

Aissatou Cisse

## Application POZOS liste des étudiants

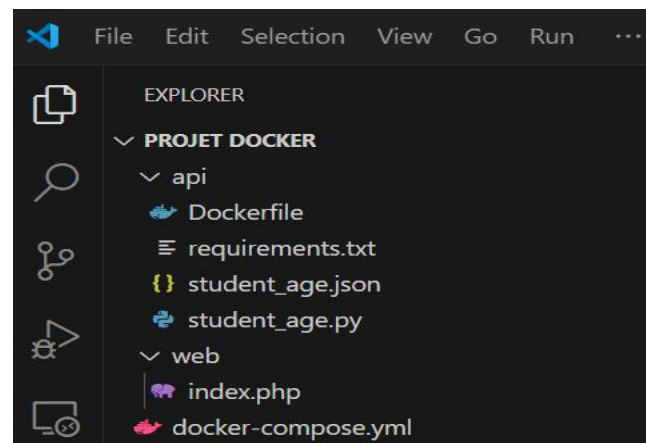
Après avoir clone tous les fichiers qui se trouvent dans le lien gitup

<https://github.com/guissepm/student-list.git>

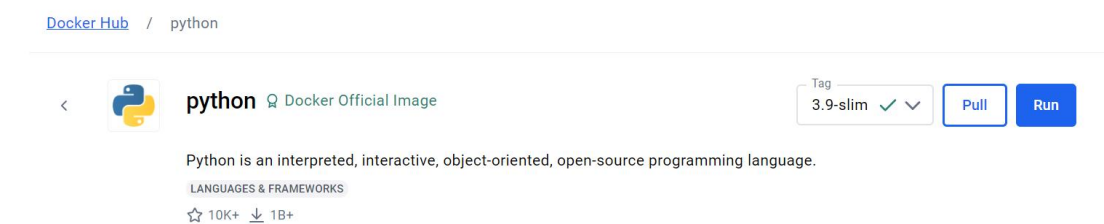
J'ai ouvert tous mes dossiers sur mon environnement Visual Studio Code afin d'éditer les fichiers Dockerfile et docker-compose

Il faut savoir que nous avons deux **dossier api** qui contiennent les fichiers suivants : Dockerfile , requirements.txt , student\_age.json et student\_age.py que nous allons expliquer à la suite et le **dossier web** qui contient le fichier index.php

### Dossier API



Maintenant pour le Dockerfile j'ai utilisé l'image 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.

```
# Dockerfile X
1 # Utiliser une image légère de Python 3.9
2 FROM python:3.9-slim
3
4 # Mainteneur de l'image
5 LABEL maintainer="aicha238"
6
7 # Définir le répertoire de travail
8 WORKDIR /app
9
10 # Copier les fichiers nécessaires avec les bonnes permissions
11 COPY --chown=appuser:appuser student_age.py requirements.txt ./
12
13 # Mettre à jour et installer les dépendances système
14 RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
15     gcc libssl2-dev libldap2-dev libssl-dev \
16     && apt-get clean && rm -rf /var/lib/apt/lists/*
17
18 # Sécurité : Ajouter un utilisateur non-root avant d'installer les paquets Python
19 RUN useradd -m appuser
20
21 # Créer un dossier pour stocker les fichiers JSON avec les bonnes permissions
22 RUN mkdir -p /data && chown -R appuser:appuser /data
23 VOLUME /data
24
25 # Passer à l'utilisateur non-root
26 USER appuser
27
28 # Installer les dépendances Python
29 RUN pip install --no-cache-dir -r requirements.txt
30
31 # Exposer le port 5000
32 EXPOSE 5000
33
34 # Commande de lancement de l'API Flask
35 CMD ["python", "student_age.py"]
36
```

