Overview

Objectif

L'objectif de ce projet est de créer une API de sentiment analysis. Cette API peut être utilisée par différents utilisateurs qui s'identifieront via un système de username/mot de passe

Deux version du modèles sont considéré, i.e. v1, v2, La liste des username , password et des permissions pour le modèle v1 et v2 sont contenues dans le fichiers credentials.csv : si un username a une colonne v1 qui vaut 1 alors il a accès à la première version de l'API, de même avec la colonne v2.

Pour télécharger credentials,csv, lancez la commande suivante.

```
wget https://dst-de.s3.eu-west-3.amazonaws.com/Flask/credentials.csv
```

Architecture Globale

Le module de base (api,py) consiste en un ensemble de fonctions accessibles via une API HTTP RESTful. Son implémentation est basée sur FastAPI [1],

Ce module et les modules d'accès/gestion des données d'authentification (data,py), ainsi que celui implémentant la logique (sentiments.py), sont les composants de base constituant notre architecture. Il s'agit d'une architecture n-tiers (également appelée multi-tiers), qui est un modèle architectural dans lequel les fonctions de présentation, de traitement des applications (logique) et de gestion des données sont séparées [2].

Format de réponse

Toutes les réponses sont de la forme suivante:

```
{
    'response_code': code

    'username': username,

    'results': {


    }
}
```

0 dans le cas d'une requête réussite, sinon le code d'erreur correspondant

en cas de requête avec authentification

le corps de la réponse, e.g. "status", "score", etc
dans le cas particulier d'une erreur (response_code >= 1), cette partie contiendra un champs appelé "error_details", donnant plus de détails sur la nature de l'erreur

Gestion des Erreurs

Concernant la gestion des erreurs, notre approche consiste à inclure, dans chaque réponse, un champ baptisé *'response_code"*. En voici les différentes valeurs :

Code	Description
0	L'appel est correctement traité
1	Type d'authentification inconnu
2	Information d'authentification incorrectes ou absentes
3	L'appel est correct, mais la ressource n'est pas accessible pour l'utilisateur en cours

En plus du code d'erreur, un champ supplémentaire est fourni, donnant plus de détails sur l'erreur.

Authentification

Deux variantes d'authentification sont supportées par notre système, chacune avec un code *"auth_type"* différent :

<auth_type = 1>

Il s'agit d'une version simple où le couple *login/pass* est passés via *"query arguments"* (par opposition au *"path parameters"*).

Exemple :

http://127.0.0.1:5000/permissions/1/?password=6837&username=Megan

<auth_type = 2>

C'est une version améliorée où le couple *user/pass* est passé en utilisant une authentification de de type *"Basic Auth"* [3,4], i.e.

> *"" It is a method for an HTTP user agent to provide a user name and password when making a request. In basic HTTP authentication, a request contains a header field in the form of Authorization: Basic <credentials>, where credentials is the Base64 encoding of ID and password joined by a single colon [3]""*

Exemple :

http://127.0.0.1:5000/permissions/2/

Ce test peut se faire très facilement avec des outils tels *postman, c.f.* ***Fig, 01***

