Francesc de Borja Moll
Centre Integrat de Formació Professional

L'FSE inverteix en el teu futur
Fons Social Europeu
UNIÓ EUROPEA

DOCUMENTACIÓN PROYECTO VLC

PROYECTO POR PAREJAS SOBRE LA REPRODUCCIÓN DE UNA LISTA ALEATORIA DE CANCIONES EN VLC

Diseñado por : **Daniel Fernández Rodríguez y José Jiménez Beltrán**

Enlace al repositorio del proyecto:
https://github.com/DanielFernandezR/vlc-random-playlist

Tabla de contenidos

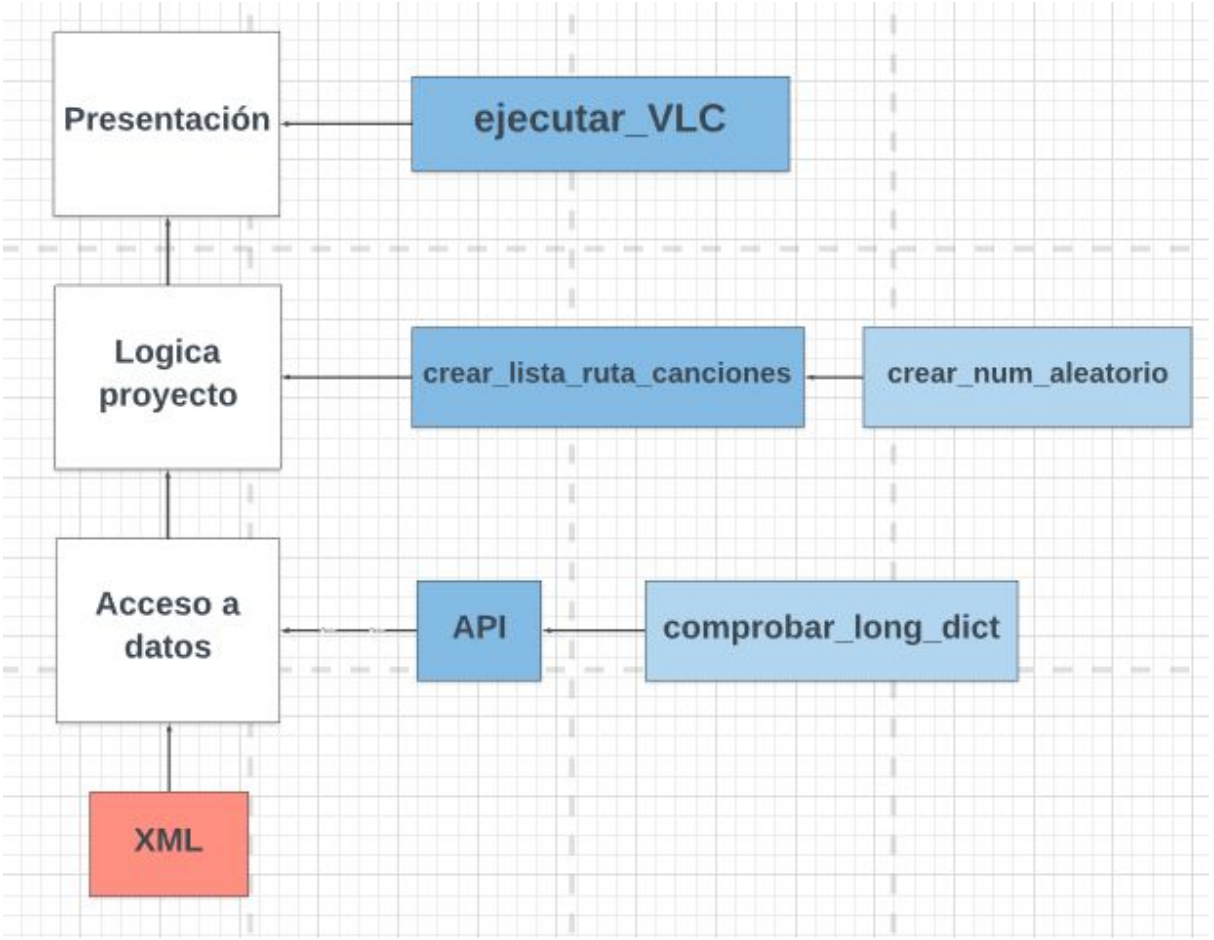
1. Arquitectura de la aplicación y tecnologías utilizadas

1.1. Arquitectura de la aplicación

Como se puede apreciar, nuestro proyecto está dividido en 3 partes y un archivo XML que sirve para definir los datos que vamos a utilizar en nuestra aplicación.

