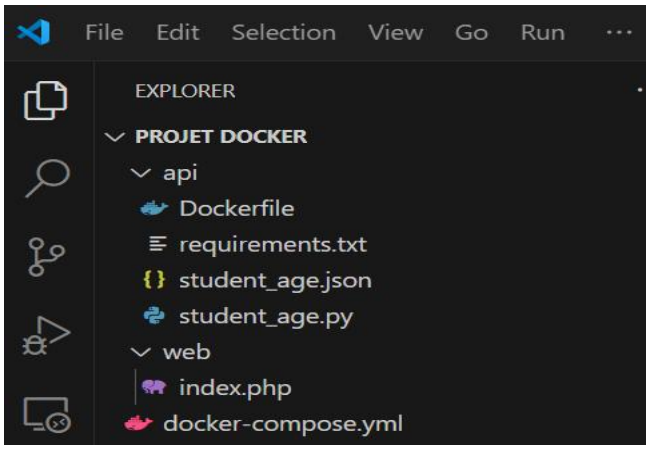
Aissatou Cisse

Application POZOS liste des étudiants

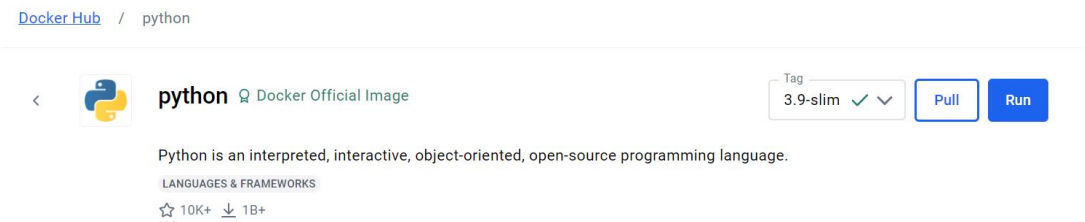
Après avoir clone tous les fichiers qui se trouve dans le liens gitup
https://github.com/guissepm/student-list.git
J'ai ouvrir tous mes dossier sur mon environnement visual studio code enfin d'éditer les fichier Dockerfile et docker-compose

Il faut savoir que nous avons deux dossier api qui contient les fichier suivant : Dockerfile , requirements.txt , student_age.json et student_age.json que nous allons expliqué à la suite et le dossier web qui contient le fichier index.php

Dossier API



Maintenant pour le Dockerfile j'ai utilisé limage Python:3.9-slim parce que c'est la version qui est disponible sur mon docker hub



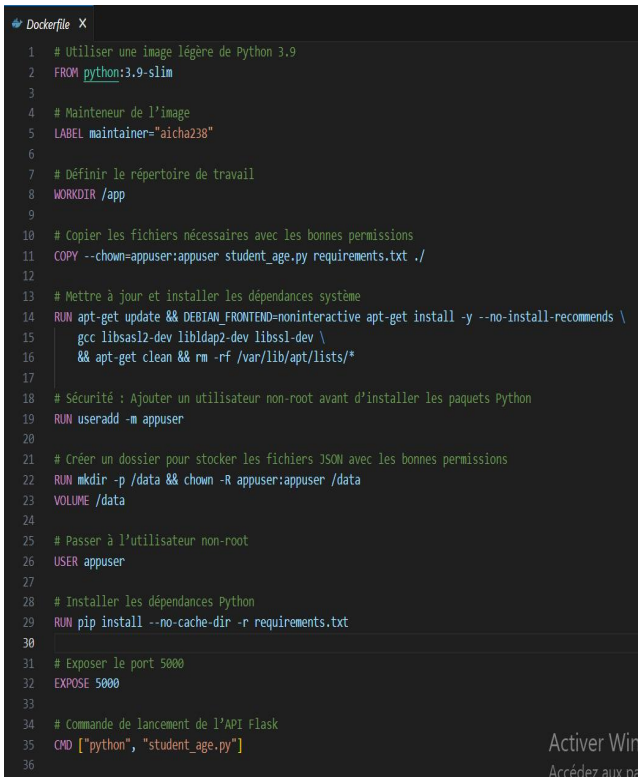
Description du code **Dockerfile**:

- Utilisation d'une version allégée (slim) de Python 3.9 pour réduire la taille de l'image Docker.
- Définit un label avec le nom du mainteneur de l'image
- Change le répertoire de travail dans le conteneur vers /app, où seront copiés les fichiers de l'application
- Copie le fichier student_age.py et requirements.txt tout en attribuant les permissions à appuser.
- Installe des paquets essentiels comme gcc et libssl-dev, puis nettoie les fichiers inutiles pour minimiser l'image.