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

```
PS C:\Users\alissa\OneDrive\Bureau\projet docker> & 'c:\Users\alissa\OneDrive\Bureau\projet docker\docker-compose.yml'
PS C:\Users\alissa\OneDrive\Bureau\projet docker> cd api
PS C:\Users\alissa\OneDrive\Bureau\projet docker\api> docker build -t student-api .
[+] Building 3.9s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.13kB
=> [internal] load metadata for docker.io/library/python:3.9-slim
=> [internal] load <dockerignore>
=> => transferring context: 2B
=> [1/7] FROM docker.io/library/python:3.9-slim
=> [internal] load build context
=> => transferring context: 71B
=> CACHED [2/7] WORKDIR /app
=> CACHED [3/7] COPY --chown=appuser:appuser student_age.py requirements.txt ./
=> CACHED [3/7] COPY --chown=appuser:appuser student_age.py requirements.txt ./
=> CACHED [4/7] RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends gcc libssl2-dev libldap2-dev libssl-dev && apt-get clean && rm -rf /var/lib/apt/lists/*
=> CACHED [5/7] RUN useradd -m appuser
=> CACHED [6/7] RUN mkdir -p /data && chown -R appuser:appuser /data
=> CACHED [7/7] RUN pip install --no-cache-dir -r requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:501349fbc18a6514f0951af60b9a78804997ed7650396714851324c2f36c8931
=> => naming to docker.io/library/student-api
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/mespjrk4s7bu7y4b1jhe1nrye
PS C:\Users\alissa\OneDrive\Bureau\projet docker\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.

```

requirements.txt X
1 flask==2.0.0
2 flask_httpauth==4.1.0
3 flask_simpleldap
4 python-dotenv==0.14.0
5 Werkzeug==2.0.0

```

**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).

```

student_age.py X
1 from flask import Flask, jsonify, abort, make_response, request
2 from flask_httpauth import HTTPBasicAuth
3 import json
4 import os
5
6 auth = HTTPBasicAuth()
7 app = Flask(__name__)
8 app.debug = True
9
10 # * Authentification sécurisée
11 @auth.verify_password
12 def verify_password(username, password):
13     if username == 'toto' and password == 'python':
14         return True
15     return False
16
17 @auth.error_handler
18 def unauthorized():
19     return make_response(jsonify({'error': 'Unauthorized access'}), 401)
20
21 # * Définition du chemin du fichier JSON
22 student_age_file_path = os.getenv('student_age_file_path', '/data/student_age.json')
23
24 # * Vérifier si le fichier JSON existe
25 if not os.path.exists(student_age_file_path):
26     print(f"Attention : Le fichier {student_age_file_path} n'existe pas !")
27     student_age = {}
28 else:
29     with open(student_age_file_path, "r") as student_age_file:
30         student_age = json.load(student_age_file)
31
32 # * Récupérer tous les étudiants
33 @app.route('/api/v1.0/get_student_ages', methods=['GET'])
34 @auth.login_required
35 def get_student_ages():
36     return jsonify({'student_ages': student_age})
37
38 # * Récupérer l'âge d'un étudiant
39 @app.route('/api/v1.0/get_student_ages/<student_name>', methods=['GET'])
40 @auth.login_required
41 def get_student_age(student_name):
42     if student_name not in student_age:
43         abort(404)
44     return jsonify({'name': student_name, 'age': student_age[student_name]})
45
46 # * Gestion des erreurs 404
47 @app.errorhandler(404)
48 def not_found(error):
49     return make_response(jsonify({'error': 'Not found'}), 404)
50
51 # * Lancement de l'API
52 if __name__ == '__main__':
53     app.run(debug=True, host='0.0.0.0')
54

```

**student\_age.json** qui est un fichier JSON et il contient un dictionnaire avec les noms des étudiants et leurs âges correspondant

