Application Parc d'Attractions

Documentation Complète du Projet

Auteur	Zaynab RIFI
Projet	Système de gestion de parc d'attractions
Date	Février 2026
Technologies	Angular 19, Python Flask, MariaDB, Docker

17 février 2026

Table des matières

1 Documentation Technique **4**

1.1 Vue d'ensemble 4

1.1.1 Description 4

1.1.2 Objectifs 4

1.2 Architecture 4

1.2.1 Architecture générale 4

1.2.2 Architecture 3-Tiers 5

1.3 Technologies utilisées 5

1.3.1 Backend 5

1.3.2 Frontend 5

1.3.3 Infrastructure 5

1.4 Structure du projet 6

1.5 Backend — API Flask 6

1.5.1 Contrôleurs 6

1.5.2 Endpoints — Attractions 7

1.5.3 Endpoints — Critiques et Authentification 7

1.6 Frontend — Angular 7

1.6.1 Interfaces TypeScript 7

1.6.2 Routing Angular 8

1.7 Sécurité 8

1.8 Installation 9

2 Documentation Fonctionnelle **10**

2.1 Rôles utilisateurs et accès 10

2.2 Interface Visiteur 10

2.2.1 Page d'accueil 10

2.2.2 Consulter les avis 10

2.2.3 Déposer un avis 10

2.3 Interface Administrateur 11

2.3.1 Connexion 11

2.3.2 Vue d'ensemble de la page Admin 11

2.3.3 Créer une attraction 11

2.3.4 Modifier une attraction 11
2.3.5 Gérer la visibilité 11
2.3.6 Supprimer une attraction 12
2.4 FAQ 12

3 Schéma de Base de Données 13
3.1 Vue d'ensemble 13
3.2 Diagramme Entité-Association 13
3.3 Table : attraction 14
3.4 Table : users 14
3.5 Table : critique 15
3.6 Requêtes SQL principales 15
3.7 Données de test 16

4 Améliorations Possibles 18
4.1 Grille d'estimation 18
4.2 Sécurité — Priorité HAUTE 18
4.2.1 1. Hashage des mots de passe (BCrypt) — 2 jours 18
4.2.2 2. Validation et sanitisation des entrées — 3 jours 18
4.2.3 3. Certificats SSL valides (Let's Encrypt) — 1 jour 18
4.3 Fonctionnalités — Priorité MOYENNE 18
4.3.1 4. Pagination des résultats — 2 jours 18
4.3.2 5. Recherche et filtres — 4 jours 19
4.3.3 6. Modification / Suppression de critiques — 5 jours 19
4.3.4 7. Upload d'images pour les attractions — 6 jours 19
4.3.5 8. Modération des critiques — 5 jours 19
4.3.6 9. Réponses aux critiques (Admin) — 3 jours 19
4.4 Performance et Technique — Priorité MOYENNE 19
4.4.1 10. Cache Redis — 4 jours 19
4.4.2 11. Tests automatisés — 10 jours 19
4.4.3 12. Pipeline CI/CD — 3 jours 19
4.4.4 13. Monitoring et Logs — 5 jours 19
4.4.5 14. Documentation API Swagger — 2 jours 19
4.5 UX/UI — Priorité BASSE 20
4.5.1 15. Dark Mode — 3 jours 20
4.5.2 16. Accessibilité WCAG 2.1 — 5 jours 20
4.5.3 17. Progressive Web App (PWA) — 4 jours 20
4.5.4 18. Multi-langue complet (i18n) — 5 jours 20
4.6 Nouvelles Fonctionnalités 20
4.6.1 19. Système de réservation — 15 jours 20

4.6.2 20. Analytics et statistiques — 8 jours 20
4.6.3 21. Notifications en temps réel — 6 jours 20
4.6.4 22. Programme de fidélité — 12 jours 20
4.7 Récapitulatif 20
4.8 Planning proposé 22

1. Documentation Technique

Ce chapitre décrit comment l'application fonctionne d'un point de vue technique : architecture, technologies, structure du code, API et déploiement.