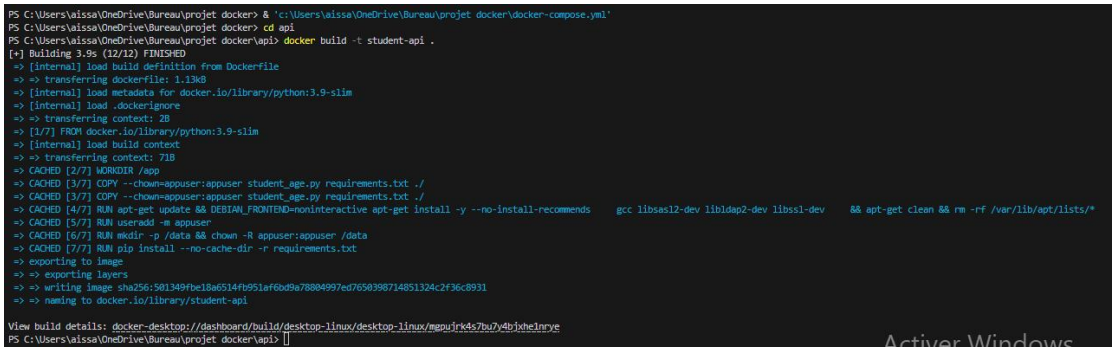
- Crée un utilisateur appuser pour éviter d'exécuter l'application avec l'utilisateur root (bonne pratique de sécurité).
- Crée un répertoire /data, assigne les bonnes permissions et le déclare comme un volume Docker pour la persistance des données

- Passe à `appuser` avant d'exécuter les commandes suivantes, garantissant que l'application ne tourne pas avec des privilèges élevés.

- Installe les dépendances de l'application en évitant d'utiliser le cache pour réduire la taille de l'image.
- Indique que l'application écoutera sur le port 5000.
- Exécute le script student_age.py au démarrage du conteneur.



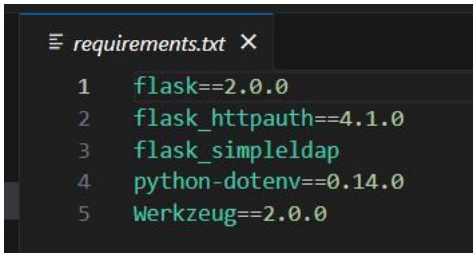
Ensuite j'ai créer l'image student_api avec la commande docker build -t student_api



requirements.txt qui est utilisé pour spécifier les dépendance dans notre cas nous avons les dépendances suivants :

- flask==2.0.0 : Spécifie la version 2.0.0 de Flask, un framework web pour Python.

- flask_httpauth==4.1.0 : Extension Flask pour l'authentification HTTP.
- flask_simpleldap : Extension Flask pour l'intégration avec LDAP (Lightweight Directory Access Protocol).
- python-dotenv==0.14.0 : Gestion des variables d'environnement.
- Werkzeug==2.0.0 : Bibliothèque WSGI utilisée par Flask.



