**Automatización de pruebas con Karate y OAuth 2.0**

**Herramientas y configuraciones necesarias**

Para realizar la automatización de pruebas sobre la API de Spotify, se utilizó el framework **Karate DSL**, complementado con las siguientes extensiones necesarias:

* **Karate Syntax**
* **Karate Runner**
* **Karate (plugin principal)**
* **Cucumber for Java (si se integra con Maven o Gradle)**

Estas extensiones permiten trabajar con archivos .feature utilizando sintaxis Gherkin, así como ejecutar escenarios automatizados desde una consola o entorno de desarrollo (como IntelliJ o VS Code).

**Autenticación: Generación de token OAuth 2.0**

Antes de ejecutar las pruebas automatizadas, es necesario contar con un **token de acceso válido (OAuth 2.0)**, ya que muchos endpoints de la API de Spotify requieren autenticación con permisos específicos.

**Pasos realizados:**

1. **Registro de aplicación en Spotify Developer Console**
   * Se creó una aplicación desde [Spotify Developer Dashboard](https://developer.spotify.com/dashboard)
   * Esto generó un client\_id y un client\_secret
2. **Solicitud del token de acceso**
   * Se realizó una petición HTTP POST al endpoint:

https://accounts.spotify.com/api/token

* + El cuerpo de la solicitud incluye el grant type:

grant\_type=client\_credentials

* + La autenticación se realiza en el encabezado como Basic base64(client\_id:client\_secret)
  + La respuesta contiene un token válido por **1 hora**

1. **Automatización del flujo de autenticación**
   * En Karate se creó un archivo .feature que se encarga de generar y capturar el token dinámicamente
   * Este token luego se reutiliza en todos los demás archivos .feature para autorizar cada solicitud contra la API

**Estructura de la automatización**

La automatización fue dividida en varios archivos .feature, cada uno enfocado en probar distintos endpoints de la API, como:

* **GET** para consulta de episodios, shows, artistas, playlists, etc.
* (Nota: no se incluyeron PUT ni DELETE en esta fase, ya que requieren configuraciones adicionales y scopes más avanzados de autorización)

Todos los casos de prueba automatizados fueron **exitosos** usando el token dinámico, lo que permitió validar con rapidez el comportamiento funcional de los recursos consultados.