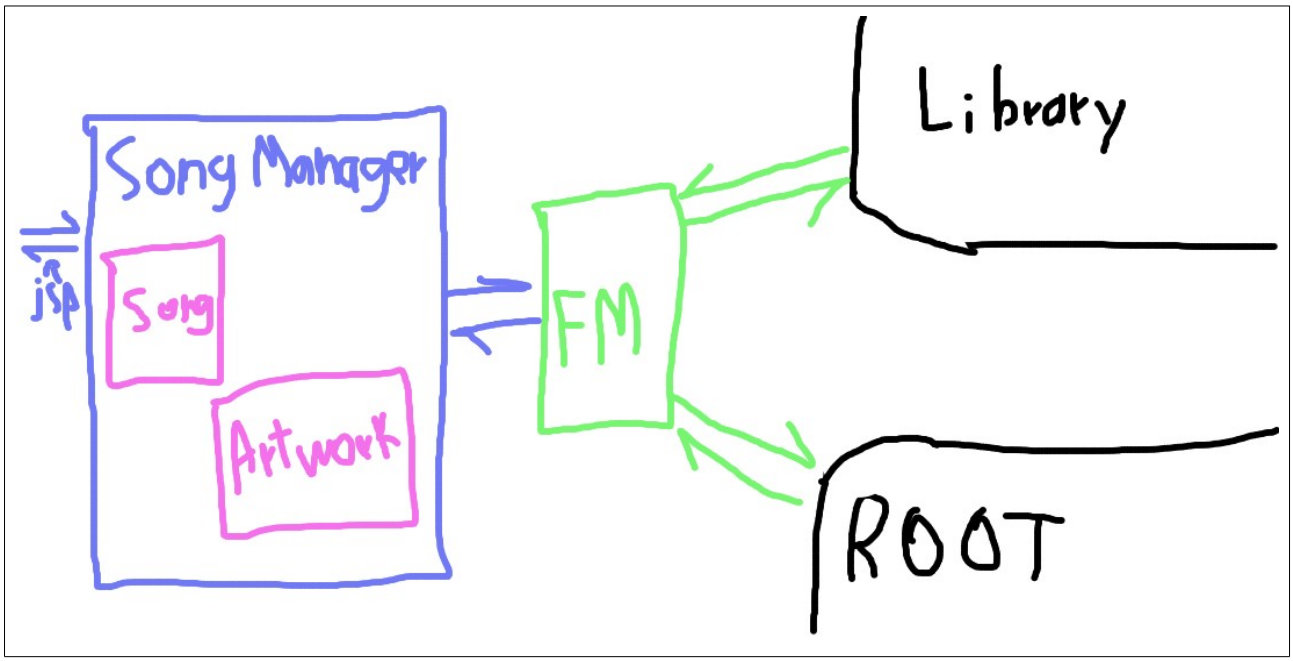
Analisis

Como requerimiento del proyecto, se debe hacer lectura de la carpeta “Library” en “eclipse/jee-2019-12/eclipse”, esto indica que el programa JAVA se ejecutará desde la carpeta de Eclipse. No obstante, el servidor y localhost será ejecutado desde la carpeta “Servers/tomcat/webapps/ROOT/”, esta carpeta no se encuentra en el directorio de eclipse por lo que genera un problema a resolver. Este problema es el de poder leer la canción deseada por el usuario fuera de la carpeta de localhost, lo cual no se puede por razones de seguridad del protocolo HTTPS. La solución a este problema es copiar o mover la canción deseada a la carpeta de localhost. Para la lectura de la carpeta de Librería no se sabe como está estructurada la carpeta internamente, solamente se menciona la posibilidad que contenga archivos de tipo .mp3, .ogg, .jpg, .jpeg, y .png con un formato especifico para poder identificar información de las canciones o artes de album que se encuentren. De igual manera se menciona que si el formato del nombre de alguno de estos archivos no coincide, se debe de agregar de igual manera, pero como canción desconocida.

