**Recuperatorio Introducción a las Metodologías de Desarrollo de Software (22/6/2022)**

Nos contactaron de una startup llamada Musicfy para que desarrollemos una plataforma web para su empresa. Musicfy busca ser la plataforma web donde todos escuchen música subida por sus propios usuarios. Para poder acceder a la plataforma, un usuario debe primero registrarse. Durante el registro se solicita, nombre y apellido, fecha de nacimiento, géneros musicales que le gustan (Rock, pop, folk y cumbia), nombre de usuario, email y contraseña (debe ser de al menos 8 caracteres alfanuméricos). Para evitar la creación de cuentas falsas, el sistema le envía un mail al usuario con un link de validación e indicando un plazo de 48hrs para completar la registración. El usuario para terminar de completar su registro, deberá acceder a su mail personal y por medio del link enviado a su mail acceder al sistema para validar su cuenta. Cuando realiza esta actividad el sistema activa la cuenta del usuario permitiéndole acceder a la plataforma.

El usuario puede buscar artistas, álbumes, canciones y/o playlists. Los resultados de búsqueda podrán ser ordenados por relevancia, alfabéticamente o por número de reproducciones. Además, deberá poder filtrarse por el tipo de resultado (artistas, álbumes, canciones, playlists). Una vez realizada la búsqueda podrá seleccionar un resultado y reproducirlo.

Cualquier usuario registrado podrá agregar música a la plataforma, pudiendo agregar artistas, álbumes y/o canciones. Para agregar un artista la plataforma deberá solicitar nombre y apellido del artista, nombre artístico (en caso de tenerlo), nacionalidad, y, opcionalmente, una o más fotos del mismo.

Para agregar un álbum, se solicitará nombre del álbum, año de edición, cantidad de canciones, una descripción, una imagen de la carátula del mismo, y deberá seleccionarse el artista al que pertenece. En caso de que el artista no exista aún en la plataforma, la misma deberá solicitarle que cargue el artista antes de continuar con la carga del álbum. Durante la carga de un álbum, el usuario deberá agregar una o más canciones que **componen** al mismo. Para agregar una canción, el usuario deberá subir el archivo de audio de la misma (sólo se aceptan en formato mp3), indicar nombre de la canción y opcionalmente la letra de la misma.

Adicionalmente, el usuario podrá incorporar canciones a un albúm existente. Para ello, la plataforma le solicitará el nombre del albúm al que desea agregar una canción. Luego, le solicitará el archivo de audio correspondiente (sólo se aceptan en formato mp3) y que indique el nombre de la canción. En caso que el albúm no exista, la plataforma le indicará la situación y no podrá cargar la canción.

Finalmente, cualquier usuario registrado podrá crear playlists. Una playlist estará **compuesta** por uno o más artistas, albunes, canciones y/o playlists que ya estén disponibles en la plataforma.

**Ejercicio 1**

**a)** Especifique los requerimientos utilizando user stories

**b)** Realice el user story mapping del sistema. No es necesario indicar los releases.

**Ejercicio 2**

Dado el siguiente código fuente parcial de la aplicación descrita en el Ejercicio 1 realice el diagrama de clases correspondiente. Solo tenga en cuenta las clases, métodos y atributos indicados en el código fuente. No es necesario incluir los nombres de parámetros. Incluya adornos cuando corresponda. No incluya relaciones que no puedan ser deducidas del código fuente *o de la narrativa del Ejercicio 1*.

```java
public class Album extends ElementoReproducible {
	private int edicion;
	private int nroCanciones;
	private Artista artista;
	private String descripcion;
	public Album(String nombre, int edicion, int nroCanciones, Artista artista,String
descripcion){
		super(nombre);
		this.edicion = edicion;
		this.nroCanciones = nroCanciones;
		this.artista = artista;
		this.descripcion=descripcion;
	}
	public void reproducir() {// TODO Auto-generated method stub}
	public List<ElementoReproducible> buscar(String keyword) {
		List<ElementoReproducible> ret=new ArrayList<ElementoReproducible>();
		if (keyword.equals(nombre) || descripcion.contains(keyword))
			ret.add(this);
		return ret;
	}
}
public class Artista extends ElementoReproducible {
	private String nombreReal;
	private String apellido;
	private String nacionalidad;
	public Artista(String nombreArtistico, String nombreReal,String apellido, String
nacionalidad){
		super(nombreArtistico);
		this.nombreReal=nombreReal;
		this.apellido=apellido;
		this.nacionalidad=nacionalidad;
	}
	public void reproducir() {// TODO Auto-generated method stub}
	@Override
	public List<ElementoReproducible> buscar(String keyword) {
		List<ElementoReproducible> ret=new ArrayList<ElementoReproducible>();
		if (keyword.equals(nombre) || keyword.equals(nombreReal) || keyword.equals(apellido))
			ret.add(this);
		return ret;
	}
}
public class Musicfy {
	public List<ElementoReproducible> pruebaDeBusqueda(String keyword, Album album){
		Cancion c1=new Cancion("Englishman in New York", album);
		Cancion c2=new Cancion("History Will Teach Us Nothing", album);
		Playlist p1=new Playlist("Canciones Sting");
		p1.addElemento(c1);
		p1.addElemento(c2);
		return p1.buscar(keyword);
	}
}
public interface Buscable {
	public abstract List<ElementoReproducible> buscar(String keyword);
}
public class Cancion extends ElementoReproducible {
	private String letra;
	private Album album;
	public Cancion(String nombre,Album album) {
		super(nombre);
		this.album=album;
	}
	public void reproducir() {// TODO Auto-generated method stub}
	@Override
	public List<ElementoReproducible> buscar(String keyword) {
		List<ElementoReproducible> ret=new ArrayList<ElementoReproducible>();
		if (keyword.equals(nombre) || letra.contains(keyword))
			ret.add(this);
		return ret;
	}
}
```

```java
public abstract class ElementoReproducible implements Buscable{
	protected String nombre;
	private int nroDeReproducciones;
	public ElementoReproducible(String nombre) {
		this.nombre=nombre;
		nroDeReproducciones=0;
	}
	public String getNombre() {
		return nombre;
	}
	public int getNroDeReproducciones() {
		return nroDeReproducciones;
	}
	public abstract void reproducir();

	public abstract List<ElementoReproducible> buscar(String keyword);

}
public class Playlist extends ElementoReproducible {
	private List<ElementoReproducible> elementos;


	public Playlist(String nombre) {
		super(nombre);
		this.elementos=new ArrayList<>();
	}

	public void addElemento(ElementoReproducible el) {
		if(!elementos.contains(el))
			elementos.add(el);
	}
	@Override
	public void reproducir() {
		// TODO Auto-generated method stub

	}


	@Override
	public List<ElementoReproducible> buscar(String keyword) {
		List<ElementoReproducible> ret=new ArrayList<ElementoReproducible>();
		for (Iterator<ElementoReproducible> iterator = elementos.iterator();
iterator.hasNext();) {
			ElementoReproducible elementoReproducible = (ElementoReproducible)
iterator.next();
			if(elementoReproducible.getNroDeReproducciones()>0)
				ret.addAll(elementoReproducible.buscar(keyword));
		}
		return ret;
	}

}
```

**Ejercicio 3**

Teniendo en cuenta el código fuente del ejercicio 2, realice el diagrama de secuencia para el caso que un objeto m1 de tipo Musicfy recibe el mensaje pruebaDeBusqueda con los parámetros “New York” y el objeto albumSting de tipo Album. Incluya en su diagrama los objetos de java.util intervinientes.