République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
École supérieure en sciences et technologies de l'informatique et du numérique

Rapport Technique Mini-projet : Génération automatique d'un portfolio GitHub

1. Introduction

Ce rapport technique présente une application développée en Python permettant d'analyser automatiquement les dépôts GitHub d'un utilisateur et de générer un portfolio professionnel sous forme d'un site web statique (HTML et CSS).

L'objectif principal de cette application est de fournir une vitrine claire, lisible et structurée des projets GitHub, en mettant en valeur des informations clés telles que le nombre de commits, les langages de programmation utilisés, le nombre d'étoiles (stars), les forks, ainsi qu'un score de valorisation calculé automatiquement.

2. Objectifs du mini projet

Les objectifs spécifiques de ce mini projet sont les suivants :

- Exploiter l'API GitHub REST pour récupérer des données réelles liées aux dépôts d'un utilisateur ;
- Analyser les repositories GitHub afin d'extraire des informations pertinentes telles que le nombre de commits, les langages utilisés, les étoiles (stars) et les forks ;
- Mettre en place un mécanisme de calcul d'un score de valorisation permettant d'évaluer l'activité et la popularité des dépôts ;
- Générer automatiquement un portfolio professionnel en HTML et CSS à partir des données analysées ;
- Appliquer les bonnes pratiques de programmation Python, notamment la structuration modulaire du code ;

- Renforcer les compétences en intégration d'API, en traitement de données et en génération de contenu web.

3. Technologies utilisées

Le projet repose sur les technologies suivantes :

- **Python** : langage principal du projet ;
- **GitHub API** : récupération des données des dépôts ;
- **Requests** : envoi de requêtes HTTP vers l'API GitHub ;
- **Jinja2** : génération dynamique du code HTML ;
- **HTML / CSS** : structure et mise en forme du portfolio ;
- **GitHub** : hébergement des dépôts analysés et du projet.

4. Architecture du mini projet

L'architecture du mini projet repose sur une organisation modulaire visant à séparer clairement les différentes responsabilités de l'application. Cette approche permet d'améliorer la lisibilité du code, de faciliter la maintenance et de rendre le mini projet évolutif.

```
mini-projet-python-portfolio-github-g04-main/
├─ .flake8
├─ LICENSE
├─ README.md
├─ Rapport Technique – mini-projet-python-portfolio-GitHub.pdf
├─ analyzer.py
├─ config.py
├─ config.yaml
├─ github_api.py
├─ html_generator.py
├─ main.py
├─ requirements.txt
├─ assets/
│  └─ style.css
├─ output/
│  └─ portfolio.html
├─ templates/
│  └─ portfolio.html
└─ tests/
   └─ test_analyzer.py
```

- **Main.py** : point d'entrée du programme, orchestre l'exécution globale ;
- **Github_api.py** : communication avec l'API GitHub (repos, commits, langages) ;
- **Analyzer.py** : analyse des données et calcul du score ;
- **Html_generator.py** : génération du portfolio HTML ;
- **Templates/** : modèle HTML avec variables dynamiques ;
- **Assets/** : ressources CSS ;
- **Output/** : résultats générés automatiquement.

5. Fonctionnement de l'application

Le fonctionnement du projet se déroule en plusieurs étapes :

1. Connexion à l'API GitHub et récupération des dépôts ciblés ;
2. Extraction des informations principales de chaque dépôt ;
3. Analyse des données et calcul des statistiques ;
4. Calcul d'un score de valorisation basé sur l'activité du dépôt ;
5. Génération automatique du portfolio HTML à partir d'un Template ;
6. Sauvegarde du résultat final dans le dossier output.

6. Analyse des données et calcul du score

Pour chaque dépôt GitHub, les informations suivantes sont analysées :

- Nombre de commits (limité à un maximum de 100 pour l'affichage) ;
- Langages de programmation utilisés ;
- Nombre d'étoiles (stars) ;
- Nombre de forks.

Un score global est calculé selon la formule suivante :

$$\textbf{Score} = (\textbf{nombre de commits} \times 2) + (\textbf{stars} \times 5) + (\textbf{forks} \times 3)$$

Ce score permet de classer les dépôts selon leur popularité et activité.

7. Génération du portfolio HTML

La génération du portfolio est réalisée à l'aide du moteur de templates **Jinja2**.

Les données analysées sont injectées dans un template HTML afin de produire un fichier statique :

output/portfolio.html

Le design est assuré par une feuille de style CSS séparée, garantissant une bonne lisibilité et une présentation professionnelle.

8. Résultat obtenu

À l'issue de l'exécution du programme, un fichier HTML est généré automatiquement.
Le portfolio final présente :

- Un en-tête décrivant le projet et le groupe ;
- Une section listant les mini-projets analysés ;
- Pour chaque projet :
  - Le nom et le lien GitHub ;
  - La description ;
  - Les statistiques (langage, commits, stars, forks) ;
  - Le score calculé ;
  - La liste des commits récents.

9. Conclusion

Ce projet a permis de mettre en pratique les concepts clés de la programmation Python, de l'exploitation d'API REST et de la génération automatique de contenu web.
Le résultat final est un portfolio professionnel et dynamique permettant de valoriser les mini-projets collaboratifs réalisés par les différents groupes.