```

1  {
2      "Yague Kebe": "24",
3      "Astou Gueye": "23",
4      "Zakaria Sidibe": "24",
5      "Aissatou Cisse": "23",
6      "Khady Fall": "23",
7      "Ndeye Gnilane Ndour": "22",
8      "Fatou Ndiaye": "23",
9      "Binta Ndiaye": "23",
10     "Omar Louis Kao": "24",
11     "Mamadou Bamba Dieye": "24",
12     "Falou Ndiaye": "24",
13     "Ndoumbé Guaye": "23"
14 }
15
16

```

## Dossier Web:

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

```

<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
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>Age</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 Windows  
Accédez aux paramètres

Et en fin le fichier docker-compose.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

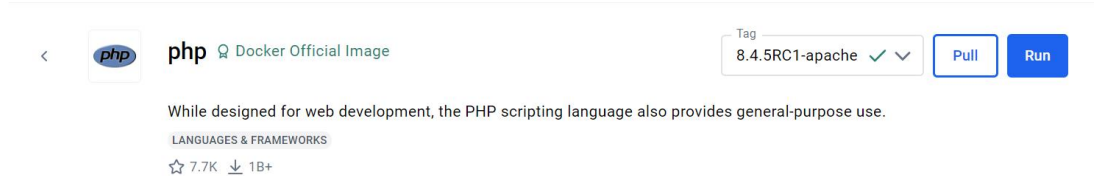
Connecté au réseau student\_network

Redémarre automatiquement sauf arrêt manuel

- [Service Web \(web\)](#)

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

[Docker Hub](#) / php



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

```
docker-compose.yml
1  version: '3.8'
2
3  > Run All Services
4  services:
5    > Run Service
6    api:
7      build: ./api
8      container_name: student_api
9      ports:
10       - "5000:5000"
11      volumes:
12       - ./api/student_age.json:/data/student_age.json
13      networks:
14       - student_network
15      restart: unless-stopped
16
17    > Run Service
18    web:
19      image: php:8.4.5RC1-apache
20      environment:
21       - USERNAME=toto
22       - PASSWORD=python
23      volumes:
24       - ./web/index.php:/var/www/html/index.php
25
26      ports:
27       - "8080:80"
28      depends_on:
29       - api
30      networks:
31       - student_network
32      restart: unless-stopped
33
34  networks:
35    student_network:
36      driver: bridge
```

**Déploiement des conteneurs :** student\_api → Pour notre API.

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

```
PS C:\Users\aisa\OneDrive\Bureau\projet docker>
>> docker-compose up -d
>>
time="2025-03-09T00:10:00Z" level=warning msg="C:\\Users\\aisa\\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
 ✓ Container projetdocker-web-1 Running
PS C:\Users\aisa\OneDrive\Bureau\projet docker>
```

**Exécuter docker pour voir si c'est réussi**

```
PS C:\Users\aisa\OneDrive\Bureau\projet docker> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NA
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\aisa\OneDrive\Bureau\projet docker>
```