Student_age.py qui est une API REST en Flask qui permet de gérer les âges des étudiants à partir d'un fichier JSON (student_age.json).

```python
from flask import Flask, jsonify, abort, make_response, request
from flask_httpauth import HTTPBasicAuth
import json
import os

auth = HTTPBasicAuth()
app = Flask(__name__)
app.debug = True

# 🔹 Authentification sécurisée
@auth.verify_password
def verify_password(username, password):
    if username == 'toto' and password == 'python':
        return True
    return False

@auth.error_handler
def unauthorized():
    return make_response(jsonify({'error': 'Unauthorized access'}), 401)

# 🔹 Définition du chemin du fichier JSON
student_age_file_path = os.getenv('student_age_file_path', '/data/student_age.json')

# 🔹 Vérifier si le fichier JSON existe
if not os.path.exists(student_age_file_path):
    print(f" Attention : Le fichier {student_age_file_path} n'existe pas !")
    student_age = {}
else:
    with open(student_age_file_path, "r") as student_age_file:
        student_age = json.load(student_age_file)

# 🔹 Récupérer tous les étudiants
@app.route('/pozos/api/v1.0/get_student_ages', methods=['GET'])
@auth.login_required
def get_student_ages():
    return jsonify({'student_ages': student_age})

# 🔹 Récupérer l'âge d'un étudiant
@app.route('/pozos/api/v1.0/get_student_ages/<student_name>', methods=['GET'])
@auth.login_required
def get_student_age(student_name):
    if student_name not in student_age:
        abort(404)
    return jsonify({'name': student_name, 'age': student_age[student_name]})

# 🔹 Gestion des erreurs 404
@app.errorhandler(404)
def not_found(error):
    return make_response(jsonify({'error': 'Not found'}), 404)

# 🔹 Lancement de l'API
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

student_age.json qui est un fichier JSON et il contient un dictionnaire avec les noms des étudiants et leurs âges correspondant

student_age.json

```json
{
    "Yague Kebe": "24",
    "Astou Gueye": "23",
    "Zakaria Sidibe": "24",
    "Aissatou Cisse": "23",
    "Khady Fall": "23",
    "Ndeye Gnilane Ndour": "22",
    "Fatou Ndiaye": "23",
    "Binta Ndiaye ": "23",
    "Omar Louis Kao": "24",
    "Mamadou Bamba Dieye": "24",
    "Falou Ndiaye": "24",
    "Ndoumbé Guaye" : "23"

}
```

Dossier Web:

Nous avons ici index.php qui contient le code HTM , CSS et PHP pour interface web des utilisateurs

```php
<html>
<head>
    <title>POZOS</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        table {
            width: 60%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 8px;
            text-align: center;
        }
        th {
            background-color: #4CAF50;
            color: white;
        }
        tr:nth-child(even) {
            background-color: #f2f2f2;
        }
        h1 {
            color: #333;
        }
        p {
            color: red;
            font-size: 20px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            cursor: pointer;
            font-size: 16px;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h1>Application de vérification des étudiants</h1>
    <form action="" method="POST">
        <button type="submit" name="submit">Liste des étudiants</button>
    </form>
```

```php
<?php
if ($_SERVER['REQUEST_METHOD'] == "POST" && isset($_POST['submit'])) {
    $username = "toto"; // Remplacez par getenv('USERNAME') si défini
    $password = "python"; // Remplacez par getenv('PASSWORD') si défini
    $url = 'http://student_api:5000/pozos/api/v1.0/get_student_ages';

    // Initialisation de cURL
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        "Authorization: Basic " . base64_encode("$username:$password")
    ]);

    $response = curl_exec($ch);
    $http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    if ($http_code == 200) {
        $list = json_decode($response, true);
        echo "<p>Voici la liste des étudiants :</p>";

        // Table HTML pour afficher les données
        echo "<table>";
        echo "<tr><th>Nom de l'étudiant</th><th>Âge</th></tr>";
        foreach ($list["student_ages"] as $key => $value) {
            echo "<tr><td>$key</td><td>$value</td></tr>";
        }
        echo "</table>";
    } else {
        echo "<p>Erreur: Impossible de récupérer les données des étudiants (Code HTTP: $http_code)</p>";
    }
}
?>
</body>
</html>
```

Activer Windo
Accédez aux paran

Et en fin le fichier docker-compse.yml nous avons deux services

- Service API (student_api)

Construit à partir du dossier ./api

Expose le port 5000

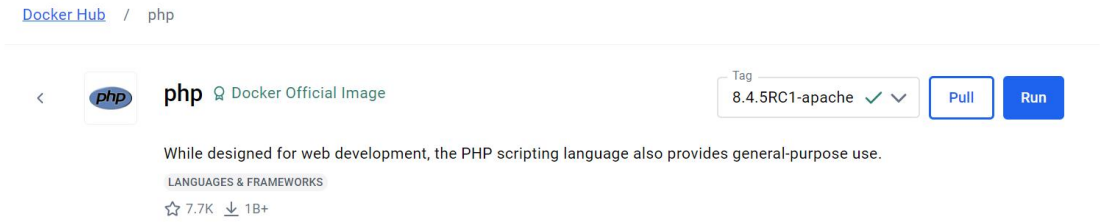
Monte le fichier student_age.json dans /data/student_age.json

Connecté au réseau student_network

Redémarre automatiquement sauf arrêt manuel

- Service Web (web)

Utilise l'image php:8.4.5RC1-apache que nous avons télécharger dans le docker Hub



Définit des variables d'environnement (USERNAME=toto, PASSWORD=python)

Monte le fichier index.php dans /var/www/html/index.php

Expose le port 8080 pour l'accès au site web

Dépend du service API (ne démarre que si api est actif)

Connecté au réseau student_network

- Réseau student_network

Utilise un bridge pour la communication entre services

