SPOTIFY TO VYNIL

HECHO POR:

- ELÍAS GONZÁLEZ VALDEPEÑAS (elias.gonzalez@alu.uclm.es)
- RODRIGO DE LA HOZ LÓPEZ (rodrigo.de@alu.uclm.es)
- ALBERTO PALENCIA QUIÑÓNES (alberto.palencia1@alu.uclm.es)

Desarrollador Backend -> Alberto Palencia

Desarrollador Frontend -> Elías González

Desarrollador Testing -> Rodrigo de la Hoz

Enlace GitHub

https://github.com/AlbertoPalenciaQuinones/ISI-LAB.git

Índice

Manual de Instalación del Entorno

1. Requisitos Generales

Para ejecutar correctamente el proyecto 'Spotify to Vinyl' en Windows, se requiere tener instaladas una serie de herramientas, así como una configuración básica del entorno.

2. Herramientas necesarias en Windows

1. Python 3.9 o superior
   - https://www.python.org/downloads/
   ✓ Asegúrate de marcar "Add Python to PATH" durante la instalación.

2. MySQL Server 8.0 o superior
   - https://dev.mysql.com/downloads/installer/
   ✓ Crear una base de datos llamada `musicfinder`.

3. Git para Windows
   - https://git-scm.com/downloads
   ✓ Incluye Git Bash, que se utilizará para ejecutar Makefile.

4. Chocolatey
   - Ejecutar en PowerShell como administrador:

Set-ExecutionPolicy Bypass -Scope Process -Force
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072
iex ((New-Object
System.Net.WebClient).DownloadString("https://community.chocolatey.org/install.ps1"))

5. make (instalado con Chocolatey)
   - Ejecutar:
     choco install make

3. Ejecución completa del proyecto

Sigue estos pasos en la carpeta `src/` para configurar, ejecutar y verificar que todo está funcionando correctamente:

Paso 1. Instalar entorno virtual y dependencias:
   make install

Paso 2. Iniciar la aplicación web:
   make run

Paso 3. Ejecutar los tests (APIs, BBDD y Flask):
   make test

Paso 4. Cargar los datos a la base de datos:

make init-db

Paso 5. Limpiar entorno virtual:
make clean

4. Verificación del entorno

Puedes comprobar que las herramientas están correctamente instaladas con los siguientes comandos:
python --version
pip --version
mysql --version
git --version
choco --version
make –version

5. Notas adicionales

- Asegúrate de ejecutar MySQL antes de usar `make init-db`.
- El archivo Makefile debe estar dentro de la carpeta `src`.
- Puedes ejecutar el proyecto completo desde Git Bash o desde VSCode.

Gestión de Testing

1. Introducción

Este documento aplica los conceptos del tema ISI-LAB, una aplicación web basada en Flask que consume APIs de música (Discogs y Last.fm) y gestiona datos de artistas y álbumes almacenados en MySQL. Se detallan los niveles de prueba, coberturas, técnicas de generación de valores, y metodologías aplicadas.

2. Niveles de Prueba Aplicados

- Pruebas Unitarias: Se aplican sobre funciones individuales como las que se conectan con las APIs.
- Pruebas de Integración: Validan la conexión de la aplicación con la base de datos MySQL.
- Pruebas de Sistema/Funcionales: Evalúan el comportamiento general de rutas y vistas Flask.

3. Cobertura y Oráculos

La cobertura alcanzada abarca principalmente las funciones de obtención de datos desde APIs, almacenamiento en base de datos y visualización en interfaz.

Oráculos utilizados:
- Visualización de respuestas correctas en consola (test_apis).
- Verificación de existencia y número de registros (test_database).
- Comparación de contenido HTML y redirecciones esperadas (test_funcional).

4. Valores de Prueba y Técnicas

Se han utilizado valores reales y relevantes como nombres de artistas y álbumes conocidos. Para las pruebas funcionales se han usado valores de equivalencia, simulando casos típicos de usuario.
Valores límite no aplican directamente, pero sí se consideran entradas no válidas (artistas inexistentes).
Técnicas utilizadas: clases de equivalencia, casos representativos, y combinación de entradas válidas.

5. Estrategias de Combinación

Dado que el sistema depende de datos de entrada externos (APIs), se aplica una combinación sencilla:
- Pairwise en pruebas manuales al combinar artista + álbum.
- Each-use implícito al recorrer listas de datos de prueba predefinidos en el script.

6. Metodología (TDD/BDD)

Se ha utilizado una aproximación **mixta**:
- En `test_funcional.py`, se usó BDD escribiendo los tests antes de desarrollar la vista.
- En `test_apis.py` y `test_database.py`, se siguió una estrategia posterior al desarrollo (validación).

7. Justificación de los Tests Existentes

- test_apis.py: valida que las APIs respondan y devuelvan información esperada.
- test_database.py: garantiza que los datos estén correctamente almacenados y se puedan consultar.
- test_funcional.py: verifica el flujo principal de navegación de la aplicación Flask y asegura que las rutas respondan con el contenido esperado.

8. Mejora Propuesta

- Incluir pruebas de mutación con pytest-mutation para validar la robustez del código.
- Añadir pruebas de inserción y eliminación de datos desde formularios.
- Automatizar la cobertura con herramientas como `coverage.py` para visualizar qué

líneas no se testean.
- Aplicar tests negativos (inputs inválidos, nombres vacíos, IDs inexistentes, etc).

9. Conclusión

El proyecto ISI-LAB cumple con una cobertura suficiente en pruebas funcionales e integración, siguiendo buenas prácticas de testing aplicadas desde la perspectiva académica y profesional. El uso de Makefile como herramienta de automatización mejora la reproducibilidad y la eficiencia del proceso.