1.1 Vue d'ensemble

1.1.1 Description

L'application web de gestion de parc d'attractions permet :

— L'affichage des attractions disponibles pour les visiteurs
— La gestion administrative complète des attractions
— Un système de critiques et d'avis des visiteurs
— La gestion de la visibilité des attractions (maintenance, fermeture)

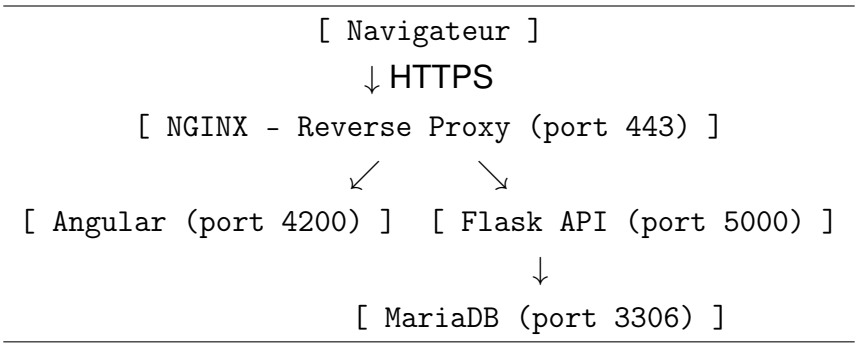
1.1.2 Objectifs

— Fournir une interface visiteur intuitive et moderne
— Permettre une administration simple et efficace
— Collecter et afficher les avis des visiteurs
— Contrôler finement la visibilité des attractions

1.2 Architecture

1.2.1 Architecture générale

L'application suit une architecture **multi-tiers** avec 4 composants principaux déployés via Docker Compose :



1.2.2 Architecture 3-Tiers

Tier	Technologie	Rôle
Tier 1	Angular (Frontend)	Interface utilisateur, gestion des routes, appels API REST
Tier 2	Flask (Backend)	API REST, contrôleurs métier, authentification, validation
Tier 3	MariaDB	Stockage persistant, relations entre tables, contraintes

1.3 Technologies utilisées

1.3.1 Backend

Technologie	Version	Rôle
Python	3.10	Langage principal du backend
Flask	3.x	Framework web léger
Flask-CORS	4.x	Gestion des requêtes Cross-Origin
MariaDB Connector	Latest	Connexion à la base de données
JWT	Latest	Authentification par tokens

1.3.2 Frontend

Technologie	Version	Rôle
Angular	19.x	Framework frontend SPA
TypeScript	5.x	Langage fortement typé
Angular Material	19.x	Bibliothèque de composants UI
RxJS	7.x	Programmation réactive (Observables)
SCSS	Latest	Préprocesseur CSS

1.3.3 Infrastructure

Technologie	Version	Rôle
Docker	Latest	Conteneurisation des services
Docker Compose	Latest	Orchestration multi-conteneurs
Nginx	Latest	Reverse proxy et SSL
MariaDB	11.7	Système de gestion de base de données

1.4 Structure du projet

Listing 1.1 – Arborescence du projet

```
parcattraction-RIFI-Zaynab/
|-- docker-compose.yml
|-- docs/
|-- python/                    (Backend Flask)
|   |-- app.py                 Point d'entree de l'API
|   |-- init.py
|   |-- controller/
|   |   |-- attraction.py
|   |   |-- critique.py
|   |   `-- auth/auth.py
|   |-- request/request.py
|   `-- sql_file/
|       |-- create.sql
|       `-- init.sql
`-- parc/src/app/              (Frontend Angular)
    |-- Interface/
    |   |-- attraction.interface.ts
    |   `-- critique.interface.ts
    |-- Service/
    |   |-- attraction.service.ts
    |   |-- critique.service.ts
    |   `-- auth.service.ts
    |-- accueil/
    |-- admin/
    `-- critique-modal/
```

1.5 Backend — API Flask

1.5.1 Contrôleurs

L'application suit le pattern **MVC** avec trois contrôleurs principaux.

