Documentation :

Documentation technique :

L'application backend est développée en utilisant python sans framework.
Les principaux composants incluent des scripts python pour la gestion des données, l'API REST pour la communication avec le frontend, et les modèles de données.

L'application frontend est développée en utilisant le framework Angular.
Le frontend communique avec le backend à travers des requêtes HTTP vers l'API REST fournie par le backend.

Description de l'API :

Authentification (Login) :

URL : POST http://127.0.0.1:5000/login
Body :
```
{
   "name": "toto",
   "password": "toto"
}
```

Ajout d'une attraction :

URL : POST http://127.0.0.1:5000/attraction
Body :
```
{
  "attraction_id": 1,
  "nom": "test",
  "description": "test",
  "difficulte": 1,
  "visible": false
}
```

Ajout d'une critique :

URL : POST http://127.0.0.1:5000/attraction/critique
Body :
```
{
  "critique_id": 1,
  "name": "test",
  "text": "test",
  "rating": 1,
  "attraction_id": 1
}
```

Suppression d'une attraction :

URL : DELETE http://127.0.0.1:5000/attraction/{attraction_id}

Suppression d'une critique :

URL : DELETE http://127.0.0.1:5000/attraction/critique/{critique_id}

Récupération d'une attraction par son ID :

URL : GET http://127.0.0.1:5000/attraction/{attraction_id}

Récupération de toutes les attractions :

URL : GET http://127.0.0.1:5000/attraction

Récupération de toutes les attractions visibles :

URL : GET http://127.0.0.1:5000/visible_attraction

Récupération de toutes les critiques d'une attraction donnée :

URL : GET http://127.0.0.1:5000/attraction/{attraction_id}/critiques

Documentation fonctionnelle :

- Les admins inscrits dans la BDD peuvent se connecter et gérer les attractions (ajout, modification, rendre visible/invisible).

- Les visiteurs peuvent consulter la liste des attractions disponibles.

- Les visiteurs peuvent consulter la liste des critiques par attraction, ils peuvent également eux-mêmes en rajouter.

Mise en place du projet :

- Git clone le projet
- Faire une nouvelle branche
- Build le docker compose
- Lancer le docker compose
- Aller dans le container python
  - docker exec -it cours-api-1 sh
  - Lancer les scripts python dans l'ordre : init.py puis create.py
- Le projet est lancé