

ECOLE SUPERIEURE D'INFORMATIQUE SALAMA

République Démocratique Du Congo

Province du Haut-Katanga

Lubumbashi

www.esisalama.org



COURS DE PROGRAMMATION RESEAUX

SUJET : SYSTEME DE RESERVATION DE BILLET D'AVION

Par :

- **KABUYA KAZADI Gaël**
- **MWABU KYOMBA Ghomer**
- **MULANGANI MAMBWE Victoire**
- **LUKENDO SALUT Samuel**

Demandé par : **Prof Blaise ANGOMA**

Filière : **M2 RESEAUX**

JUILLET 2023

TABLE DES MATIERES

Table des matières	0
LISTE DES FIGURES	0
I. Introduction	1
II. Contenu	2
III. Test de l'application	4

LISTE DES FIGURES

Figure 1 Code du serveur	3
Figure 2 Code du client	3
Figure 3 Les ressources de l'application.....	4
Figure 4 demmarage du serveur	4
Figure 5 IDE de python	5
Figure 6 Affichage de tous les vols	7
Figure 7 Affiche du vol avec l'ID 2.....	8
Figure 8 Ajout d'un vol 1.....	9
Figure 9 Ajout d'un vol.....	10
Figure 10 Liste des vols	11

I. INTRODUCTION

La programmation réseau est un domaine de l'informatique qui se concentre sur la création d'applications qui permettent la communication entre différents ordinateurs et périphériques via un réseau, tel que l'internet. Cela implique la mise en place de protocoles de communication, la gestion des connexions et des échanges de données, ainsi que la résolution de problèmes liés à la sécurité, à la fiabilité et à la performance des applications réseau.

La programmation réseau est essentielle pour de nombreuses applications modernes, telles que les applications web, les applications mobiles, les jeux en ligne, les services de communication en temps réel et les systèmes de stockage et de traitement de données distribués. En utilisant des langages de programmation tels que Java, Python, C# et JavaScript, les développeurs peuvent créer des applications réseau performantes et fiables qui répondent aux besoins de leurs utilisateurs et de leurs clients.

Quant à la programmation des applications RESTful, elle se concentre sur la création d'interfaces de programmation d'applications (API) qui utilisent les protocoles du web pour permettre la communication entre différentes applications. Les API RESTful sont basées sur une architecture web qui utilise des méthodes HTTP standard comme GET, POST, PUT et DELETE pour permettre aux développeurs de créer des applications qui peuvent interagir avec d'autres applications de manière cohérente et prévisible.

En combinaison, la programmation réseau et la programmation des applications API RESTful permettent de créer des applications distribuées et connectées au web, ce qui ouvre la voie à une multitude de possibilités pour les développeurs et les entreprises. Les applications créées avec la programmation réseau et les API RESTful peuvent être utilisées pour la communication entre différentes plateformes, la collecte de données à partir de différentes sources, la création de services web et bien plus encore.

Dans ce présent travail, nous présentons la conception d'une application client-serveur de réservation de billet d'avion utilisant les 4 méthodes GET, POST, PUT et DELETE.

II. CONTENUE

Notre application est un système de réservation de billets d'avion. La fonctionnalité CRUD est utilisée pour gérer les informations sur les vols, les passagers et les réservations. Notre système utilise les 4 méthodes HTTP à savoir :

- La méthode POST (Create) : Les agences de voyage et les compagnies d'aviations utilisent l'interface de l'application pour ajouter de nouveaux vols à la base de données, avec des informations telles que la compagnie aérienne, l'heure de départ et d'arrivée, le nombre de sièges disponibles, etc. Il ajoute aussi les nouveaux passagers avec leurs informations personnelles : nom, téléphone, courriel.
- La méthode GET (Read) : Les utilisateurs de l'application peuvent utiliser l'interface pour rechercher des vols disponibles en fonction de leur destination, de leur date de départ, de leur date de retour, etc. Les utilisateurs peuvent également trouver des informations sur les passagers enregistrés pour un vol spécifique.
- La méthode PUT (Update) : Les agences de voyage utilisent l'interface de l'application pour mettre à jour les informations des vols existants et les informations personnelles des passagers, telles que leur numéro de téléphone ou leur adresse de courriel.
- La méthode DELETE (Delete) : Les agences de voyage peuvent utiliser l'interface de l'application pour supprimer des vols de la base de données, par exemple lorsque le vol est annulé. De même, les agents peuvent supprimer des passagers de la base de données, par exemple lorsque le passager annule sa réservation.