El analisis a lo anteriormente mencionado ayuda a tener ciertas conclusiones:

1. Tiene que existir un unico objeto que haga manejo de todo archivo fuera del proyecto, que pueda leer, mover, o copiar archivos y enlistar rutas hacia estos archivos.
2. Debe de existir un objeto de tipo Canción, en el cual contenga información de la misma.
3. Debe existir un objeto de tipo Arte de Album, para obtener informacion de la misma.
4. Habrá una clase que encapsule y administre los dos objetos anteriormente mencionados, para mejor seguridad y solo brindar la informacion necesaria al cliente.

Visualización del analisis:



Del siguiente analisis, deben surgir los siguientes archivos:

Directorio	Archivo
JAVA Resources	**FileManager.java:** Esta clase se encargará del manejo de archivos, haciendo lectura, escritura, y borrar. Para obtener las canciones se utilizará un recorrido de arbol, en donde cada carpeta dentro de Library será tomada como un sub-arbol para recursivamente buscar el tipo de archivo que se desea. Para copiar o mover la canción deseada se deberá de copiar sus contenidos como bytes y crear un archivo con el mismo contenido pero en la carpeta webapps/ROOT de tomcat.
	Song.java: Es un objeto que representa un archivo de canción encontrada en el directorio de Library. Contiene información tal como el nombre de la canción, el artista de la canción y el album. Asimismo este contendrá la ruta en donde se encuentre la canción.
	Artwork.java: En lugar de representar una canción, este objeto representará una foto de arte de album, su artista, su album y su ruta dentro de Library.
	SongManager.java: Esta clase administrará una lista de Song y Artwork para la información que se necesite obtener de las mismas. Entre las operaciones que tendrá estará el de crear la lista de objetos Song y Artwork haciendo uso de FileManager para obtener la información de los archivos que se encuentren en Library. Cuando se desee escuchar una canción, SongManager se encargará de buscar la cancion dentro de su lista y enviarsela a FileManager para realizar la operación de copiar el archivo a webapps/ROOT. De no tener un nombre de archivo esperado, se agregará la cancion como desconocida.
	Constants.java: Esta clase guardara variables que se utilizaran frecuentemente por todo el proyecto.
	Validator.java: Esta clase nos ayudara para validar request que son isntancias de un HttpServletRequest y cualquier otro objeto que se requiera llegar a validar.
	GetLyrics.java: Esta clase se encargara de leer el html y

	almacenar lo para poder hacer el parseo de la letra y tambien obtendra el JSON de una api para usar la letra.
	ExecuteZIP.java: Esta clase se encarga de crear el archivo .py para la creacion del zip, tambien crea un archivo .sh para la ejecucion de scripts necesarios para el manejo de los archivos a utilizar.
controllers	**LyricsController.jsp:** Se encargar de controlar el flujo de las letras y se encarga de decidir si la letra sera buscada por medio de una API o por medio de la lectura del html y el parseo de la misma para encontrar la letra
	SoapController.jsp: Controlara el webService que se utilizara para tener otro metodo el cual se encarga de buscar letras de canciones ya sea por una api o por un parse de un HTML
	SessionManager.jsp Este controla la sesion que guarda los elementos seleccionados por el usuario, a su vez maneja la creacion del .zip de las canciones seleccionadas, para poder descargar dicho archivo comprimido.
public	**Index.jsp:** Este es quien contiene todo el contenido **html** de la pagina principal, asi como sus eventos y creacion de objetos.
scripts	**Index.js:** Este archivo controla todos los elementos html de la pagina principal del reproductor, tanto para el control de las canciones haciendo uso de la etiqueta de <audio>. Tambien contiene las funciones de los eventos de elementos como: descargar, cambiar metodo de obtencion de letra, redireccion a la pagina de lyric web service, busqueda de cancion en la barra de busqueda.
styles	**Index.css** Este archivo css contiene todo el estilo de la pagina principal, es decir, del reproductor de musica.