GUIDE DE DÉPLOIEMENT DE CARQUEST

FLASK,MONGODB

Préparé par:

- **Lamrini Imane**
- **Adbib Ilham**
- **El Madani Khadija**

Introduction

Ce guide détaille les étapes nécessaires au mise en place d'un environnement d'un projet de plateforme web de location de voitures, codé avec Flask et MongoDB.

Étapes du Processus de Déploiement

-Configuration de Python.

-Configuration de MongoDB Compass. (Le nom de la BD de l'application est: **carquest**)

-Configuration de l'environnement virtuel:

- Installation de l'environnement virtuel: pip install virtualenv

- Création d'un nouvel environnement virtuel via la commande: virtualenv env

-Configuration de l'Application Flask:

- Installation des packages requis pour le fonctionnement de l'application:

- **Flask**: Le framework web utilisé pour construire l'application: pip install Flask
- **Livereload:** Pour surveiller les modifications des fichiers de projet et recharger automatiquement le navigateur quand il détecte un changement:
  pip install livereload
- **PyMongo** et **Flask-PyMongo**: Pour interagir avec MongoDB depuis Flask:
  pip install pymongo Flask-PyMongo
- **BSON**: Installé généralement avec pymongo, mais peut être installé séparément si nécessaire pour manipuler les ObjectId, par exemple: pip install bson
- **Werkzeug**: Utilisé ici pour "secure_filename", permettant de sécuriser les noms de fichiers pour les uploads: pip install Werkzeug
- **FPDF:** Une librairie pour générer des fichiers PDF en Python: pip install fpdf
- **python-dotenv**: Pour gérer les fichiers .env permettant de stocker des configurations sensibles comme les clés secrètes et les URI de base de données:
  pip install python-dotenv

=>Pour s'assurer, au niveau du projet, que:

- L'application Flask est configurée pour se connecter à la base de données MongoDB en utilisant l'URI fourni par MongoDB.

- Utilisation des fichiers .env ou des variables d'environnement système pour stocker les informations sensibles telles que les clés secrètes et les URI de base de données.

-Versionnage du code:

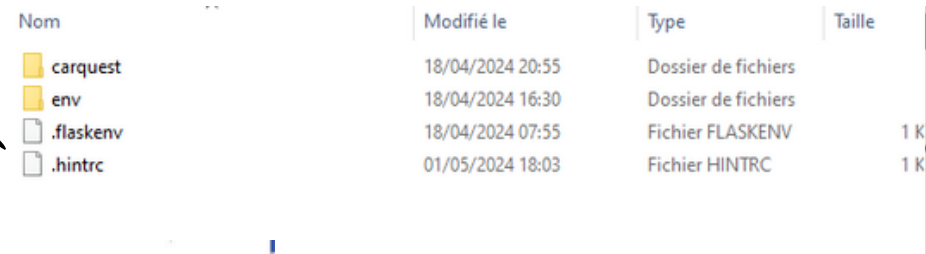
- Utilisation de Git comme outil de versionnage, qui nous a permis de collaborer efficacement dans le projet, et gérer les divers versions du code.