Listing 1.2 – Méthodes du controller Attraction

```
class Attraction:
    def get_all_attraction(self, visible_only=True):
        """Recupere toutes les attractions (avec filtre)"""
    def get_attraction_by_id(self, id):
        """Recupere une attraction par son ID"""
    def add_attraction(self, nom, description, difficulte,
```

```
        visible):
            """Ajoute une nouvelle attraction"""
    def update_attraction(self, id, nom, description, difficulte
        , visible):
            """Met a jour une attraction existante"""
    def delete_attraction(self, id):
            """Supprime une attraction"""
```

Listing 1.3 – Méthodes du controller Critique

```
class Critique:
    def get_critiques_by_attraction(self, attraction_id):
        """Recupere toutes les critiques d'une attraction"""
    def add_critique(self, attraction_id, texte, note, prenom,
        nom, anonyme):
        """Ajoute une nouvelle critique"""
    def get_critique_stats(self, attraction_id):
        """Calcule les statistiques (moyenne, nombre)"""
```

1.5.2 Endpoints — Attractions

Méthode	Endpoint	Description	Auth
GET	`/attraction`	Liste des attractions visibles	Non
GET	`/attraction?admin=true`	Toutes les attractions	Oui
GET	`/attraction/<id>`	Détails d'une attraction	Non
POST	`/attraction`	Créer une attraction	Oui
PUT	`/attraction/<id>`	Modifier une attraction	Oui
DELETE	`/attraction/<id>`	Supprimer une attraction	Oui

1.5.3 Endpoints — Critiques et Authentification

Méthode	Endpoint	Description	Auth
GET	`/attraction/<id>/critique`	Critiques d'une attraction	Non
POST	`/attraction/<id>/critiqu`	Ajouter une critique	Non
GET	`/attraction/<id>/critique/stats`	Statistiques des avis	Non
POST	`/login`	Connexion (retourne token)	Non

1.6 Frontend — Angular

1.6.1 Interfaces TypeScript

Listing 1.4 – Interfaces TypeScript

```
export interface Attraction {
  attraction_id?: number;
  nom: string;
  description: string;
  difficulte: number;
  visible: boolean;
}

export interface Critique {
  critique_id?: number;
  attraction_id: number;
  texte: string;
  note: number;          // 1 a 5
  prenom: string;
  nom: string;
  anonyme: boolean;
  date_creation?: string;
}
```

1.6.2 Routing Angular

— `/` → redirige vers `/accueil`
— `/accueil` → AccueilComponent (page visiteur)
— `/login` → LoginComponent (authentification admin)
— `/admin` → AdminComponent (protégé par AuthGuard)

1.7 Sécurité

Mesure	Description	Statut
JWT Tokens	Authentification admin par tokens	OK
Protection des routes	Guard Angular côté frontend	OK
CORS	Restriction des origines autorisées	OK
HTTPS	Chiffrement via NGINX + SSL	OK
Requêtes paramétrées	Protection contre injection SQL	OK
Hashage mots de passe	BCrypt pour les admins	À faire
Rate Limiting	Limite de requêtes par IP	À faire

1.8 Installation

1. `git clone <repository-url>` puis `cd parcattraction-RIFI-Zaynab`
2. `docker compose up -d` (démarre tous les services)
3. Créer les tables via MariaDB (première fois uniquement)
4. Accéder au frontend : `http://localhost:4200`
5. Accéder à l'API : `http://localhost:5000`

2. Documentation Fonctionnelle

Ce chapitre décrit comment utiliser l'application, étape par étape, pour les visiteurs et les administrateurs.