- Capa de acceso a datos: Tenemos un módulo llamado "API" que se encarga de recoger la información del archivo XML y devolver unos datos a la capa de Lógica proyecto. Para poder devolver los datos, dentro de API utilizamos otro módulo auxiliar llamado "comprobar_long_dict".
- Capa de Lógica proyecto: Tenemos un módulo llamado "crear_lista_ruta_canciones" que devuelve un string de las rutas de las canciones a la capa de Presentación, utilizando como módulo auxiliar "crear_num_aleatorio".
- Capa de Presentación: Tenemos un módulo llamado "ejecutar_VLC" que se encarga ejecutar un comando para abrir el programa VLC con las canciones recibidas de Lógica proyecto.

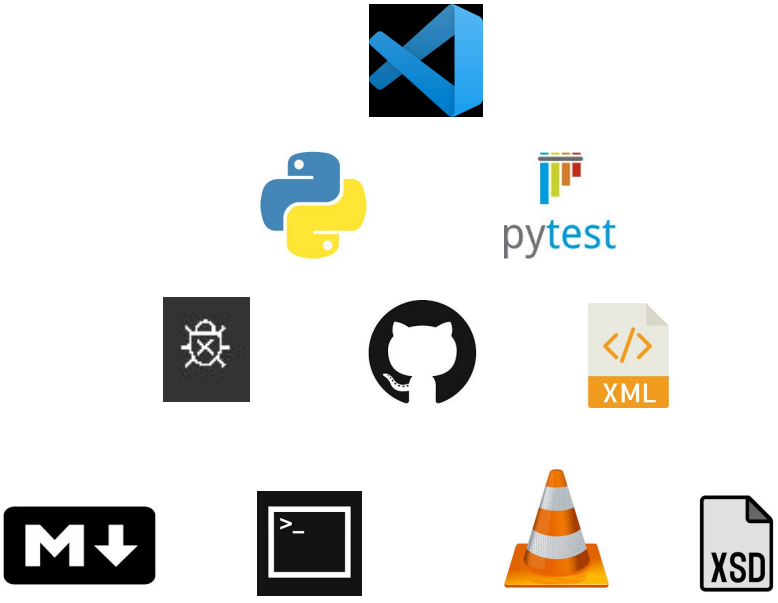
1.1.1 Ilustración de la arquitectura de la aplicación



1.2. Tecnologías utilizadas

Las tecnologías que hemos utilizado para este proyecto son los siguientes:

- VSCode
  - Emmet
  - Pep8
  - Visual Studio Code Commitizen Support
  - Pytest
- Debugging
- XML-XSD
- Markdown
- VLC
- Git
- Python 3.7.4
- Shell Windows



