Dario Bonino, Luigi De Russis

LAB 4 – REST SERVICES IN PYTHON

EXERCISE 1 – FLASK MUSIC PLAYER

Starting from the Flask Music Server developed in-class, design and implement the following additional functions:

- Album listing
  - Design a proper resource URL to support album retrieval
  - Extract the album collection from the track collection (either at creation-time or at every request)
  - Each album resource should at minimum contain the name of the album and the artist
- Track filtering
  - Design a proper resource URL to support filtering
  - Modify the code handling the tracks collection to apply filter parameters on the collection. Allowed parameters to filter correspond to the track metadata, i.e., artist, album, genre and title

EXERCISE 2 – FLASK MUSIC SERVER UI

Starting from the Flask Music Server UI developed in-class, and from outcomes of the Lab3, extend the Flask Music Server interface to provide different views for available tracks

- An Album view,
  - Shows available albums as thumbnails
  - When the album thumbnail is selected, the list of tracks belonging to the album shall be visualized and UI elements shall be provided to play/stop the single tracks
- A Playlist view,
  - Shows the current playlist
  - Allows to play, stop and skip tracks belonging to the playlist

EXERCISE 3 – OPTIONAL

Starting from the Flask Music Server developed in-class, design and implement the following additional functions:

- Multiple playlist
  - Design the needed set of resources and corresponding URLs
  - Implement the server code needed to support the functionality