```yaml
version: '3.8'

services:
  api:
    build: ./api
    container_name: student_api
    ports:
      - "5000:5000"
    volumes:
      - ./api/student_age.json:/data/student_age.json
    networks:
      - student_network
    restart: unless-stopped

  web:
    image: php:8.4.5RC1-apache
    environment:
      - USERNAME=toto
      - PASSWORD=python
    volumes:
      - ./web/index.php:/var/www/html/index.php

    ports:
      - "8080:80"
    depends_on:
      - api
    networks:
      - student_network
    restart: unless-stopped

networks:
  student_network:
    driver: bridge
```

Déploiement des conteneurs : student_api → Pour notre API.

· projetdocker-web-1 → pour le service web lié.

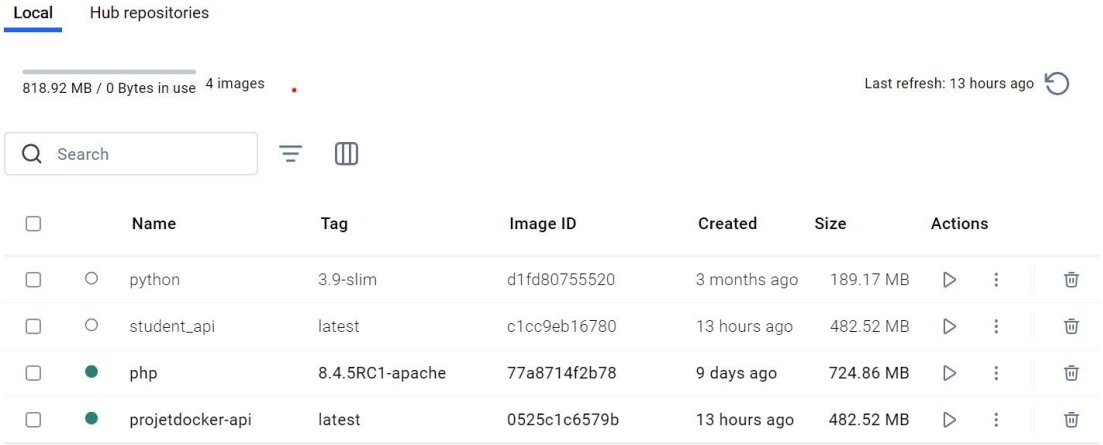
```
PS C:\Users\aissa\OneDrive\Bureau\projet docker>
>> docker-compose up  -d
>>
time="2025-03-09T00:10:00Z" level=warning msg="C:\\Users\\aissa\\OneDrive\\Bureau\\projet docker\\docker-compose.yml: the
attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 2/2
 ✓ Container student_api          Running                                                    0.0s