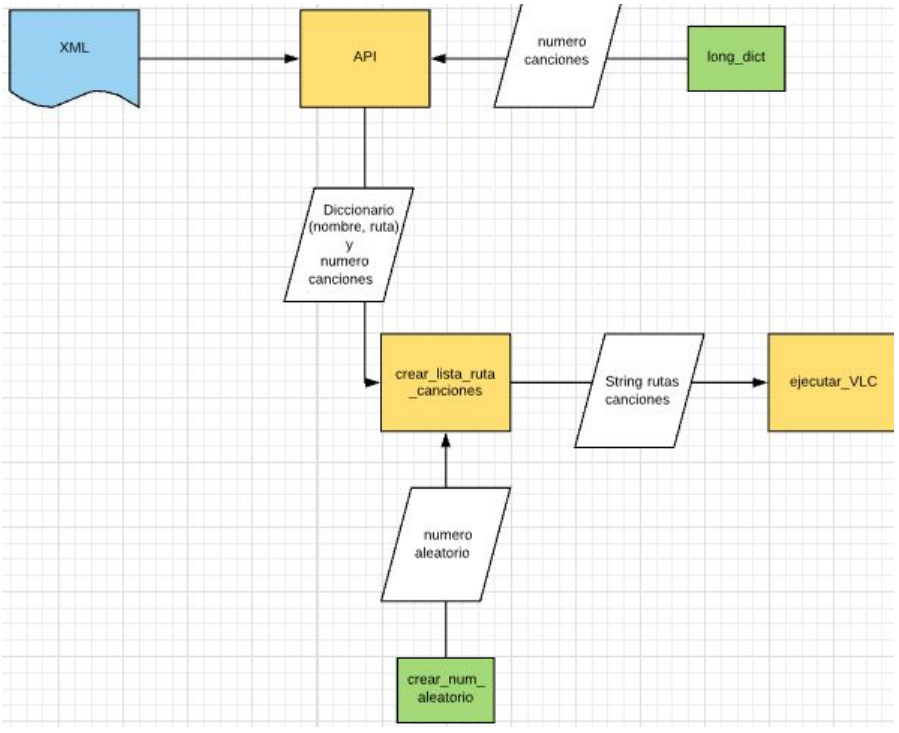
2. Diagrama de componentes

2.1. Cómo funciona nuestra aplicación

La imagen que mostramos a continuación muestra cómo funciona nuestra aplicación de forma completa.

- XML: Es el archivo que contiene las canciones que nuestro programa va a leer.
- API: Esté se encarga de leer el documento XML (parseado) y devuelve un diccionario (con el nombre de la canción y la ruta) y un número, qué es la cantidad de canciones que tiene el susodicho archivo.
    - long_dict: Para que la API sepa qué cantidad de canciones tiene el archivo XML, esté dispone de un submódulo qué se encarga de dar la longitud del diccionario, devolviendo así, el número de la cantidad de canciones.
- crear_lista_ruta_canciones: El módulo recibe lo que la API ha devuelto, y esté se encarga de devolver un string de las rutas de las canciones ordenadas.
    - crear_num_aleatorio: Devuelve un número aleatorio, qué es el orden de la canción que se va a mostrar en el VLC. Este número solo sirve para ordenar dentro del módulo **crear_lista_ruta_canciones**.
- ejecutar_vlc: Recibe las canciones ordenadas en crear_lista_ruta_canciones y e intenta ejecutar el VLC mediante comandos, utilizando primero el comando que ejecuta el programa vlc si está en la variable PATH, si no, intenta ejecutarlo con una ruta especificada dentro, y si no funciona dicha ruta, se muestra un mensaje de error diciendo que el VLC no está en la ruta especificada.