2.1 Rôles utilisateurs et accès

Rôle	Accès	Authentification
Visiteur	Consultation des attractions + dépôt d'avis	Non requise
Administrateur	Gestion complète (CRUD, visibilité)	Login / Mot de passe

URLs d'accès :

— Page Visiteur : `http://localhost:4200/accueil`
— Connexion Admin : `http://localhost:4200/login`
— Page Administration : `http://localhost:4200/admin`

2.2 Interface Visiteur

2.2.1 Page d'accueil

La page d'accueil affiche uniquement les attractions **visibles** (non masquées par l'administrateur). Chaque attraction est présentée sous forme de carte avec :

— Le nom et la description de l'attraction
— Le niveau de difficulté (de 1 à 5)
— La note moyenne des avis et le nombre total d'avis
— Deux boutons : **Voir les avis** et **Donner mon avis**

2.2.2 Consulter les avis

1. Cliquer sur **Voir les avis** sur une carte d'attraction
2. Un panneau s'ouvre et affiche la liste des avis
3. Chaque avis montre : note (étoiles), auteur (ou *Anonyme*), date et texte

2.2.3 Déposer un avis

1. Cliquer sur **Donner mon avis**
2. Une fenêtre modale s'ouvre avec un formulaire

3. Remplir les champs :
   — **Note** (obligatoire) : cliquer sur 1 à 5 étoiles
   — **Commentaire** (obligatoire) : minimum 10 caractères
   — **Prénom** / **Nom** : requis si non anonyme
   — **Publier en anonyme** : case à cocher optionnelle
4. Cliquer sur **Publier**
5. L'avis apparaît immédiatement, les statistiques sont mises à jour

2.3 Interface Administrateur

2.3.1 Connexion

1. Accéder à `http://localhost:4200/login`
2. Saisir les identifiants : **admin** / **admin123**
3. Cliquer sur **Se connecter**
4. Redirection automatique vers la page d'administration

2.3.2 Vue d'ensemble de la page Admin

La page admin affiche **toutes les attractions** (visibles ET masquées).

Affichage	Signification
Bordure bleue	Attraction visible par les visiteurs
Bordure rouge	Attraction masquée (maintenance, fermeture)

2.3.3 Créer une attraction

1. Cliquer sur **Nouvelle Attraction**
2. Remplir le formulaire : Nom, Description, Difficulté (1–5), Visible (coché par défaut)
3. Cliquer sur **Enregistrer**

2.3.4 Modifier une attraction

1. Cliquer sur l'attraction à modifier
2. Le formulaire se remplit avec les données actuelles
3. Modifier les champs souhaités et cliquer sur **Enregistrer**

2.3.5 Gérer la visibilité

1. Sélectionner l'attraction concernée
2. Basculer le toggle **Visible par les visiteurs**
3. Cliquer sur **Enregistrer**

Cas d'usage : attraction en maintenance, fermeture temporaire, test avant ouverture officielle.

2.3.6 Supprimer une attraction

1. Cliquer sur le bouton **Supprimer**
2. Confirmer la suppression

Note : La suppression est irréversible. Toutes les critiques associées seront également supprimées (suppression en cascade).

2.4 FAQ

Faut-il un compte pour laisser un avis ?

Non, n'importe quel visiteur peut déposer un avis, même en anonyme.

Peut-on modifier un avis après publication ?

Non, actuellement les avis ne sont pas modifiables (prévu dans les améliorations futures).

Pourquoi certaines attractions n'apparaissent pas ?

Elles sont temporairement masquées par un administrateur (maintenance, fermeture).

Les critiques sont-elles supprimées si on masque une attraction ?

Non, elles sont conservées et réapparaissent si l'attraction est remise visible.

L'application fonctionne-t-elle sur mobile ?

Oui, l'interface est responsive et s'adapte à toutes les tailles d'écran.

