First Assignment

Oriol Viñes, David Mola, Blai Gené, Pol Masip & Carlos Mazarico

Projecte Web - Curs 2024/25

1. Github public address

https://github.com/caarlos-04/projecteWeb.git

2. Important notes

2.1 How to use it

1. Clone the repository.
2. Run the Docker app.
3. Go to the “Selected-artist-track” branch to see the latest version of the project.
4. Introduce the command *docker compose up*.
5. Check the localhost 8000 in the URL.
6. To check the admin interface user: **edf** password: **12345678***

2.2. Design decisions

- **Models:**

The models are designed to reflect real-world relationships between key entities in a music streaming context: artists, albums, tracks, and listening history. The structure supports both data storage and querying needs for generating user statistics. We have created the following models:

`Artist`

- Represents a music artist.
- Fields: `name` (the artist's name).
- The `__str__` method returns the name for readability in admin or debug output.

`Album`

- Represents a music album.
- Fields: `title`, `artist` (linked to `Artist`), and `release_date`. Relationship: Many albums can belong to one artist (`ForeignKey` with `on_delete=models.CASCADE`).
- `related_name='albums'` allows easy reverse lookup (e.g., `artist.albums.all()`).

`Track`

- Represents a music track (song).
- Fields: `title`, `artist`, `album`, `duration_ms` (length in milliseconds), and `popularity`.
- `album` is optional (`null=True`, `blank=True`) to support singles or standalone tracks.
- If an album is deleted, the track remains but loses its album reference (`on_delete=models.SET_NULL`).
- `related_name='tracks'` allows reverse access like `artist.tracks.all()` or `album.tracks.all()`.

`ListeningHistory`

- Represents a record of when a track was listened to.
- Fields: `artist`, `track`, and `played_at` (timestamp of the play).
- Tracks user behavior for analytics or stats generation.
- `related_name='listening_history'` allows queries like `artist.listening_history.all()`.

2.3. 12factor Guidelines

The provided code adheres to several principles of the 12-Factor App methodology. The **Codebase** factor is satisfied, as the project follows a structured approach and is managed under version control. The **Dependencies** factor is also met, as the project uses a dependency manager file (‘pyproject.toml’). Additionally, the **Port Binding** principle is followed, since the project uses Django, which binds to a port to serve HTTP requests. However all these criteria are met, the project is still under development, it is not yet possible to fully follow all the 12-Factor guidelines.