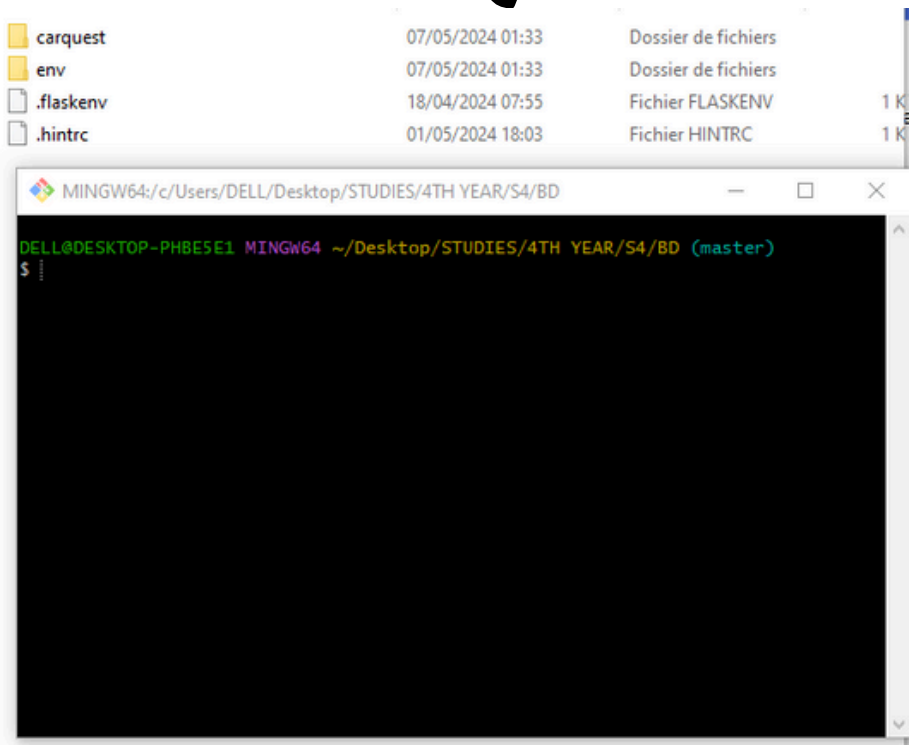
Tests et Vérifications:

- On avait effectué des tests pour assurer que tout fonctionne correctement dans l'application, et que l'application peut se connecter à la base de données MongoDB et que toutes les fonctionnalités sont opérationnelles.
- On avait utilisé **Git BASH** pour exécuter le code. (il faut que git soit installé dans la machine: git-scm.com/downloads

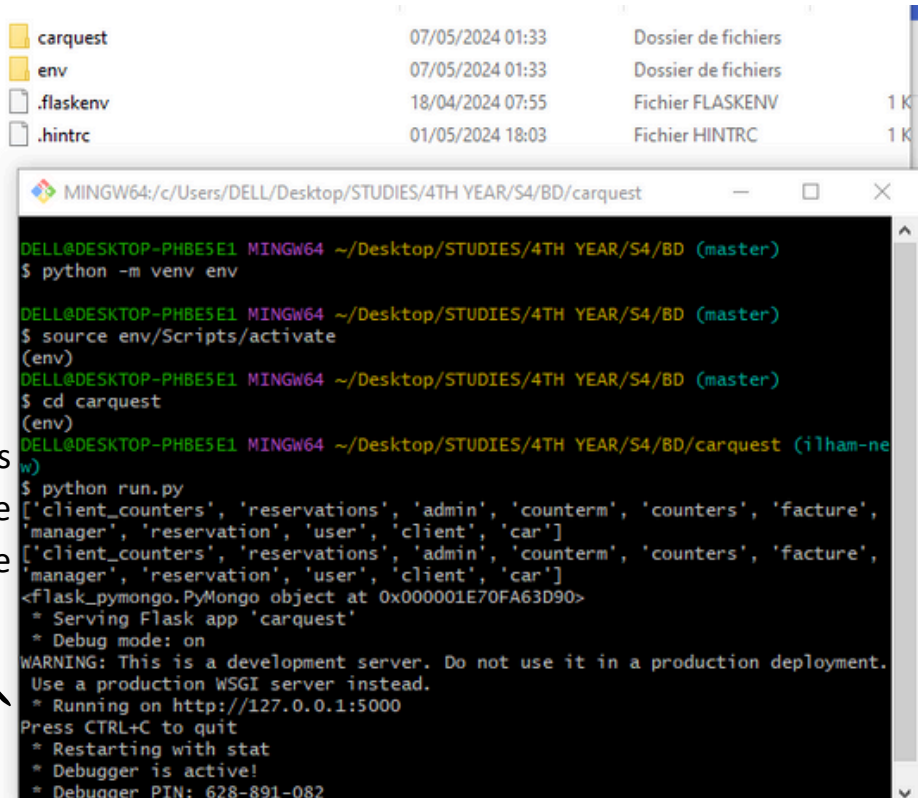
=> Pour tester le code, il faudrait suivre les étapes suivantes:

1-Décompresser le dossier du projet (**carquest**), et ouvrir le dossier décompressé.



