Cours GMQ 580 : Géo-Informatique II

Date : 11 mars 2025

Responsables : Yves Voirin

Travail dirigé #2

Mettre en place un service web

Règles :

- La remise est prévue pour le 26 mars à minuit. Une remise en retard d'un travail à la maison implique une réduction de la valeur de la note de 20 % par jour de retard (la période de retard commence à la fin du jour prévu de remise).
- Vous devrez rendre ce qui est mentionné comme **Livrables**.
- Le travail est noté sur 20.
- Vous avez la possibilité de poser des questions durant la semaine. Les réponses seront envoyées à toute la classe.
- Vous devez vérifier les spécifications du document à rendre dans le document du Plan de Cours.
- Équipe : Anthony Desrochers

Contexte

Vous êtes développeur junior dans une petite entreprise de développement spécialisée en solutions géomatiques, GeoInnovations. Un client a contacté l'entreprise pour mettre en place une solution web permettant de consulter des informations spatiales. Il souhaite obtenir un outil simple qui permet de lire des données géospatiales, de réaliser une analyse de base et de produire une visualisation claire pour aider à la prise de décision. Votre responsable vous confie la tâche d'évaluer la faisabilité d'utiliser Flask et Docker dans ce projet. Vous avez un délai de deux semaines. L'entreprise souhaite présenter l'approche au client afin de confirmer qu'il sera capable de maîtriser ces technologies.

Objectif

Développer une API Flask qui :

1. Expose des *endpoints* pour interroger des données géospatiales sur les parcs (ex. : localisation, superficie, équipements disponibles).
2. Utilise des données géospatiales stockées dans un fichier (GeoJSON ou autre format).
3. Est intégrée dans Docker pour faciliter le déploiement.

4. Inclut une page web simple pour tester l'API.

Données fournies

Utilisez les Données ouvertes de la Ville de Sherbrooke :

- Aires aménagées

Tâches à réaliser

1. **Développement de l'API avec Flask** :
    - Créez une application Flask qui charge les données géospatiales (définir une librairie).
    - Implémentez au moins quatre endpoints REST :
        - GET /parks : Retourne la liste de tous les parcs (nom, superficie, coordonnées).
        - GET /park/<id> : Retourne les détails d'un parc spécifique en fonction de son id.
        - GET /parks/near?lat=<latitude>&lon=<longitude>&distance=<distance> : Retourne les parcs dans un rayon de X m autour d'une position donnée.
        - GET /parks/stats? lat=<latitude>&lon=<longitude>&distance=<distance> : Retourne la superficie des parcs dans un rayon de X m autour d'une position donnée.
    - Les réponses doivent être au format JSON.
2. **Conteneurisation avec Docker** :
    - Créez un fichier Dockerfile pour packager l'application Flask.
    - Incluez un fichier requirements.txt avec les dépendances (ex. : flask, geopandas, shapely).
    - Assurez-vous que l'API est accessible depuis l'extérieur du conteneur (port 5000 par défaut).
3. **Test et visualisation** :
    - Fournissez une page HTML simple (servie par Flask) permettant de tester les endpoints (ex. : un formulaire pour entrer une latitude/longitude et voir les parcs proches).
    - Intégrez une carte interactive (comme Leaflet) pour afficher les résultats (on se limite à l'opération qui permet de trouver les parcs à proximité d'une position).
4. **Documentation** :
    - Rédigez un fichier README expliquant :
        - Comment construire et lancer le conteneur Docker.
        - Comment accéder à l'API et tester les endpoints (ex. : avec curl ou un navigateur).
        - Une brève description des endpoints et de leurs paramètres.

Contraintes

- Utilisez Python avec Flask pour l'API.

- Utilisez Docker pour conteneuriser l'application.
- Les données géospatiales doivent être manipulées avec une bibliothèque comme geopandas ou shapely.
- Le code doit être structuré, commenté et versionné
- Délai : 2 semaines.

Livrables

- Un dossier contenant :
    - Le code source Flask (lien GitHub)
    - Le fichier spatial utilisé
    - Le Dockerfile et requirements.txt.
    - Une page HTML
    - Un fichier README avec les instructions.

Évaluation

Critère	Sous-critère	Points attribués	Maximum
Compréhension des consignes	Pertinence		2
	Respect des contraintes		2
Qualité technique/académique	Exactitude et rigueur		3
	Complexité et profondeur		3
Organisation et présentation	Structure		2
	Clarté		2
Fonctionnalité ou utilité	Réalisation des objectifs		2
	Praticité/applicabilité		2
Effort et originalité	Investissement personnel		1
	Créativité/initiative		1
Total brut			**20**
Pénalité (si applicable)			
Total final			**20**

Comprendre les critères :

- Compréhension et respect des consignes
  - Pertinence par rapport aux objectifs :
    - Le travail répond-il clairement aux attentes énoncées ?
  - Respect des contraintes :
    - Les règles, formats, outils ou délais spécifiés ont-ils été suivis ?
- Qualité technique ou académique
  - Exactitude et rigueur :
    - Le contenu est-il correct, précis et exempt d'erreurs majeures ?
  - Complexité et profondeur :
    - Le travail démontre-t-il un niveau de compétence ou une réflexion adapté au contexte?
- Organisation et présentation
  - Structure :
    - Le travail est-il bien organisé?
  - Clarté de la présentation :
    - Les idées, résultats ou livrables sont-ils présentés de manière compréhensible ?
- Fonctionnalité ou utilité
  - Réalisation des objectifs :
    - Le produit final atteint-il son but principal ?
  - Praticité ou applicabilité :
    - Le travail est-il utilisable ou utile dans le contexte donné ?
- Effort et originalité
  - Investissement personnel :
    - Le travail reflète-t-il un effort significatif ?
  - Créativité ou initiative :
    - Y a-t-il des éléments innovants ou des améliorations au-delà des exigences minimales ?