Django Training - 1. Project basics, models, and queries

Materials

1. Python Poetry package manager
2. Django Overview and Installation (hint: don't use `pip install` use `poetry add`) You may use SQLite or PostgreSQL
3. Beginner Tutorial (Part 5: Testing and Part 6: Static Files are optional for now)
4. django-environ
5. Topics
    - Projects and apps
    - Models, Queries and ORM basics
    - Aggregation
    - Search
    - Model Relationship Examples

Task

For this task and the following ones, we'll try to build a music platform on which artists can sign up and put up their albums for sale

1. Create a personal repository on GitLab called `django-training`
2. Add ahmed.khaled@bld.ai and ahmed.wessam@bld.ai as maintainers to your repository (Project Information => Members => Invite members)
3. Create a Django project under the repository called `musicplatform`
4. Create an app `artists`
5. In `artists` Create an `Artist` model with the following fields (make the right assumptions)
    - Stage name
        - required
        - unique
        - must be used as the model's default ordering
    - Social link field
        - This field is to store each artist's social media profile (e.g. https://www.instagram.com/drake/)
        - What is a suitable field for this type of data?
        - This field is optional, but it shouldn't be null (why?)
6. Create an app `albums`
7. In `albums`, Create an `Album` model with the following fields (you might think it's an overkill creating a whole app for one model, for the sake of this practice we could have added the `Album` model to the `artists` app, but in the real world, think of how your app will scale if there are multiple types of artists, each with its own model, for example `GuestArtist`, and multiple types of albums, each with its own model)
    - an artist can have `0 or many` albums, an album must have an artist associated with it
    - name (if name is not provided, it should be called `New Album`)
    - creation datetime (the date when the album instance is created and stored in the database) (hint: respect the timezone)
    - release datetime **which cannot be empty** (hint: respect the timezone)
    - cost
        - required
        - use what you find suitable between `FloatField` and `DecimalField`
8. Create a `RESULTS.md` under `musicplatform/`
9. using `manage.py shell` (type the queries you used and their results in `RESULTS.md`):
    - create some artists
    - list down all artists
    - list down all artists sorted by name
    - list down all artists whose name starts with `a`
    - in 2 different ways, create some albums and assign them to any artists (hint: use `objects` manager and use the related object reference)
    - get the latest released album
    - get all albums released before today
    - get all albums released today or before but not after today
    - count the total number of albums (hint: count in an optimized manner)

- in 2 different ways, for each artist, list down all of his/her albums (hint: use `objects` manager and use the related object reference)
- list down all albums ordered by cost then by name (cost has the higher priority)

Guidelines

1. Use this gitignore template
2. List your environment variables (if any) in a `.env.example` file
3. Don't forget to run `manage.py makemigrations` and `python manage.py migrate`
4. You are allowed to refer to any articles, documentation source, questions on StackOverFlow
5. You are not allowed to consult other people through any medium of communication
6. Your grade will be affected if your code isn't clean, you should maintain code cleanliness as much as you can