On peut aussi voir les images dans notre **docker Desktop**

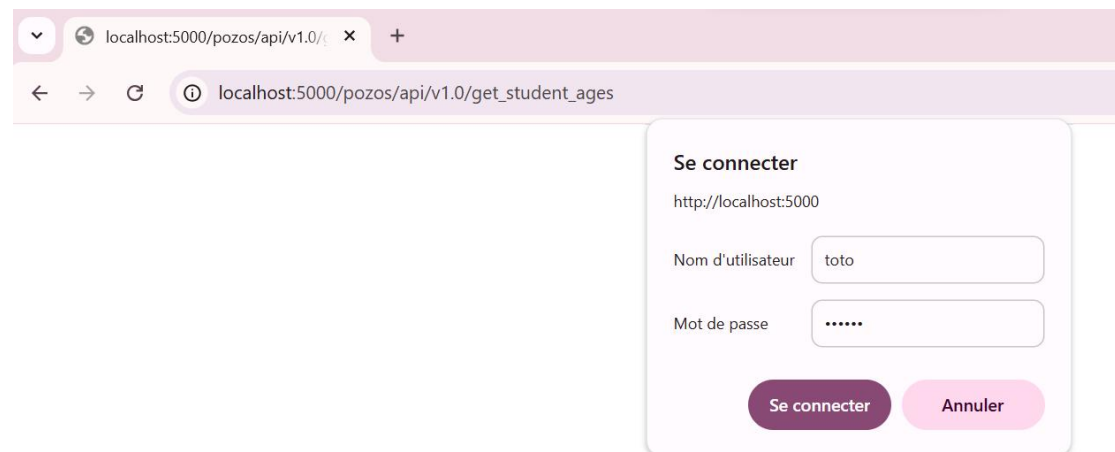
LocalHub repositories

818.92 MB / 0 Bytes in use 4 imagesLast refresh: 13 hours ago

<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	python	3.9-slim	d1fd80755520	3 months ago	189.17 MB	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	student_api	latest	c1cc9eb16780	13 hours ago	482.52 MB	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	php	8.4.5RC1-apache	77a8714f2b78	9 days ago	724.86 MB	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	projetdocker-api	latest	0525c1c6579b	13 hours ago	482.52 MB	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

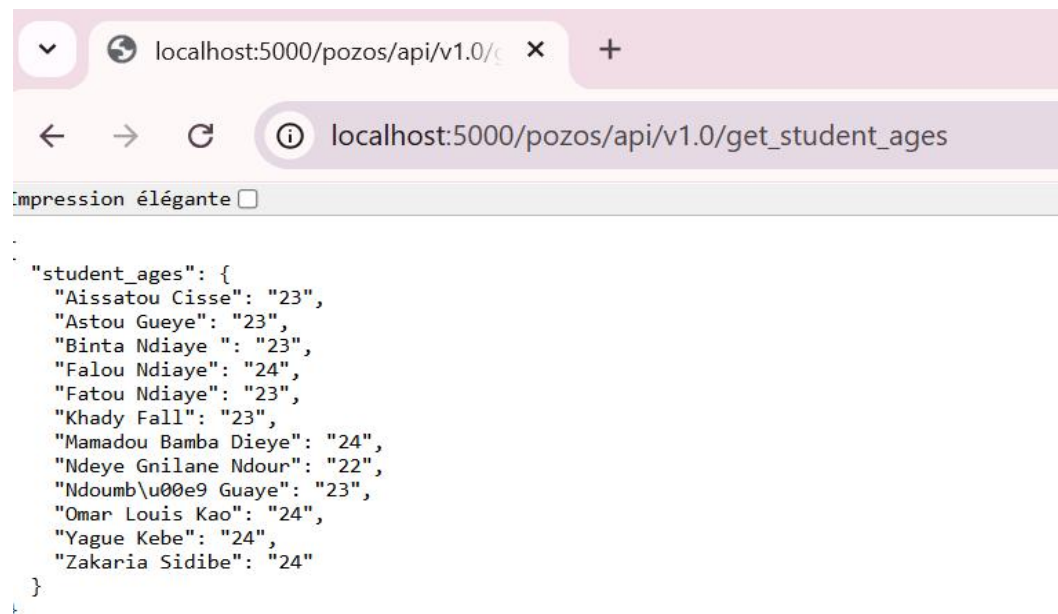
Maintenant que nous avons fini nous allons tester si l'application fonctionne