Nous avons créé deux fichiers : un fichier `serveur_gestion_vols.py` qui est l'application serveur et un fichier `client_avec_interface.py` qui est une application client. Voici un bref aperçu du contenu des deux fichiers :

```

1 from flask import Flask, jsonify, request
2
3 app = Flask(__name__)
4
5 # Initialisation des données
6 flights = {
7     1: {'id': 1, 'compagnie': 'congo_airways', 'depart': 'kinshasa', 'arrivée': 'likasi'},
8     2: {'id': 2, 'compagnie': 'caa', 'depart': 'kolwezi', 'arrivée': 'lubumbashi'}
9 }
10
11 passengers = {
12     1: {'id': 1, 'name': 'victoire', 'email': 'victoire@esis.com', 'phone': '1001', 'flight_id': 1},
13     2: {'id': 2, 'name': 'gomer', 'email': 'gomer@esis.com', 'phone': '1002', 'flight_id': 2}
14 }
15
16 # Routes pour les vols
17 @app.route('/flights', methods=['GET'])
18 def get_flights():
19     return jsonify(flights)
20
21 @app.route('/flights/<int:flight_id>', methods=['GET'])
22 def get_flight(flight_id):
23     if flight_id in flights:
24         return jsonify(flights[flight_id])
25     else:
26         return jsonify({'error': 'Le vol n a pas été trouvé'})

```

127.0.0.1 - - [31/Jul/2023 08:48:15] "GET /flights HTTP/1.1" 200 -

Figure 1 Code du serveur

```

1
2
3 import requests
4 import tkinter as tk
5 from tkinter import *
6
7
8 #-----
9
10
11 # Fonctions pour effectuer les actions sur les vols et les passagers
12 def get_flights():
13     response = requests.get('http://localhost:5000/flights')
14     flights = response.json()
15     result_text.delete(1.0, tk.END)
16     flight_id_entry.delete(0, tk.END)
17     airline_entry.delete(0, tk.END)
18     departure_entry.delete(0, tk.END)
19     arrival_entry.delete(0, tk.END)
20     result_text.insert(tk.END, f"Liste des vols : {flights}")
21
22
23 def get_flight():
24     # flightsuite de la réponse :
25
26     flight_id = flight_id_entrv.eet()

```

127.0.0.1 - - [31/Jul/2023 08:48:15] "GET /flights HTTP/1.1" 200 -

Figure 2 Code du client

Nous avons préalablement installé les bibliothèques flask et la bibliothèque Tkinter car notre application client fonctionne avec une interface graphique.

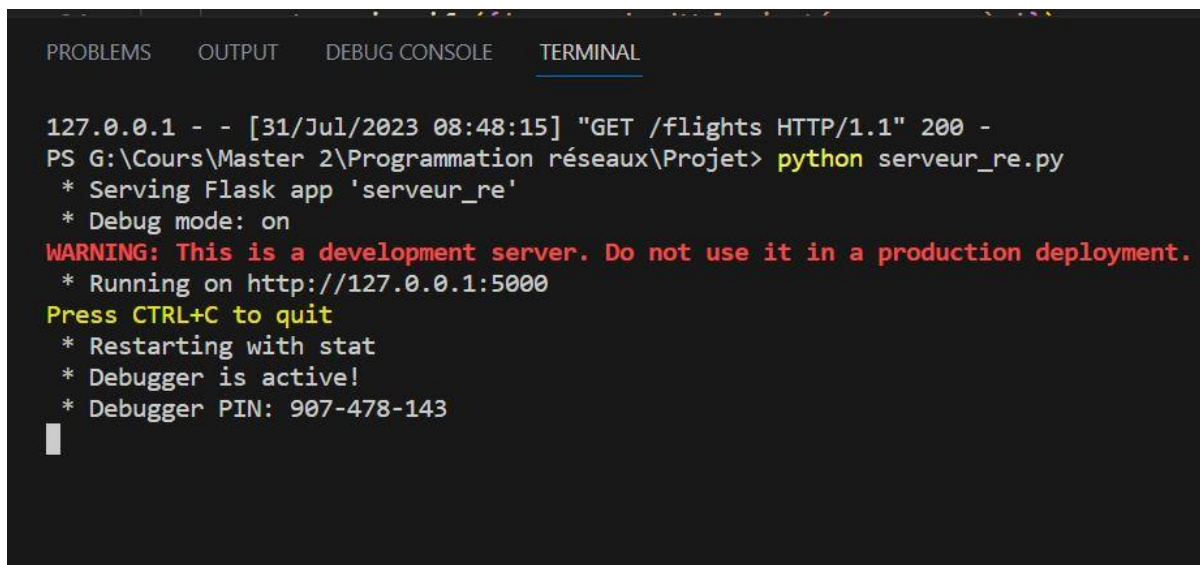
Au niveau du serveur nous avons créé deux ressources pour notre application : la ressource flights et la ressource passagers qui sont stockées dans des tables sous format JSON.

```
5  # Initialisation des données
6  flights = {
7      1: {'id': 1, 'compagnie': 'congo_airways', 'depart': 'kinshasa', 'arrivée': 'likasi'},
8      2: {'id': 2, 'compagnie': 'caa', 'depart': 'kolwezi', 'arrivée': 'lubumbashi'}
9  }
10
11  passengers = {
12      1: {'id': 1, 'name': 'victoire', 'email': 'victoire@esis.com', 'phone': '1001', 'flight_id': 1},
13      2: {'id': 2, 'name': 'gomer', 'email': 'gomer@esis.com', 'phone': '1002', 'flight_id': 2}
14  }
```