2.1.1 Ilustración del diagrama de componentes

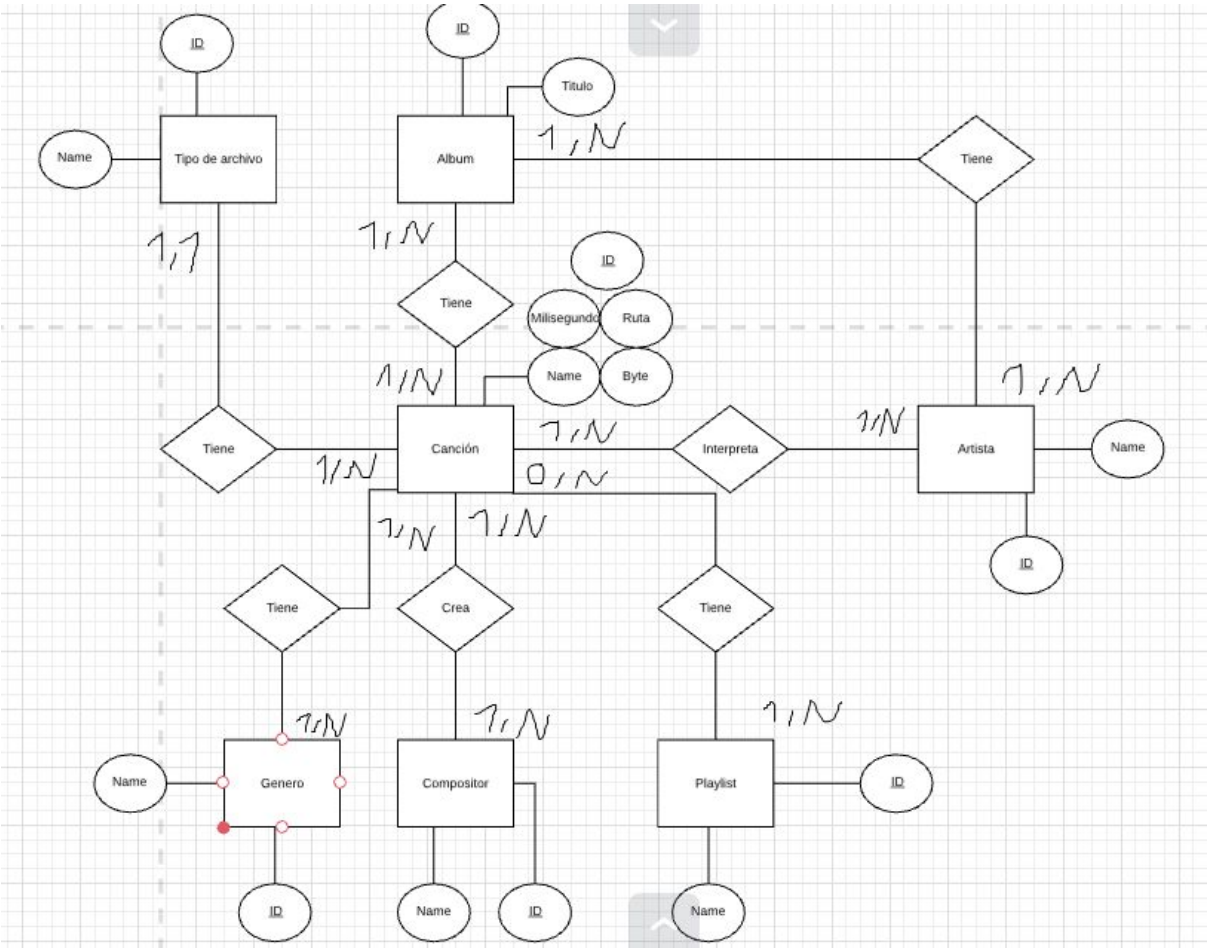


3. Diagrama E/R

3.1. Diagrama Entidad Relación

Hemos creado el diagrama entidad/relación sobre el modelo relacional del ejercicio propuesto en Lenguaje de Marcas.

3.1.1 Ilustración del diagrama E/R



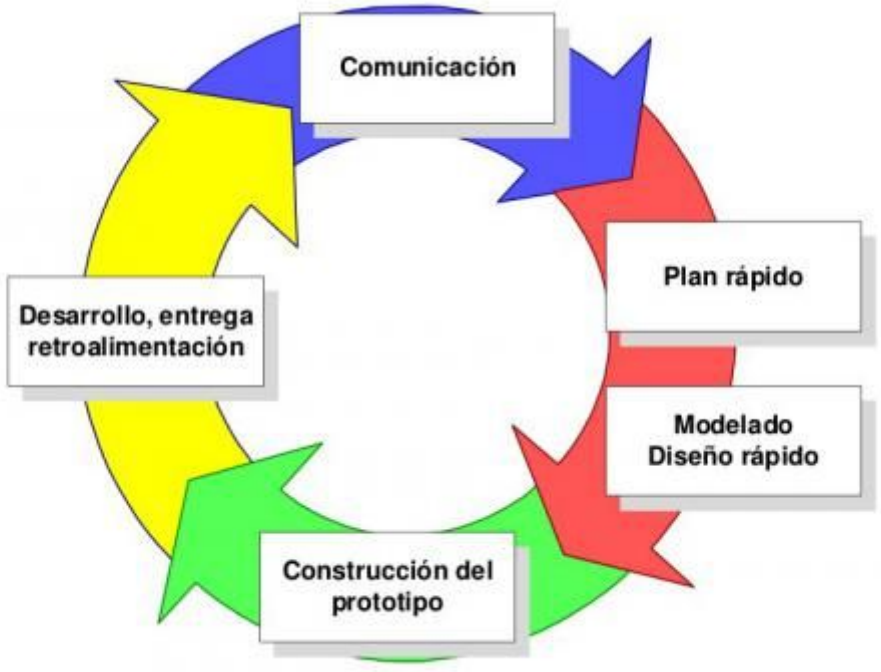
4. Metodologías utilizadas en el proyecto

4.1. Cascada

Lo primero que pensamos fue utilizar la metodología en cascada e ir haciendo todo un módulo y después ir al siguiente solo si hemos terminado con el anterior.
Llegó un momento en el proyecto que no funcionó dicha metodología porque tuvimos fallos en módulos anteriores, y por eso llegamos a la conclusión que la metodología que íbamos a utilizar era la de prototipado.

