CAHIER DES CHARGES PROJETS WEB

Projet	Site web projet
Personne en charge	Nathan Vachin, Tony Fournet

Objectifs

- L’application doit répertorier des établissements (nom, adresse, localisation).
- L’utilisateur non inscrit peut voir une carte et filtrer les établissements par type (magasin, restaurant).
- Chaque établissement peut recevoir une note de 1 à 5, comme sur Google.
- Les notes sont regroupées pour afficher une moyenne des avis de l’établissement.

1. Projet – Structure et Informations Générales

- **2. Public**
  - **Utilisateur non enregistré** :
    - Peut voir la carte avec des établissements autour de lui.
    - Peut utiliser une barre de recherche et un filtre pour affiner les résultats.
    - L'affichage doit être adapté à l'appareil (mobile, tablette, PC).
  - **Utilisateur enregistré** :
    - En plus des fonctionnalités d’un utilisateur non enregistré, l'utilisateur peut :
    - Ajouter des avis et des notes pour chaque établissement.
    - Voir la moyenne des notes des établissements.
    - Affichage et gestion des informations adaptées selon l'appareil (responsive design).

2. Backoffice

- **Admin**

- **Rôle de l'admin** :
  - Modifier, supprimer, et ajouter des établissements dans la base de données.
  - L'admin pourra accéder à une interface de gestion via un back-office sécurisé.
  - Une page de connexion est nécessaire pour l'accès à l'admin.

3. Architecture du Projet (MVC)

- **Backend (Python)**
  - Le backend sera en Python pour la gestion des données et du traitement des actions des utilisateurs.
- **Frontend (HTML, CSS, JS)**
  - Le frontend sera développé avec HTML, CSS, et JavaScript pour l'interaction avec l'utilisateur.
- **Architecture MVC**
  - **Model (M)** : Représente les objets du projet, tels que l'établissement et la catégorie.
  - **View (V)** : La partie visuelle de l'application, ce que l'utilisateur voit (interface graphique).
  - **Controller (C)** : La logique de l'application, qui traite les actions des utilisateurs et met à jour les modèles ou vues.

4. Fonctionnalités

- **Public**
  - L'application répertorie des établissements (Nom, Adresse).
  - L'utilisateur peut voir la carte autour de lui grâce à l'API de géolocalisation et un système de filtrage des établissements.
  - L'utilisateur peut filtrer les établissements par catégorie (par exemple, magasin, restaurant).
  - Pour afficher un établissement sur la carte, des coordonnées GPS sont nécessaires (longitude, latitude).
- **Utilisateur enregistré**
  - Possibilité de laisser un avis et une note sur les établissements (note de 1 à 5, comme sur Google).
  - Affichage de la moyenne des notes laissées par les utilisateurs.
- **Backoffice/Administration**
  - L'admin peut effectuer les opérations suivantes :
    - Ajouter des établissements.
    - Modifier les informations existantes d'un établissement.
    - Supprimer des établissements.
    - Page de connexion sécurisée pour l'accès à l'interface d'administration.

5. Technologies et Outils

- **API pour la carte** :

    - Utilisation d'OpenStreetMap via Leaflet pour l'affichage de la carte.
    - Utilisation d'API de géolocalisation pour déterminer la position de l'utilisateur.
- **Frontend** :
    - **CSS** : Utilisation de templates CSS prêts à l'emploi (par exemple, Bootstrap), avec une distinction entre le public et le backoffice pour une meilleure gestion de l'interface.
    - **Responsivité** : L'affichage doit être adapté pour différentes tailles d'écrans (PC, mobile, tablette).
- **Backend** :
    - Python pour gérer la logique côté serveur (gestion des utilisateurs, des établissements, etc.).
    - **Base de données** : Elle contiendra les informations sur les établissements et les avis des utilisateurs. Elle est à modéliser avec un diagramme UML.
- **Gestion des données (CRUD)** :
    - **CRUD** : Chaque établissement pourra être créé, lu, mis à jour et supprimé dans la base de données.

6. Base de Données et Modélisation

- **Modèle de données :**
    - **Entités principales** :
        - **Établissement** : Contient des informations sur chaque établissement (nom, adresse, catégorie, géolocalisation, etc.).
        - **Utilisateur** : Information sur les utilisateurs (inscrit ou non) avec les avis/notes.
- **Schéma UML :**
    - À dessiner sous forme de diagramme UML pour représenter les relations entre les entités et définir la structure de la base de données.

7. Critères d'Évaluation

7. Critères d'Évaluation

- **Rédaction du cahier des charges** :
    - Description technique du projet, de son but et de son architecture (MVC).
    - Détails sur les spécifications techniques, comme la structure de la base de données, les API utilisées, et les interfaces.
- **Base de données** :
    - Schéma UML de la base de données.
    - Définition des entités et des relations entre elles (par exemple, un utilisateur peut laisser plusieurs avis).
- **Normes de codage** :
    - Respect des normes du **W3C.**
    - Code commenté de manière claire et précise.
    - Nommage des variables ayant un sens clair.
    - Diagramme de base de données UML pour la gestion des entités.
- **Sécurité** :

- La sécurité doit être prise en compte, en particulier pour la gestion des utilisateurs (inscription, connexion, avis/notes).
- Aucun accès sans connexion (problème si un utilisateur peut accéder au site sans être connecté).

8. Ressources et API à Utiliser

- **Google Maps API (optionnel)** : Pour l'affichage de cartes, mais attention à ne pas utiliser une carte payante.
- **Leaflet avec OpenStreetMap** : Pour une solution gratuite d'affichage des cartes.
- **API de géolocalisation** : Pour la détection de la position de l'utilisateur.

1. **RECAP**

- **Objectifs** :
  - L’application doit répertorier des établissements (nom, adresse, localisation).
  - L’utilisateur non inscrit peut voir une carte et filtrer les établissements par type (magasin, restaurant).
  - Chaque établissement peut recevoir une note de 1 à 5, comme sur Google.
  - Les notes sont regroupées pour afficher une moyenne des avis de l’établissement.
- **Backoffice/Administration** :
  - L'admin peut gérer les établissements via le back-office (ajout, modification, suppression).
  - Une page de connexion pour l'accès à l'admin.
- **Technologie** :
  - Utilisation de Python pour la partie serveur.
  - Frontend avec HTML, CSS et JS.
  - Architecture MVC à respecter.
  - Application responsive design, peu importe la taille de l'écran (PC, mobile, tablette).
  - Carte avec OpenStreetMap ou Leaflet, en utilisant des coordonnées GPS.
- **Critères de Notation** :
  - Cahier des charges fonctionnel et technique, respect des contraintes de base de données.
  - Diagramme de la base de données UML, incluant les objets et leurs attributs.
  - Respect des normes de codage W3C.
  - Code propre et commenté, avec des noms de variables clairs.
- **Sécurité** :
  - Aucun accès sans connexion (problème si un utilisateur peut accéder au site sans être connecté).
  - *Utilisation de Git pour la gestion du code.*

Liste des Objets pour le Cahier des Charges

1. **Établissement**
   - Attributs : nom, adresse, catégorie, géolocalisation, avis/notes.
2. **Utilisateur**
   - Attributs : identifiant, prénom, nom, email, mot de passe, historique des avis.
3. **Avis**
   - Attributs : note (de 1 à 5), commentaire, utilisateur associé, établissement associé.