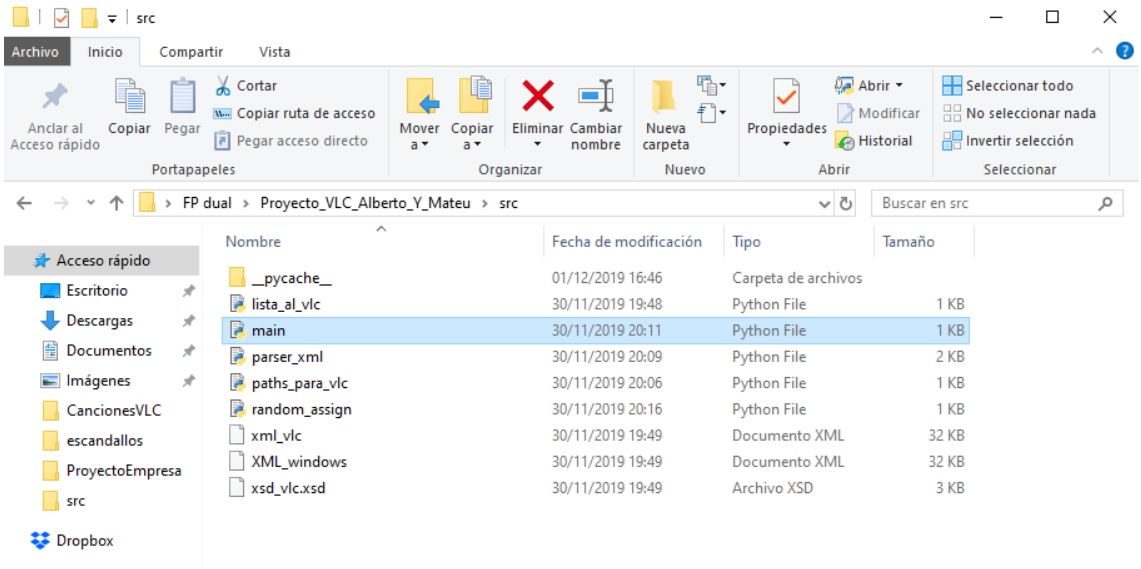

Ilustración 1: ejecución del programa desde carpeta.

Definición general 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 realizar 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 mejorada.

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:

