TEC.AZUAY
INSTITUTO UNIVERSITARIO

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

EVALUACIÓN PRÁCTICA DE TENDENCIAS ACTUALES DE LA PROGRAMACIÓN

El objetivo de esta práctica es **implementar una API REST** de listas de reproducción musicales. Para ello haremos uso de Spring Boot para Java. La Figura 1 muestra el esquema general de la práctica.

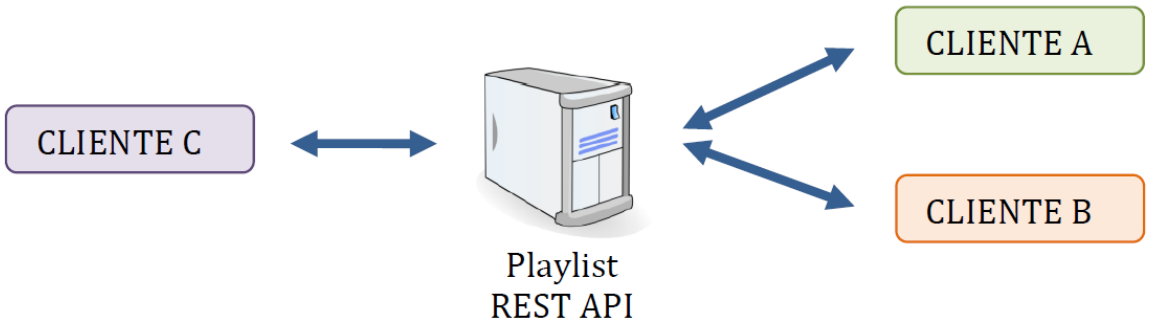

Figura 1. Esquema general de la práctica

El objetivo de este ejercicio es implementar una API de listas de reproducción cuyos requisitos se detalla en la Tabla.

HTTP	Plantilla URI	Descripción
POST	/crear	Añadir una nueva lista de reproducción. Si se añade satisfactoriamente, devuelve "201 Created" con la referencia a la URI y el contenido de la lista. Si el nombre de la lista no es válido (ej: null) debe devolver un error "400 Bad Request".
GET	/lists	Ver todas las listas de reproducción existentes
	/{listName}	Ver descripción de una lista de reproducción seleccionada. Si la lista no existe, debe devolver un "404 Not Found".
PUT	/{actualizar}	Modificar el contenido de una lista de reproducción (salvo el nombre). Si se realiza correctamente, debe devolver un "204 No Content". Si la lista no existe, debe devolver un "404 Not Found". Si intenta modificar el nombre, debe devolver un "409 Conflict".
DELETE	/{listName}	Borrar una lista de reproducción. Si se realiza correctamente, debe devolver un "204 No Content". Si la lista no existe debe devolver un "404 Not Found".

La representación JSON de los recursos del servicio es:

Lista de Reproducción: Una lista de reproducción tiene nombre, descripción y un conjunto de canciones.

```
{
        “name": "Lista 1",
        "description": "Lista de canciones de Spotify",
        “songs”: [
                {
                        "title": "",
                        "artist": "",
                        "album": "",
                        "year": ""
                },
                {
                        "title": "",
                        "artist": "",
                        "album": "",
                        "year": ""
                }, ...
            ]
}
```

Canción: Cada canción tiene un título, nombre del artista, álbum y año.

```
{
        "title": "",
        "artist": "",
        "album": "",
        "year": ""
}
```

CRITERIO	DESCRIPCIÓN	VALORACIÓN	CRITERIOS DE VALORACIÓN
Define la Entidad y un Repositorio	Crea la entidad con los atributos validados correctamente Define y extiende la interfaz del Repositorio correctamente	1.0 0.5	• No crea **(0 pts.)** • Crea la entidad, pero los atributos no son los correctos **(0.5 pts.)** • Crea la entidad, con los atributos correctos (**1 pts.**) • No extiende la interfaz del repositorio correctamente (**0 pts.**) • Extiende la interfaz del repositorio Correctamente (**0.5 pts**.)
Define la Clase Controladora	Crea la clase controlador con los métodos GET, POST, PUT, DELETE correctamente	2.5	• No crea **(0 pts.)** • Crea la clase controlador con la anotación correcta pero no define los operadores (**0.5 pts.**) • Crea la clase controlador con la anotación correcta pero solo define el operador GET (**1 pts.**) • Crea la clase controlador con la anotación correcta pero solo define los operadores GET, POST (**1.5 pts.**) • Crea la clase controlador con la anotación correcta pero solo define los operadores GET, POST, PUT (**2.0 pts**.) • Crea la clase controlador con la anotación correcta, define todos los operadores GET, POST, PUT, DELETE (**2.5 pts.**)
Crea el microservicio	Crea el microservicio e implementa los métodos de las operaciones CRUD correctamente	2.0	• No crea **(0 pts.)** • Crea la clase service con la anotación correcta pero no define los métodos CRUD (**0.5 pts.**) • Crea la clase service con la anotación correcta pero solo define la operación Create de los métodos CRUD (**1 pts.**) • Crea la clase service con la anotación correcta pero solo define las operaciones Create y Read de los métodos CRUD (**1.5 pts.**)

			• Crea la clase service con la anotación correcta pero solo define las operaciones Create, Read y Update de los métodos CRUD (**1.75 pts.**) • Crea la clase service con la anotación correcta y define todas las operaciones: Create, Read, Updatey Delete de los métodos CRUD (**2.0 pts.**)
Validaciones	Realiza el microservicio con las validaciones correctas	2.0	• No despliega (**0 pts.**) • Las validaciones son parcialmente correctas (**1 pts.**) • Todas las validaciones son correctas (**2 pts.**)
Ejecuta las pruebas del microservicio en postman/insomnia/wsagger	Realizar las pruebas del microservicio con los métodos de validación solicitados	1.0	• No se ejecuta la prueba (**0 pts**.) • Se ejecuta pero las validaciones no son correctas (**0.25 pts**.) • Se ejecuta pero se valida parcialmente los campos (**0.50 pts**.) • Se ejecuta y todas las validaciones son correctas (**1.0 pts**.)
Despliega el código fuente en GitHub	Despliega el código fuente en el repositorio de GitHub	1.0	• No despliega (**0 pts.**) • Si despliega (**1 pts.**)