3. Schéma de Base de Données

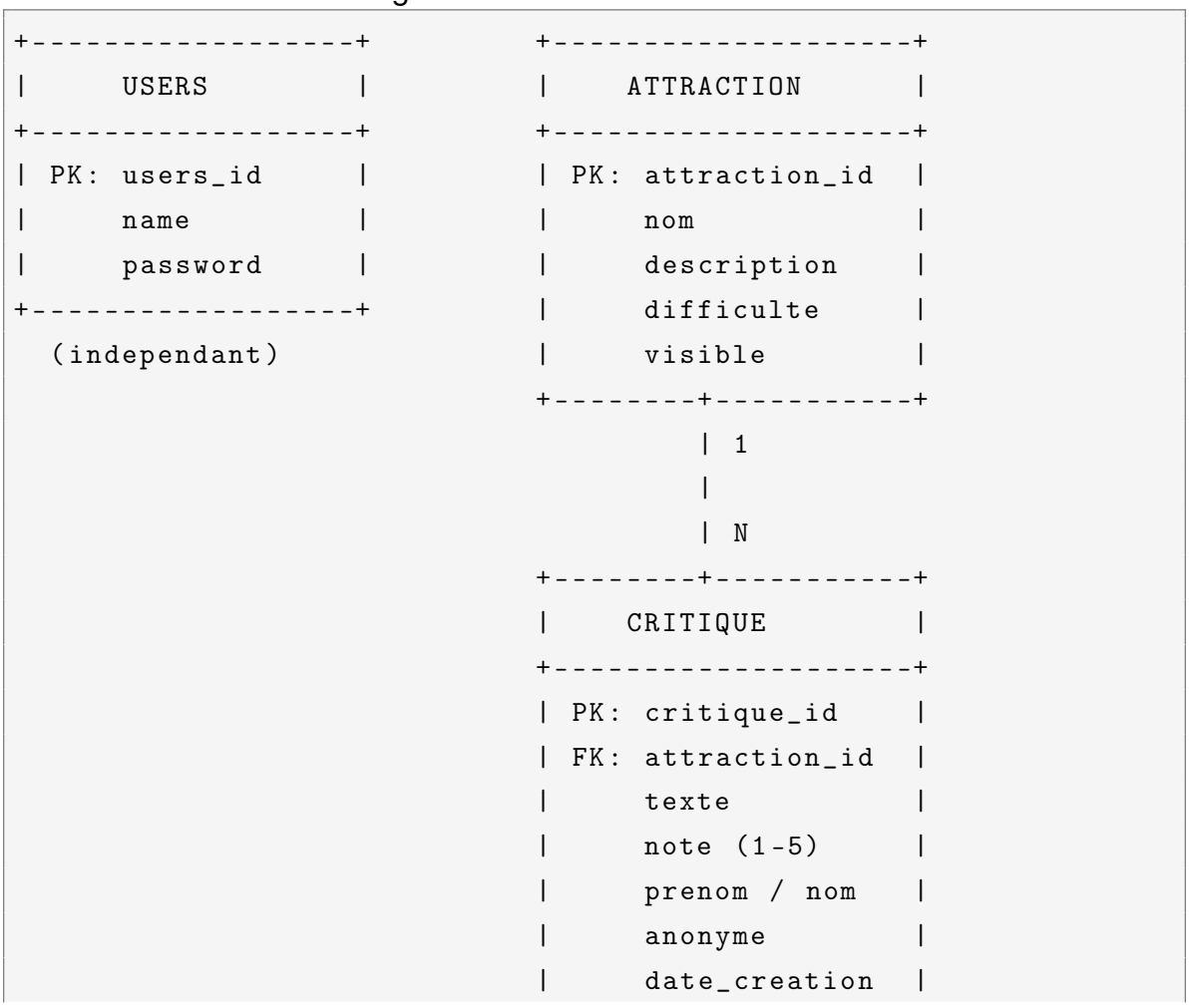
Ce chapitre présente le schéma complet de la base de données MariaDB 11.7.