 ✓ Container projetdocker-web-1   Running                                                    0.0s
PS C:\Users\aissa\OneDrive\Bureau\projet docker>
```

Activer Windows
Accédez aux paramètres pour activer Windows.

Exécuter docker pour voir si c'est réussi

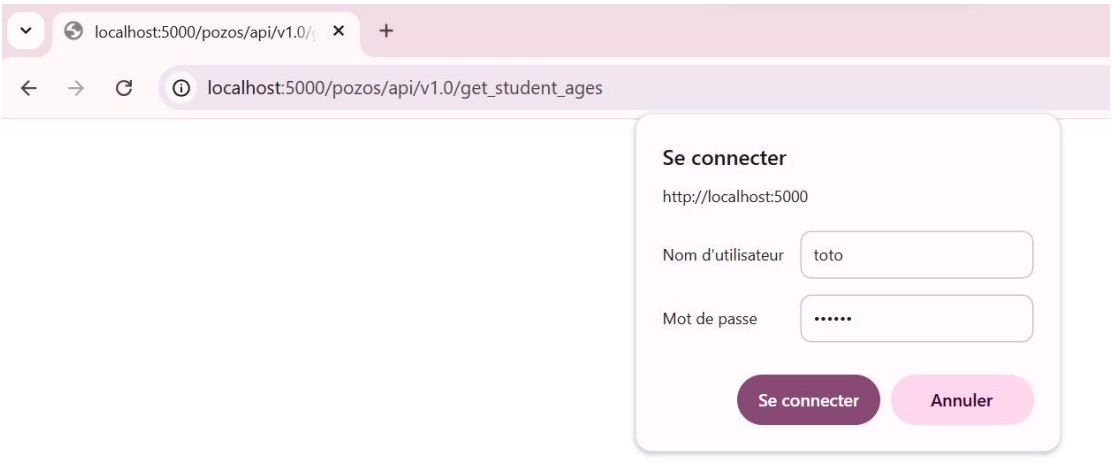
```
PS C:\Users\aissa\OneDrive\Bureau\projet docker> docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS                    NA
MES
f5de598208be   projetdocker-api      "python student_age.…"   46 seconds ago   Up 38 seconds   0.0.0.0:5000->5000/tcp   st
udent_api
bedb404d62d7   php:8.4.5RC1-apache   "docker-php-entrypoi…"   6 minutes ago    Up 36 seconds   0.0.0.0:8080->80/tcp     pr
ojetdocker-web-1
PS C:\Users\aissa\OneDrive\Bureau\projet docker>
```

On peut aussi voir les images dans notre docker Desktop

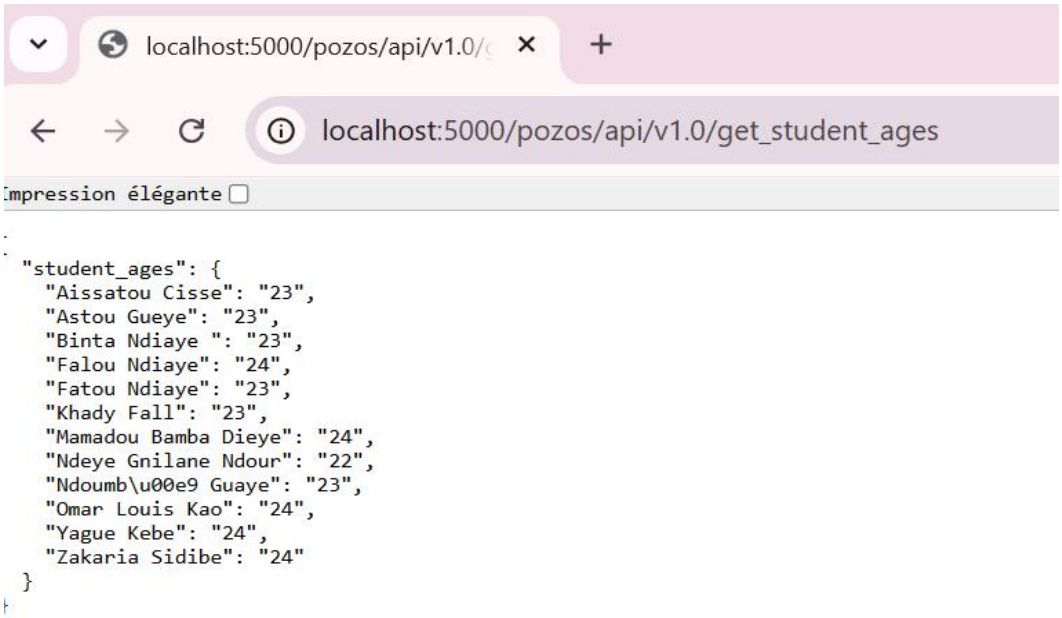


Maintenant que nous avons fini nous allons tester si l'application fonctionne

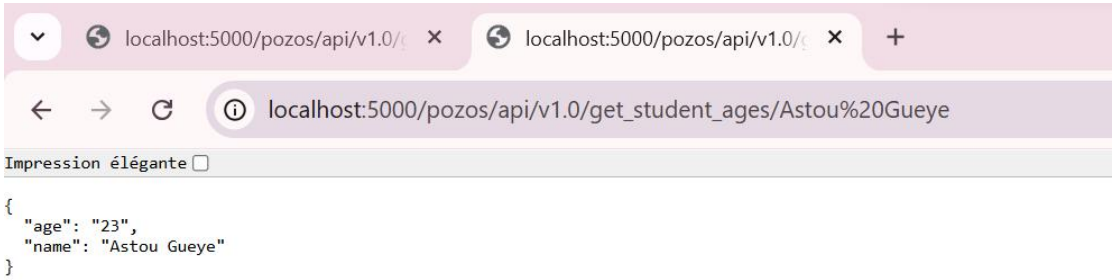