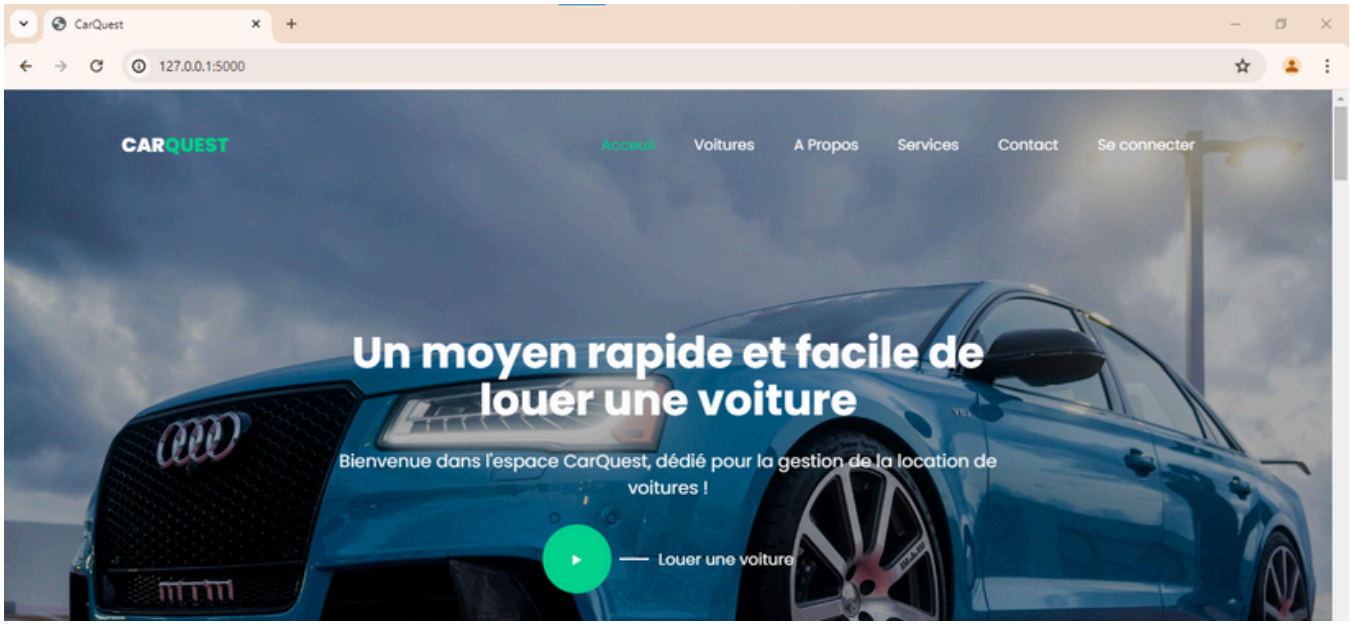


2-Ouvrir le Git BASH dans le chemin du répertoire contenant le projet comme le montre cette capture d'écran ci-contre.



3-Exécuter les commandes suivants avec le même ordre montré dans cette capture d'écran ci-contre.

4-Ouvrir le navigateur tout en saisissant comme URL: **127.0.0.1:5000**, afin de consulter toutes les fonctionnalités du projet.



Conclusion

Avec ce guide, nous avons mis en place d'un environnement au profit de notre projet de plateforme web de location de voiture CarQuest, codé avec Flask et MongoDB de manière sécurisée et fiable.