3.1 Vue d'ensemble

Table	Description	Lignes (test)
`attraction`	Informations sur les attractions du parc	5
`users`	Comptes administrateurs	2
`critique`	Avis et critiques des visiteurs	4+

3.2 Diagramme Entité-Association

Listing 3.1 – Schéma entité-association



```
                                    +--------------------+
```

— **ATTRACTION** ↔ **CRITIQUE** : Relation 1–N (une attraction peut avoir plusieurs critiques)
— Suppression en **CASCADE** : si une attraction est supprimée, toutes ses critiques le sont aussi
— **USERS** : table indépendante, utilisée pour l'authentification admin

3.3 Table : attraction

Colonne	Type	Null	Défaut	Description
attraction_id	INT	NON	AUTO	Identifiant unique (PK)
nom	VARCHAR(25	NON	—	Nom de l'attraction
description	VARCHAR(255)	NON	—	Description courte
difficulte	INT	OUI	NULL	Niveau de 1 à 5
visible	BOOL	OUI	TRUE	Visibilité visiteurs

Listing 3.2 – Création de la table attraction

```
CREATE TABLE attraction (
    attraction_id INT AUTO_INCREMENT,
    PRIMARY KEY(attraction_id),
    nom           VARCHAR(255) NOT NULL,
    description   VARCHAR(255) NOT NULL,
    difficulte    INT,
    visible       BOOL DEFAULT TRUE
);
```

3.4 Table : users

Listing 3.3 – Création de la table users

```
CREATE TABLE users (
    users_id INT AUTO_INCREMENT,
    PRIMARY KEY(users_id),
    name     VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Note : En version MVP, les mots de passe sont en clair. En production, utiliser BCrypt.

3.5 Table : critique

Colonne	Type	Null	Défaut	Description
`critique_id`	INT	NON	AUTO	Identifiant unique (PK)
`attraction_id`	INT	NON	—	Référence attraction (FK)
`texte`	TEXT	NON	—	Contenu de l'avis
`note`	INT	NON	—	Note de 1 à 5 (CHECK)
`prenom`	VARCHAR(100)	OUI	NULL	Prénom auteur
`nom`	VARCHAR(10(	OUI	NULL	Nom auteur
`anonyme`	BOOL	OUI	FALSE	Publication anonyme
`date_creation`	DATETIME	OUI	NOW()	Date automatique

Listing 3.4 – Création de la table critique

```
CREATE TABLE critique (
    critique_id   INT AUTO_INCREMENT,
    PRIMARY KEY(critique_id),
    attraction_id INT      NOT NULL,
    texte         TEXT     NOT NULL,
    note          INT      NOT NULL CHECK (note >= 1 AND note <=
        5),
    prenom        VARCHAR(100),
    nom           VARCHAR(100),
    anonyme       BOOL     DEFAULT FALSE,
    date_creation DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (attraction_id)
        REFERENCES attraction(attraction_id)
        ON DELETE CASCADE
);
```

3.6 Requêtes SQL principales

Listing 3.5 – Requêtes principales de l'application

```
-- Attractions visibles (page visiteur)
SELECT * FROM attraction WHERE visible = 1;

-- Toutes les attractions (admin)
SELECT * FROM attraction;

-- Critiques d'une attraction
SELECT * FROM critique
```

```
WHERE attraction_id = 1
ORDER BY date_creation DESC;

-- Statistiques d'une attraction
SELECT a.nom,
       AVG(c.note)           AS note_moyenne,
       COUNT(c.critique_id)  AS nb_avis
FROM attraction a
LEFT JOIN critique c ON a.attraction_id = c.attraction_id
WHERE a.attraction_id = 1
GROUP BY a.attraction_id;

-- Masquer une attraction
UPDATE attraction SET visible = FALSE WHERE attraction_id = 5;

-- Supprimer une attraction (+ critiques en CASCADE)
DELETE FROM attraction WHERE attraction_id = 5;
```

3.7 Données de test

Listing 3.6 – Jeu de données de test

```
-- Attractions (4 visibles + 1 masquee)
INSERT INTO attraction (nom, description, difficulte, visible)
   VALUES
  ('Silver Star',          'Montagne russe grande vitesse', 3,
     1),
  ('Montagne 8',           'Montagne russe familiale',      4,
     1),
  ('La Tour de la Terreur','Chute libre vertigineuse',      5,
     1),
  ('Carrousel Magique',    'Manege pour enfants',           1,
     1),
  ('Grand Huit Extreme',   'En maintenance',                5,
     0);