4.2. Prototipado

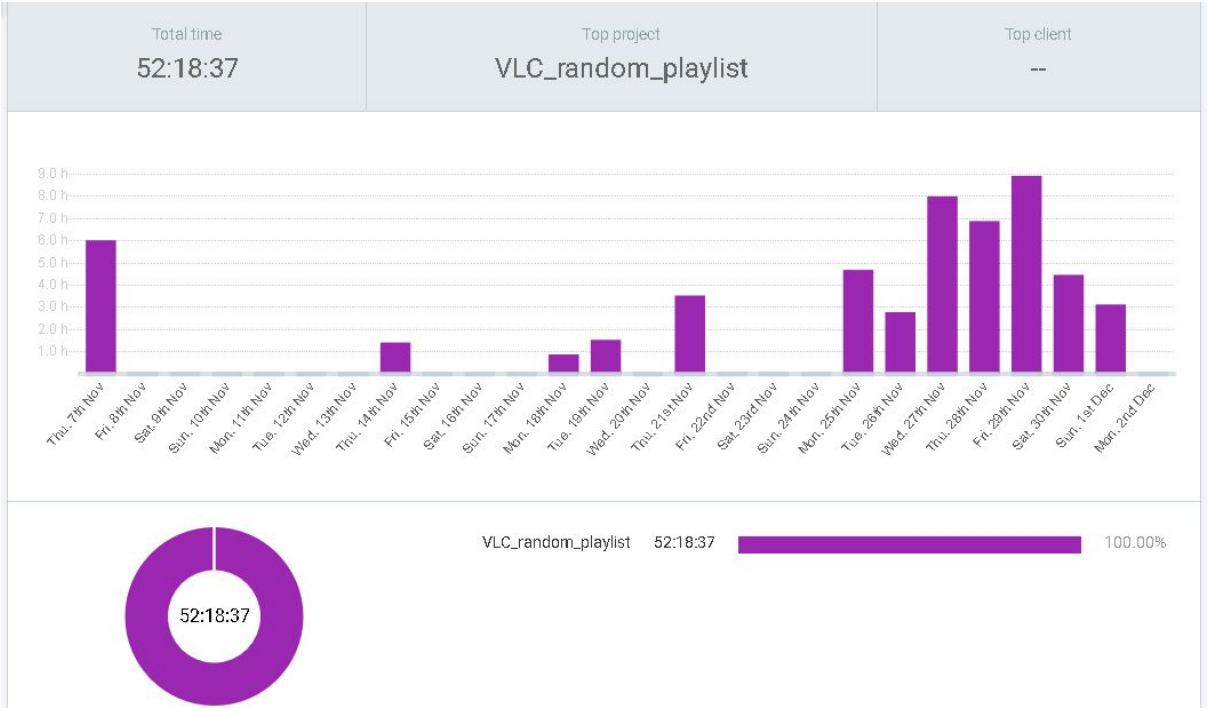
Con la metodología de prototipado conseguimos que funcionará el proyecto con las características mínimas posibles. Entonces llevamos el prototipo a nuestro proyecto y fuimos refactorizando poco a poco añadiendo funcionalidades que veíamos a la hora de refactorizar que iba necesitando el proyecto. Finalmente, con esté logramos avanzar en él proyecto con más comodidad y menos contratiempos.



