Université de Lorraine/ IUT Saint-Dié

BUT Informatique – Année 3

Maintenance Applicative

Documentation technique et fonctionnelle

Maintenance site web parc d'attractions

Auteur : Ewan Gailliegue
Année : 2025–2026

Table des matières

Chapitre 1

Présentation générale

L'application développée dans le cadre de ce projet est une plateforme web de gestion d'attractions intégrant un système de consultation et d'évaluation des attractions par les utilisateurs.

L'objectif principal du projet est de proposer une application simple, modulaire et évolutive permettant de gérer un catalogue d'attractions tout en offrant une expérience utilisateur intuitive et moderne.

L'application se décompose en deux grandes parties :

- **Une partie publique** destinée aux visiteurs, permettant :
  - de consulter la liste des attractions disponibles,
  - de filtrer et rechercher des attractions selon différents critères (nom, difficulté),
  - de visualiser les statistiques d'évaluation (moyenne des notes et nombre d'avis),
  - de déposer une critique comportant une note et un commentaire, avec possibilité d'anonymat.
- **Une partie administration** accessible via authentification, permettant :
  - de créer, modifier ou masquer des attractions,
  - de gérer la visibilité des attractions,
  - de modérer les critiques en activant ou désactivant leur affichage public.

L'application repose sur une architecture web moderne respectant une séparation claire des responsabilités :

- Un **frontend développé en Angular**, responsable de l'interface utilisateur, de la gestion des formulaires et de l'expérience interactive.
- Un **backend développé en Python avec Flask**, chargé de la logique métier, du traitement des requêtes HTTP et de l'accès aux données.
- Une base de données **SQLite**, utilisée pour le stockage persistant des attractions, utilisateurs et critiques.

L'ensemble est conteneurisé via **Docker Compose**, garantissant un environnement d'exécution reproductible et isolé. Le service **Nginx** joue le rôle de reverse proxy, assurant la communication entre le frontend et le backend, ainsi que la gestion du protocole HTTPS.

Cette architecture permet :

- une séparation nette entre interface et logique métier,
- une meilleure maintenabilité du code,
- une facilité de déploiement,

— une évolutivité future vers une architecture plus robuste si nécessaire.

Malgré la contrainte temporelle de 24 heures de développement, l'application intègre déjà un système d'avis optimisé (endpoint dédié aux statistiques), une interface administrateur fonctionnelle et une organisation modulaire facilitant les évolutions futures.

Chapitre 2

Documentation technique

2.1 Architecture

2.1.1 Frontend (Angular)

Technologies :
- Angular 17 (composants standalone),
- Angular Material (UI),
- HttpClient pour les appels API.

Organisation (simplifiée) :

```
src/app/
|-- accueil/              Page publique
|-- critique-dialog/      Popup avis
|-- admin/                Administration + modération
|-- login/                Authentification
|-- Service/              Services HTTP
|__ Interface/            Interfaces TypeScript
```

2.1.2 Backend (Flask + SQLite)

Technologies :
- Python / Flask,
- SQLite,
- JWT pour l'administration.

Organisation :

```
python/
|-- app.py
|-- controller/
|   |-- attraction.py
|   |-- critique.py
|   |__ auth.py
|-- request/
|   |__ request.py
```

```
|-- sql_file/
|   |-- init.sql
|   |__ create.sql
|__ parc.db
```

2.1.3 Infrastructure (Docker + Nginx)

L'application est déployée via Docker Compose :
- **web** : application Angular,
- **api** : application Flask,
- **nginx** : reverse proxy (HTTPS) et routage.

2.2 Fonctionnement technique

2.2.1 Consultation des attractions

1. Angular appelle l'API pour récupérer les attractions : `GET /attraction`.
2. Flask interroge SQLite et renvoie du JSON.
3. Le frontend applique filtres/tri et affiche les cartes.

2.2.2 Système d'avis

- Liste des critiques d'une attraction : `GET /attraction/{id}/critiques` (visibles uniquement).
- Ajout d'une critique : `POST /attraction/{id}/critiques`.
- Statistiques optimisées (moyenne + nombre) : `GET /attraction/{id}/stats`.

2.2.3 Modération

- Récupération de toutes les critiques : `GET /admin/critiques`.
- Changement de visibilité : `PATCH /admin/critique/{id}/visible`.

2.3 Sécurité

L'administration est protégée via un token JWT transmis dans l'en-tête :

```
Authorization: Bearer <token>
```

Chapitre 3

Documentation fonctionnelle

3.1 Partie publique

3.1.1 Page Accueil

L'utilisateur peut :
- consulter la liste des attractions,
- rechercher par mot-clé,
- filtrer par difficulté minimale,
- trier par nom ou difficulté,
- visualiser la moyenne des avis (5 étoiles style Google Reviews) et le nombre d'avis,
- ouvrir une popup pour lire/ajouter un avis.

3.1.2 Ajout d'un avis

L'avis contient :
- prénom (optionnel),
- nom (optionnel),
- note (obligatoire : 1 à 5),
- commentaire (obligatoire),
- mode anonyme (prénom/nom envoyés en `NULL`).

3.2 Partie administration

3.2.1 Gestion des attractions

L'administrateur peut :
- créer/modifier une attraction,
- modifier la visibilité,
- enregistrer des modifications en une fois (sauvegarde groupée).