-- Admins
INSERT INTO users (name, password)
  VALUES ('admin', 'admin123'), ('toto', 'toto');

-- Critiques de test
INSERT INTO critique
```

```
(attraction_id, texte, note, prenom, nom, anonyme) VALUES
(1, 'Attraction incroyable !',          5, 'Jean',  'Dupont',
   0),
(1, 'Un peu rapide mais bien concu.',   4, '',      '',
   1),
(2, 'Parfait pour la famille !',        5, 'Marie', 'Martin',
   0),
(2, 'Bien mais un peu d attente.',      3, '',      '',
   1);
```

4. Améliorations Possibles

Ce chapitre propose 22 améliorations classées par priorité, avec une estimation en jours-homme pour chacune.

4.1 Grille d'estimation

Complexité	Jours	Description
Triviale	0.5 – 1	Configuration simple, modification mineure
Faible	1 – 3	Fonctionnalité simple, peu de dépendances
Moyenne	3 – 7	Fonctionnalité complète avec tests
Haute	7 – 15	Fonctionnalité complexe, plusieurs composants
Très haute	15 – 30	Refonte majeure, architecture complexe

4.2 Sécurité — Priorité HAUTE

4.2.1 1. Hashage des mots de passe (BCrypt) — 2 jours

Les mots de passe sont actuellement stockés en clair. Il faut utiliser BCrypt pour hasher à la création et vérifier le hash à la connexion. Bénéfices : sécurité renforcée, conformité RGPD, protection contre les fuites.

4.2.2 2. Validation et sanitisation des entrées — 3 jours

Renforcer la validation des données côté backend pour prévenir les injections SQL et les attaques XSS. Valider strictement tous les paramètres API.

4.2.3 3. Certificats SSL valides (Let's Encrypt) — 1 jour

Remplacer les certificats auto-signés par des certificats Let's Encrypt valides, avec Certbot et renouvellement automatique.

4.3 Fonctionnalités — Priorité MOYENNE

4.3.1 4. Pagination des résultats — 2 jours

Charger les attractions par pages (10 par page) au lieu de tout charger d'un coup. Améliore les performances et l'expérience mobile.

4.3.2 5. Recherche et filtres — 4 jours

Permettre de rechercher par nom et filtrer par difficulté, note et ordre de tri. Interface avec barre de recherche et filtres latéraux.

4.3.3 6. Modification / Suppression de critiques — 5 jours

Permettre à l'auteur d'un avis de le modifier ou supprimer via un système de session ou cookie. Limitation temporelle (modifiable dans les 24h).

4.3.4 7. Upload d'images pour les attractions — 6 jours

Ajouter des photos aux attractions avec galerie. Stockage sur disque ou S3, génération automatique de miniatures.

4.3.5 8. Modération des critiques — 5 jours

Permettre aux visiteurs de signaler des avis inappropriés et aux admins de valider ou rejeter les critiques signalées.

4.3.6 9. Réponses aux critiques (Admin) — 3 jours

Permettre aux administrateurs de répondre officiellement aux avis des visiteurs, avec affichage sous les critiques (badge *Réponse officielle*).

4.4 Performance et Technique — Priorité MOYENNE

4.4.1 10. Cache Redis — 4 jours

Mettre en cache les résultats fréquents (liste d'attractions, statistiques) pour réduire la charge sur MariaDB. Bénéfice estimé : temps de réponse réduit de 80%.

4.4.2 11. Tests automatisés — 10 jours

Créer une suite de tests : Backend (pytest), Frontend (Jasmine/Jest), E2E (Cypress). Objectif : couverture de code supérieure ou égale à 80%.

4.4.3 12. Pipeline CI/CD — 3 jours

Automatiser le build, les tests et le déploiement avec GitHub Actions : lint, tests, build Docker, déploiement automatique (staging) et manuel (production).