Premièrement nous allons vérifier API avec le route qu'on a spécifié dans notre fichier student_age.py et le port 5000 expose et nous allons ensuite s'authentifier avec le non d'utilisateur toto et mots de passe python



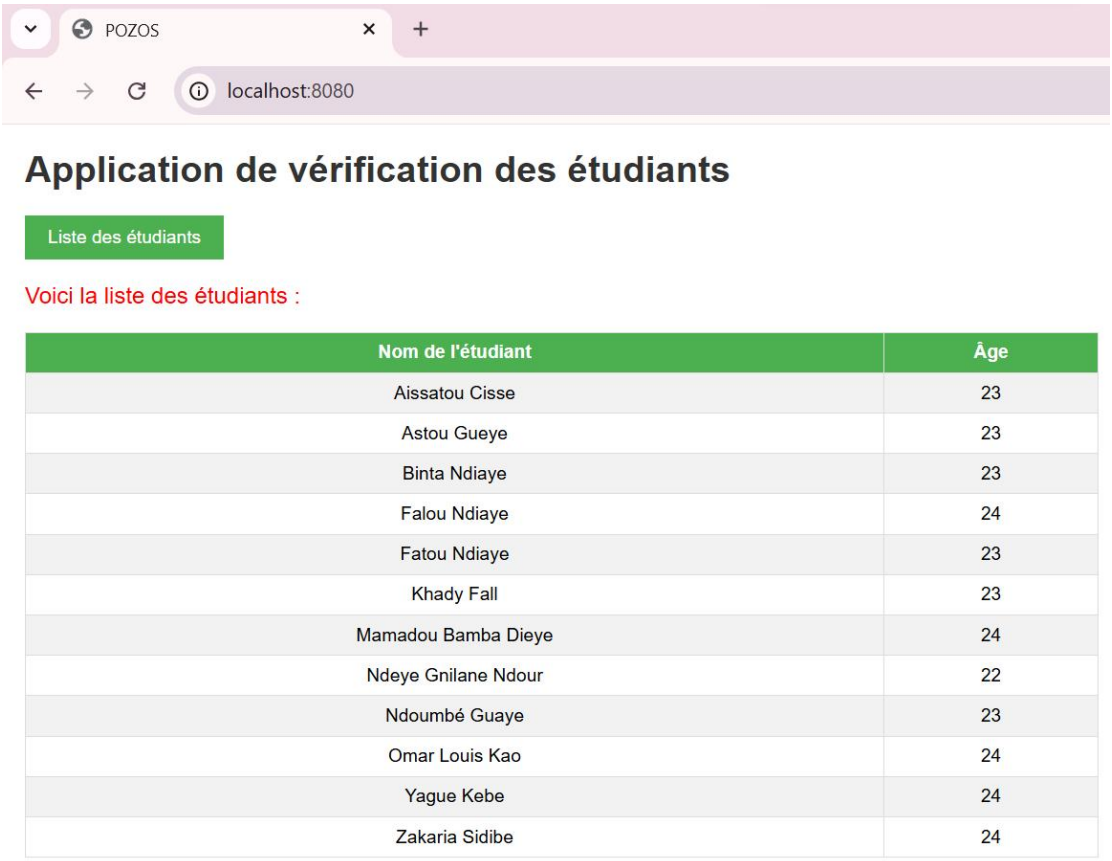
Ici nous avons accès et on vois les donner que nous avons définis dans notre fichier JSON c'est a dire les noms des étudiants et leurs âges.



Dans notre fichier students_age.py nous avons spécifier un route pour savoir l' age d'un étudiant donner par exemple j'ai filtre pour voir l' âges de étudiant nommer Astou Gueye



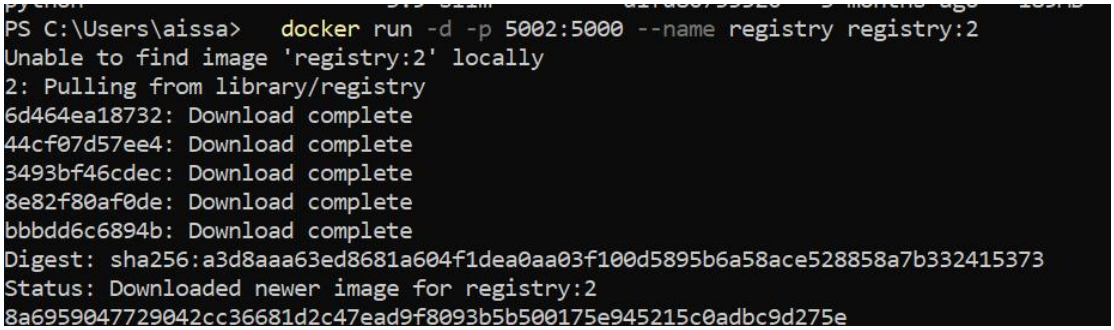
On vois bien que notre API rest fonctionne parfaitement vérifions la partis WEB dans la partis web nous avons image php apache et nous avons aussi expose le port 8080 donc dans notre navigateur on va taper localhost:8080 pour voir la liste des étudiants



registre privé

- un registre Docker

Exécute cette commande pour démarrer un registre local sur le port 5002



Tagger ton image pour le registre local

docker tag student_api localhost:5002/student-api

Pousser l'image dans le registre

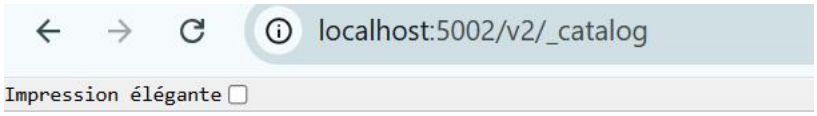
docker push localhost:5002/student-api