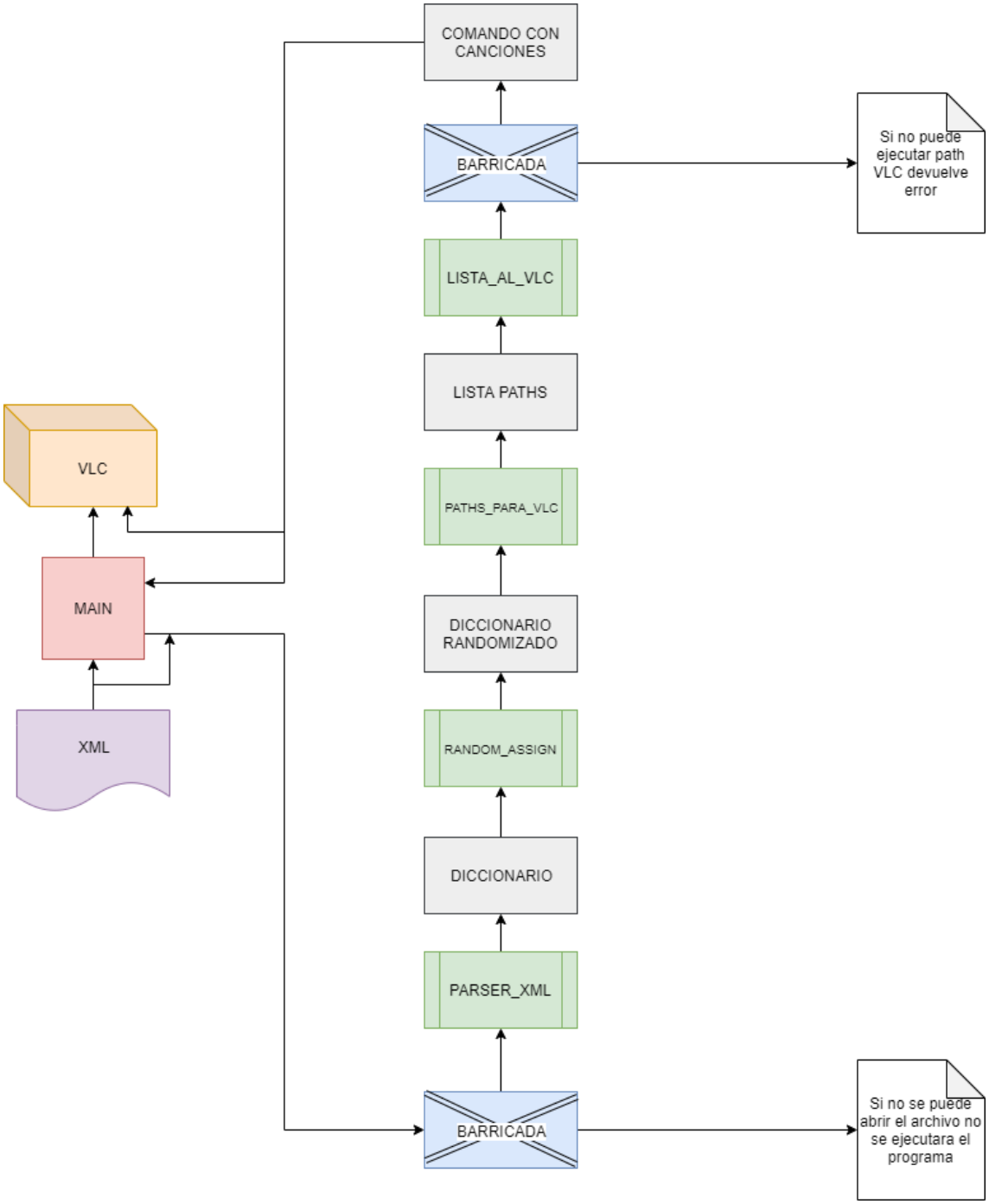

Ilustración 2: Diagrama de módulos.

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. Esta librería hace que el parseo se pueda realizar con éxito.

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_assign.py

DESCRIPCIÓN GENERAL Y PROPÓSITO → Éste módulo coje un diccionario, hace una copia, añade un contador (En un inicio desde 0) y añade un diccionario auxiliar vacío que posteriormente usaremos para guardar el valor de canción anterior.

Elegimos hacer un while porque no sabemos exactamente cuantas veces se repetirá el bucle.

Importamos la función random.choice, la cual su función trata de elegir un valor de una lista aleatoria, la lista aleatoria será una lista de las keys del diccionario, una vez que el random.choice elija una de las keys del diccionario, miraremos que el valor del artista, album, genero y compositor de dentro de la key no coincida con ningún valor del diccionario auxiliar. Si no coincide, guardaremos en una copia del diccionario una key valor dentro de la key principal que sea el orden y el contador respectivamente.

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 éste en el diccionario del random.choice.

Cuando suceda el caso solo haya una canción por pasar por el diccionario y tenga el mismo artista, album, genero 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 última posición del diccionario, no se aplica.

DEPENDENDIAS → Éste modulo depende de una librería externa llamada random e importamos la función choice.

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, 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.

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 la ruta de VLC. Hemos creado una barricada que en el caso de que no exista la ruta determinada te avise de que no tienes VLC instalado. En el caso que se cumpla, convertimos la lista en una string dado a que la librería os no nos 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 convertidas en strings.

RESPONSABILIDAD Y RESTRICCIONES → Su principal función es llamar a VLC desde python y pasarle el comando para ejecutar las canciones.

DEPENDENDIAS → Depende de que la ruta VLC sea la correcta y que lista sea una lista al igual que la variable declarada comando 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.

RESPONSABILIDAD Y RESTRICCIONES → La responsabilidad que tiene es llamar a todas las demás funciones y ejecutarlas. Para que el programa funcione, debemos pasarle a la función main una string con la ruta absoluta del xml.