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

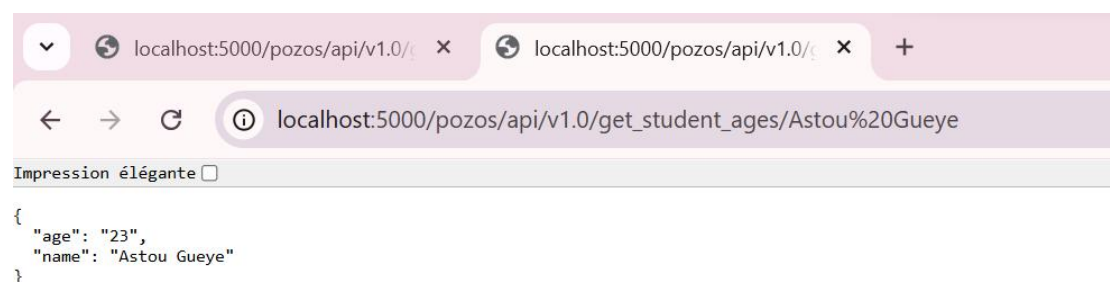




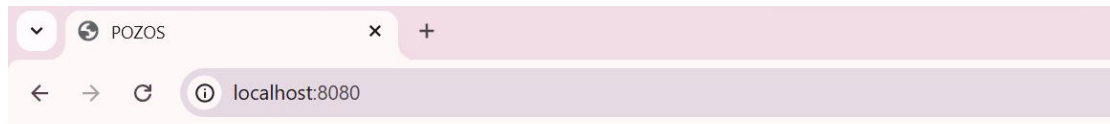
Ici nous avons accès et on voit les données que nous avons définies dans notre fichier JSON c'est à dire les noms des étudiants et leurs âges.



Dans notre fichier `students_age.py` nous avons spécifié une route pour savoir l'âge d'un étudiant donné par exemple j'ai filtré pour voir l'âge de l'étudiant nommé Astou Gueye



On voit bien que notre API REST fonctionne parfaitement. Vérifions la partie WEB dans la partie web nous avons image PHP Apache et nous avons aussi exposé le port 8080. Donc dans notre navigateur on va taper `localhost:8080` pour voir la liste des étudiants.



## Application de vérification des étudiants

Liste des étudiants

Voici la liste des étudiants :

Nom de l'étudiant	Âge
Aissatou Cisse	23
Astou Gueye	23
Binta Ndiaye	23
Falou Ndiaye	24
Fatou Ndiaye	23
Khady Fall	23
Mamadou Bamba Dieye	24
Ndeye Gnillane Ndour	22
Ndombé Guaye	23
Omar Louis Kao	24
Yague Kebe	24
Zakaria Sidibe	24

## registre privé

- [un registre](#) Docker

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

```
PS C:\Users\aiissa> docker run -d -p 5002:5000 --name registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
6d464ea18732: Download complete
44cf07d57ee4: Download complete
3493bf46cdec: Download complete
8e82f80af0de: Download complete
bbbdd6c6894b: Download complete
Digest: sha256:a3d8aaa63ed8681a604f1dea0aa03f100d5895b6a58ace528858a7b332415373
Status: Downloaded newer image for registry:2
8a6959047729042cc36681d2c47ead9f8093b5b500175e945215c0adbc9d275e
```

### 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`



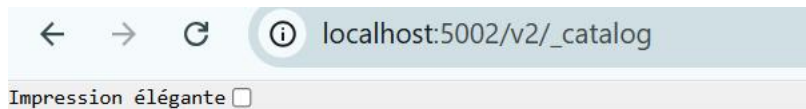
```

PS C:\Users\aisa> docker tag student_api localhost:5002/student-api
PS C:\Users\aisa> 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\aisa>

```

### Test de l'accès au registre



Impression élégante ☐

```
{"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\aisa> 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
e57ebb9e2067: Already exists
a3a550dcd386: Already exists
cf7a173b6dfb: Already exists
c306db4532fb: Already exists
5a4fd72e702: Already exists
Digest: sha256:5594b76bf8dd9de479648e28f38572d020d260568be40b7e52b9758b442275e1
Status: Downloaded newer image for joxit/docker-registry-ui:latest
bbabd1721ba436e77eaa64baddd5384ee900e4758b5c2105e8b56f0530240732
PS C:\Users\aisa>

```

### Ouvre ensuite http://localhost:8082