```
PS C:\Users\aissa>  docker tag student_api localhost:5002/student-api
PS C:\Users\aissa> docker push localhost:5002/student-api
Using default tag: latest
The push refers to repository [localhost:5002/student-api]



6933c80f6539: Pushed
fb0009da06dd: Pushed
a48466e237ef: Pushed
27a041f54951: Pushed
ad2a6251d8bd: Pushed
82a59f028fa3: Pushed
09f8c21e9f8e: Pushed
5089b4537685: Pushed
c2b670ec11bc: Pushed
7cf63256a31a: Pushed
793774c5f467: Pushed
latest: digest: sha256:c1cc9eb1678070ca24fb6f0f6433b0d1347ac02c1f7e74abb9b60676d26e0c48 size: 856
PS C:\Users\aissa>
```

Test de l'accès au registre



{"repositories":["student-api"]}

une interface Web pour voir l'image poussée comme un conteneur

Utilise joxit/docker-registry-ui, qui fonctionne bien avec Docker Registry

```
PS C:\Users\aissa>  docker run -d -p 8082:80 --name registry-ui -e REGISTRY_URL=http://registry:5002 --link registry:registry joxit/docker-registry-ui
Unable to find image 'joxit/docker-registry-ui:latest' locally
latest: Pulling from joxit/docker-registry-ui
d18780149b81: Already exists
4f4fb700ef54: Already exists
b8f1be213d73: Already exists
6c866301bd2c: Download complete
6ace07744b36: Already exists
9e03973bc803: Already exists
4ea31a8fb875: Already exists
e57ebb3e2067: Already exists
a3a550dcd386: Already exists
cf7a173b6dfb: Already exists
c306db4532fb: Already exists
5a4f7d72e702: Already exists
Digest: sha256:5594b76bf8dd9de479648e28f38572d020d260568be40b7e52b9758b442275e1
Status: Downloaded newer image for joxit/docker-registry-ui:latest
bbabd1721ba436e77eaa64baddd5384ee900e4758b5c2105e8b56f0530240732
PS C:\Users\aissa>
```

Ouvre ensuite http://localhost:8082