4.4.4 13. Monitoring et Logs — 5 jours

Mettre en place Prometheus + Grafana pour les métriques et ELK Stack pour la centralisation des logs. Alertes Slack ou Email.

4.4.5 14. Documentation API Swagger — 2 jours

Générer automatiquement la documentation de l'API avec Flasgger et une interface Swagger UI interactive.

4.5 UX/UI — Priorité BASSE

4.5.1 15. Dark Mode — 3 jours

Thème sombre avec switch Light/Dark, palette de couleurs adaptée et sauvegarde de la préférence utilisateur.

4.5.2 16. Accessibilité WCAG 2.1 — 5 jours

Attributs ARIA, navigation clavier, contraste AAA, support lecteurs d'écran. Rendre l'application accessible aux personnes en situation de handicap.

4.5.3 17. Progressive Web App (PWA) — 4 jours

Rendre l'application installable sur mobile avec Service Worker, mode hors-ligne (lecture) et notifications push.

4.5.4 18. Multi-langue complet (i18n) — 5 jours

Finaliser le système i18n Angular pour supporter le français, l'anglais, l'espagnol et l'allemand. Switch de langue dans l'interface.

4.6 Nouvelles Fonctionnalités

4.6.1 19. Système de réservation — 15 jours

Permettre aux visiteurs de réserver un créneau horaire pour une attraction. Calendrier de disponibilités, QR Code de confirmation, email automatique.

4.6.2 20. Analytics et statistiques — 8 jours

Dashboard administrateur avec graphiques interactifs : attractions les plus populaires, évolution des notes, taux de conversion. Export PDF/Excel.

4.6.3 21. Notifications en temps réel — 6 jours

Système WebSocket pour notifier les visiteurs : attraction ouverte/fermée, réponse à un avis, nouveauté. Support notifications push (PWA).

4.6.4 22. Programme de fidélité — 12 jours

Compte visiteur avec système de points, niveaux (Bronze, Argent, Or) et récompenses (réductions, accès prioritaire, accès VIP).

4.7 Récapitulatif

#	Amélioration	Priorité	Jours
1	Hashage mots de passe (BCrypt)	HAUTE	2

#	Amélioration	Priorité	Jours
2	Validation des entrées	HAUTE	3
3	Certificats SSL valides	HAUTE	1
4	Pagination des résultats	MOYENNE	2
5	Recherche et filtres	MOYENNE	4
6	Modification / Suppression critiques	MOYENNE	5
7	Upload d'images	MOYENNE	6
8	Modération des critiques	MOYENNE	5
9	Réponses aux critiques	BASSE	3
10	Cache Redis	MOYENNE	4
11	Tests automatisés	MOYENNE	10
12	Pipeline CI/CD	MOYENNE	3
13	Monitoring et Logs	BASSE	5
14	Documentation API Swagger	BASSE	2
15	Dark Mode	BASSE	3
16	Accessibilité WCAG 2.1	MOYENNE	5
17	Progressive Web App (PWA)	BASSE	4
18	Multi-langue complet (i18n)	MOYENNE	5
19	Système de réservation	MOY-HAUTE	15
20	Analytics et statistiques	MOYENNE	8
21	Notifications en temps réel	MOYENNE	6
22	Programme de fidélité	BASSE	12
TOTAL			**~113 j**

TABLE 4.1: Récapitulatif des améliorations

4.8 Planning proposé

Phase	Nom	Contenu	Durée
1	Sécurité & Stabilité	Hashage, validation, SSL, tests auto	16 jours
2	UX Critiques	Pagination, recherche, images	17 jours
3	Fonctions Sociales	Modif critiques, réponses, modération	13 jours
4	Performance DevOps	Redis, CI/CD, doc API	9 jours
5	Design & Accessibilité	Accessibilité, i18n, Dark Mode	15 jours
6	Fonctions Avancées	Réservation, analytics, notifications	29 jours
Total estimé			**~6 mois**

Recommandations prioritaires :

1. **Phase 1 en priorité** : sécuriser l'application (BCrypt, validation, SSL)
2. **Phase 2 ensuite** : améliorer l'UX (pagination, recherche, images)
3. **Tests automatisés** : essentiels pour maintenir la qualité long terme