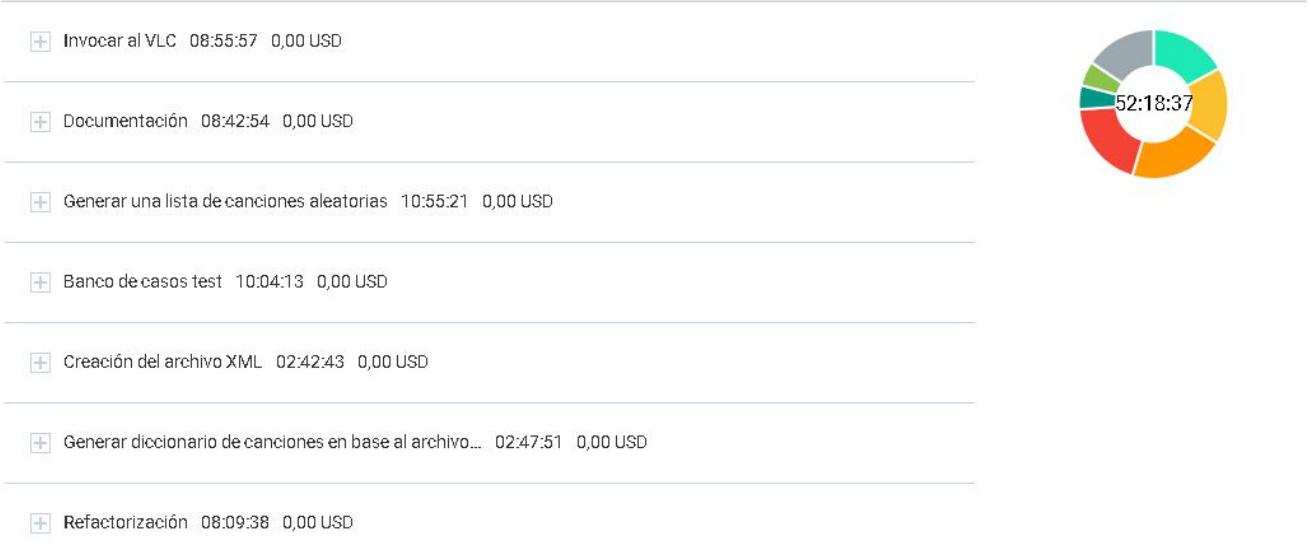
5. Clockify del desarrollo del proyecto

5.1. Diagrama de barras y circular

5.1.1 Ilustración del diagrama de barras del proyecto en Clockify



5.1.2 Ilustración del diagrama circular del proyecto en Clockify



5.2. Tiempo invertido

Como se puede apreciar en las imágenes, donde más hemos invertido tiempo fue en:

- **Generar una lista de canciones aleatorias**: Tuvimos muchas dudas desde el principio y no sabíamos cómo implementarlo correctamente.
- **Invocar al VLC desde python**: No supimos encontrar todas las opciones que ofrece el módulo subprocess, y nos llevó demasiado tiempo encontrar la manera que nos viniera bien para nuestro proyecto.
- **Refactorización**: Una vez terminado el prototipo, el resto del tiempo fue ir mejorando lo mínimo que teníamos hecho a un proyecto en condiciones.
- **Banco de casos test**: No parábamos de darle vueltas a la cabeza para poder encontrar un caso test que fuera correcto y conciso.
- **Documentación:** Queríamos que nuestra documentación fuera lo más clara posible para el lector dando toda la información posible.
- **Generar diccionario de canciones en base al archivo XML:** Seguimos las instrucciones de como encontrar lo que queríamos dentro de un XML y no tuvimos apenas problemas con este tema.
- **Creación del archivo XML:** Simplemente íbamos añadiendo poco a poco más canciones para seguir probando nuestra aplicación.

6. Conclusiones

6.1. Posibles Mejoras

- Más casos test.
- Realizar más commits correctos
- Trabajar más con las ramas de trabajo
- Definir desde el principio una estructura del proyecto más sólida
- Utilizar desde el principio clockify

6.2. Dificultades

La mayor dificultad del proyecto fue no saber qué estructura queríamos que tuviera el proyecto, y hasta el final del tiempo establecido para la entrega, no lo tuvimos bien estructurado, pero gracias a la ayuda de nuestro tutor pudimos ver cual era la estructura que debía tener. Al saber cómo quería que estuviera estructurado el proyecto, lo hicimos sin problemas.

Otra dificultad de este proyecto han sido los casos test, ya que aún sabiendo que casos test queríamos hacer, no conseguimos hacer que funcionara. No conseguimos hacer que el resultado del caso test fuera el que nosotros esperábamos.

También tuvimos muchos problemas a la hora de comunicar python con el VLC, ya que no sabíamos que código utilizar para llamarlo correctamente.