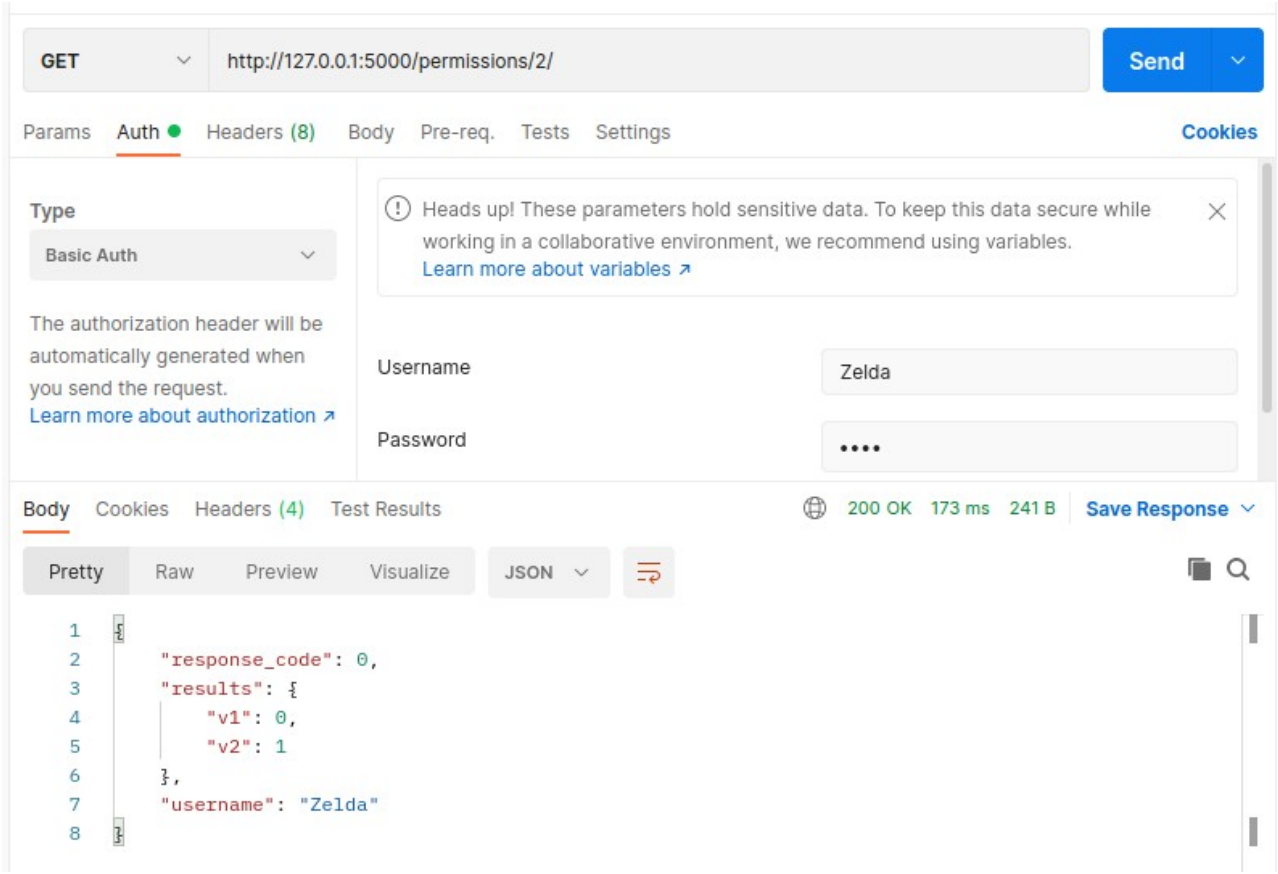

Fig. 01 : Exemple de requête effectué par Postman

End Points

Les différents endpoints sont les suivants:

Method	Path	Parameters	Details
GET	/status	None	Renvoie 1 si l'API fonctionne.
GET	/permissions/<auth_type>/	auth_type + (username et password)	Renvoie la liste des permissions d'un utilisateur authentifié
GET	/v1/<auth_type>/sentiment/	auth_type + sentence (username et password)	Une version plus simple qui simule le renvoie du score de sentiment d'une phrase proposée par l'argument *sentence* si l'utilisateur *username* est bien identifié.
GET	/v2/<auth_type>/sentiment/	auth_type + sentence (username et password)	Une 2éme version qui renvoie le compound d'un *VaderSentiment (SentimentIntensityAnalyzer()).*

Exemples

- Request URL: 127.0.0.1:5000/v2/2/sentiment/?sentence='I am happy'
- Résultat :


```
{
    "response_code": 0,
    "results": {
        "score": 0.5719,
        "sentence": "'I am happy'"
    },
    "username": "Zelda"
}
```


- Request URL: 127.0.0.1:5000/v1/2/sentiment/?sentence='I am happy'
- Résultat :


```
{
    "response_code": 3,
    "results": {
        "error_details": "Forbiden for the current user",
    },
    "username": "Zelda"
}
```


- Request URL: 127.0.0.1:5000/v2/1/sentiment/?username=noname&password=anything&sentence='I am happy'
- Résultat :


```
{
    "response_code": 2,
    "results": {
        "error_details": "Unkown <username> or <password>",
    },
    "username": "noname"
}
```


Références

[1] https://fastapi.tiangolo.com/

[2] https://en.wikipedia.org/wiki/Multitier_architecture

[3] https://en.wikipedia.org/wiki/Basic_access_authentication

[4] https://swagger.io/docs/specification/authentication/basic-authentication/