DEPENDENDIAS → Depende de todos los demás módulos.

URL REPOSITORIO →
https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/src/main.py

Diagrama E-R.

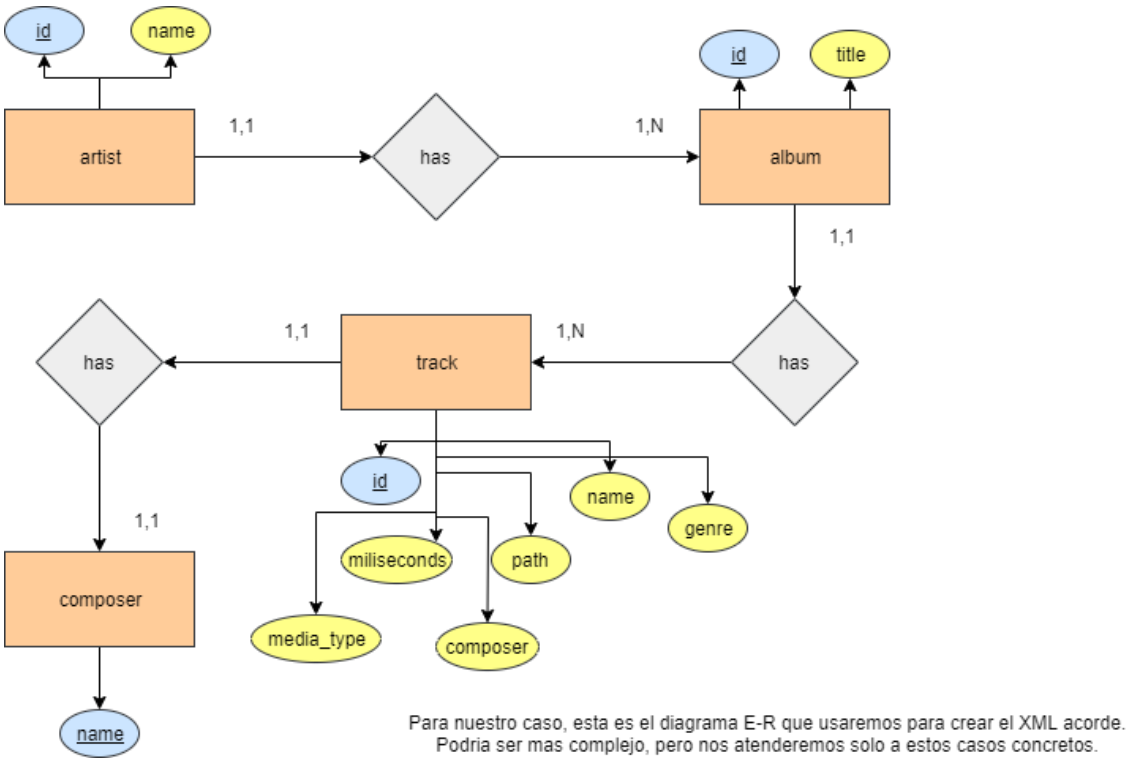

Ilustración 3: E-R del XML.

URL REPOSITORIO →
https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/doc/E-R%20Diagrama.png

Planificación y duración.

URL REPOSITORIO COMPUTO CLOCKIFY TOTAL →
https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/doc/Clockify_Summary_Report_11_07_2019-12_02_2019.pdf

URL REPOSITORIO COMPUTO CLOCKIFY POR USUARIO →
https://github.com/Albertomanas/Proyecto_VLC_Alberto_Y_Mateu/blob/master/doc/Clockify_Summary_Report_11_07_2019-12_02_2019%20(2).pdf

Dificultades y posibles mejoras.

Nuestra mayor dificultad ha sido la búsqueda de información en páginas de terceros para poder completar el proyecto con éxito.

Respecto a las posibles mejoras, un punto a destacar sería variables de entorno para que éste programa sea compatible con distintos sistemas operativos y que el usuario pueda asignar el nombre de la carpeta.

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.