Mastère SIO - Octobre 2020 - Projet Fil Rouge
V 1.0

CentraleSupélec

L'objectif de ce mini projet est de réaliser une API de type REST en Python, accessible sur un serveur distant hébergé dans le cloud.

Ce projet initial volontairement simple dans sa forme initiale sera repris par les autres enseignants du MS - potentiellement sur tous les aspects - dans le cadre de leur cours/TDs/TPs/ Évaluation de fin de module, et donc est susceptible de se compliquer (par exemple une distribution supplémentaire à l'aide d'un Docker…) !

La réalisation comporte une partie **infrastructure,** une partie **conception** et enfin un **codage de logiciel**.

Infrastructure :

- Le choix du système d'exploitation importe peu pourvu qu'il soit de la famille des « unix libres » ;
- L'infrastructure qui supporte ce service devra être hébergée dans le cloud (à préciser ultérieurement) ;
- L'API devra être accessible à distance sur le port de votre choix ;
- La machine devra être accessible par ssh pour sa maintenance, sur le port de votre choix ;
- **Bonus 1** : Usage du « packet filter » vu en TD pour protéger et limiter les accès à cette API, notamment limiter le nombre de requêtes par secondes pour les usagers ;
- **Bonus 2** : Usage du protocole **`https`** au lieu d'**`http`** associé à une identification des usagers ;
- **Bonus 3** : Usage de l'OS FreeBSD vu en IPv4 et qui est nouveau pour la plupart d'entre vous !

Mastère SIO - Octobre 2020 - Projet Fil Rouge

CentraleSupélec

Conception et réalisation logicielle :

Le propos applicatif de cette API est d'accepter le dépôt de tout type de fichier et de le restituer au format JSON(1). Pour certain type de fichiers (images…) il faudra songer à un encodage de type 'base64'.

Une API de type RESTfull a des propriétés décrites ici : https://restfulapi.net/

L'usage de l'API est simple à décrire : l'usager soumet un fichier (.txt, .csv, .pdf, .jpg, .png, .gif, …) et récupère sa traduction en JSON associée à des méta-données établies lors de sa traduction (type MIME reconnu (4), taille, …)

Le retour devra comporter les méta-données et les données, bien séparées.

Au minimum l'API devra traiter du texte (.txt, .pdf), des nombres (.csv) et des images. Tout ajout de format en plus des contraintes initiales sera considéré comme un **bonus** !!

En cas de format insupporté il faudra retourner une erreur explicite à l'usager, (un crash de l'API n'est pas une erreur explicite…).

Le service sera implémenté en Python (3.x) avec « Flask (3) » comme moteur web.

(1)http://json.org/
(2)https://swagger.io/specification/
(3)http://flask.palletsprojects.com/
(4)https://fr.wikipedia.org/wiki/Type_de_médias

CentraleSupélec

Un outil shell très commode pour tester une API du point de vue du client : **curl** (1), sinon vous pouvez également écrire un script en Python, par exemple à l'aide du framework **requests**(2)

Restitution du projet :

- Ce projet comporte volontairement des zones non définies. Votre travail de conception et d'architecture devra les identifier et y apporter une réponse technique ;

- Le rendu du projet est individuel, sous forme d'**un rapport technique** (choix techniques, usages et limitations, difficultés….) structuré avec introduction et conclusion, de 20 pages au maximum, et d'**une machine virtuelle** dans le cloud AWS (à confirmer), à laquelle vous nous donnerez accès ;

- La date limite de **rendu** est le vendredi **02 avril 2021 à 23h59m59s** !

Heureuse programmation !

(1) https://curl.haxx.se

(2) https://requests.readthedocs.io/en/master/

Quelques précisions et recommandations :

1. Pas d'interface web graphique demandée dans ce projet IPv4 ;
2. Un serveur type « serverless » (AWS Lambda) ne consomme pas de ressource si on ne l'invoque pas ;
3. Penser les services « Stateless/Idem potent » ;
4. Prévoir un script (module "requests" de Python) de tests des fonctionnalités testables à distance et montrer son fonctionnement dans le rapport est un **bonus ;**
5. Keep-it "simple" !! Trop de features non demandées n'engendrent pas nécessairement de bonus supplémentaire ;
6. La seule commande imposée est "upload" pour télécharger un fichier de données, s'il y en a d'autres et qu'elles sont utiles c'est du bonus ;
7. Accès anonyme possible (pensez au correcteur) du genre :

```
 curl -k -X POST -H "mon_tag:… "https://mon_ip/upload" -F "data=@mon_fichier.truc"
Eventuellement avec –user pour une auth de type "Basic
```

(1) https://curl.haxx.se