3.2.2 Modération des avis

L'administrateur peut :

- voir toutes les critiques (visibles et invisibles),
- filtrer par recherche,
- afficher uniquement les critiques invisibles,
- activer/désactiver la visibilité via un interrupteur.

Chapitre 4

Schéma de base de données

4.1 Tables

4.1.1 Table `attraction`

Champ	Type	Description
attraction_id	INTEGER (PK)	Identifiant unique
nom	TEXT	Nom de l'attraction
description	TEXT	Description détaillée
difficulte	INTEGER	Niveau 1 à 5
visible	BOOLEAN	Visibilité publique (0/1)

4.1.2 Table `users`

Champ	Type	Description
users_id	INTEGER (PK)	Identifiant unique
name	TEXT	Nom d'utilisateur
password	TEXT	Mot de passe

4.1.3 Table `critique`

Champ	Type	Description
critique_id	INTEGER (PK)	Identifiant unique
attraction_id	INTEGER (FK)	Référence vers `attraction`
prenom	TEXT (NULL)	Prénom (optionnel)
nom	TEXT (NULL)	Nom (optionnel)
note	INTEGER	Note 1 à 5
commentaire	TEXT	Texte de l'avis
visible	BOOLEAN	Visible/modéré (0/1)

created_at	TEXT	Date de création

4.2 Relations

ATTRACTION (1) → (N) CRITIQUE

La suppression d'une attraction entraîne la suppression des critiques associées (ON DELETE CASCADE).

Chapitre 5

Améliorations (techniques et fonctionnelles)

5.1 Améliorations techniques

Amélioration	**Estimation**
Sécurisation de l'accès aux attractions visibles uniquement (filtrage côté backend et contrôle via API pour empêcher l'accès réseau aux attractions masquées)	4 heures
Mise en place complète du module de gestion des avis (modèle de données, relations base de données, validation des champs obligatoires, gestion du mode anonyme avec valeurs `NULL`)	4 heures
Ajout d'un système de modération robuste (statut de visibilité en base, requêtes filtrées, optimisation des performances sur les listes d'avis)	2 heures
Internationalisation de l'application avec i18n (externalisation des chaînes, fichiers de traduction FR/EN, configuration Angular i18n, tests multilangues)	3 heures
Amélioration de la qualité du code et maintenabilité (refactorisation des composants, factorisation des services, gestion centralisée des erreurs, commentaires techniques)	2 heures

5.2 Améliorations fonctionnelles

Amélioration	**Estimation**

Amélioration ergonomique de l'interface utilisateur (SCSS, harmonisation des couleurs, amélioration des cartes attractions, meilleure lisibilité des avis et système d'étoiles type Google Reviews)	4 heures
Ajout d'indicateurs dynamiques (affichage temps réel de la moyenne des notes, nombre d'avis, messages utilisateur en cas d'absence d'avis ou d'attraction visible)	2 heures
Optimisation de l'expérience d'administration (sauvegarde groupée sécurisée, confirmation visuelle des actions, filtres avancés en modération)	3 heures

5.3 Estimation globale

L'ensemble de ces améliorations représente environ **24 heures** de développement supplémentaires.

Cette estimation comprend :

- le développement technique,
- les tests unitaires et fonctionnels,
- les vérifications multi-navigateurs,
- la mise à jour de la documentation technique et fonctionnelle,
- la validation finale.

Ces améliorations renforcent la sécurité, la maintenabilité et l'expérience utilisateur de l'application, tout en assurant la conformité avec le cahier des charges initial.

Chapitre 6

Conclusion

Ce projet de maintenance du site web parcattraction a permis de mettre en oeuvre une architecture complète, cohérente et moderne, répondant aux exigences fonctionnelles et techniques du cahier des charges.

Malgré une limite de temps importante (24 heures de développement), l'application intègre :

- une séparation claire entre frontend (Angular) et backend (Flask),
- une gestion sécurisée de l'administration via JWT,
- un système de critiques structuré avec modération,
- une base de données relationnelle cohérente respectant l'intégrité référentielle,
- un déploiement conteneurisé garantissant la reproductibilité de l'environnement.

Au-delà de l'implémentation fonctionnelle, une attention particulière a été portée à :

- la maintenabilité du code (organisation modulaire),
- l'optimisation des requêtes (endpoint dédié aux statistiques),
- la sécurité des accès (filtrage des ressources visibles côté backend),
- la qualité de l'expérience utilisateur.

Les améliorations proposées démontrent que l'application possède un potentiel d'évolution important. Elle pourrait être enrichie par :

- une internationalisation complète,
- une amélioration continue de l'ergonomie,
- une montée en charge vers une base de données plus robuste,
- une industrialisation plus poussée (CI/CD, tests automatisés).

Ce projet démontre la capacité à concevoir, maintenir et faire évoluer une application web complète en respectant des contraintes techniques, fonctionnelles et organisationnelles.

Il met en évidence la maîtrise des technologies frontend et backend, la compréhension des enjeux de sécurité et d'architecture, ainsi que l'importance de la documentation dans un contexte professionnel.

En conclusion, l'application constitue une base solide, fonctionnelle et évolutive, répondant aux objectifs fixés tout en ouvrant des perspectives d'amélioration réalistes.