Figure 3 Les ressources de l'application

III. TEST DE L'APPLICATION

Une fois que nous exécutons notre application serveur celui se lance sur port 500 de notre machine locale.

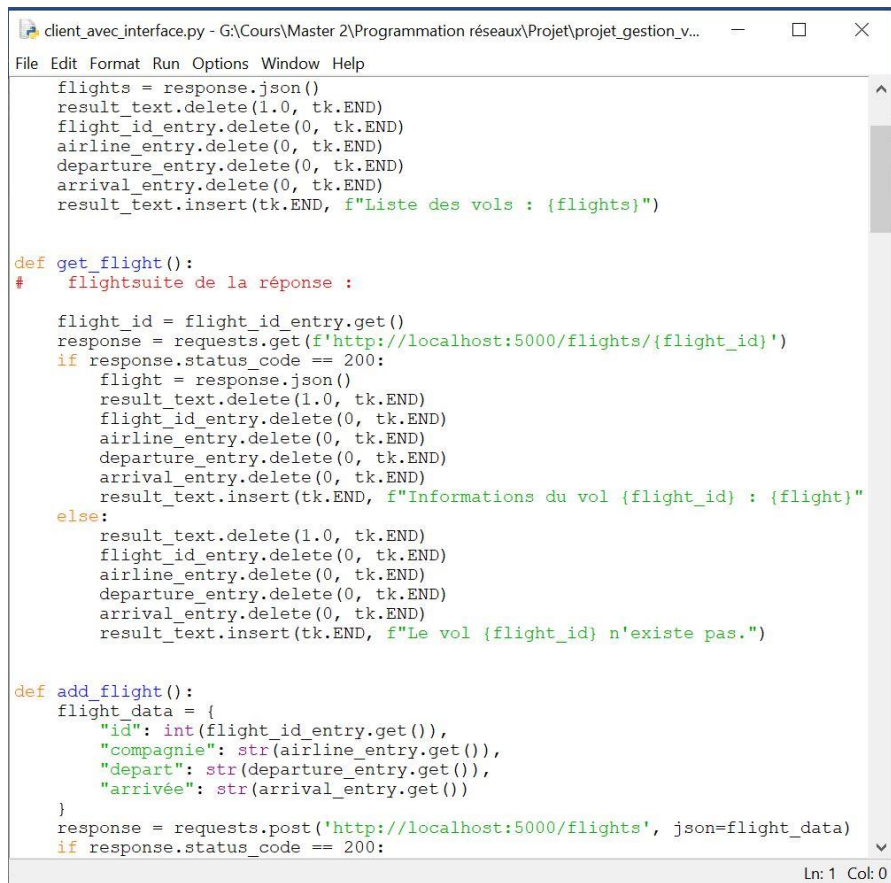


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

127.0.0.1 - - [31/Jul/2023 08:48:15] "GET /flights HTTP/1.1" 200 -
PS G:\Cours\Master 2\Programmation réseaux\Projet> python serveur_re.py
* Serving Flask app 'serveur_re'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 907-478-143
```

Figure 4 démarrage du serveur

On ouvre notre code client avec l'IDE de python et on le lance aussi

The image shows a screenshot of a Python IDE window titled 'client_avec_interface.py - G:\Cours\Master 2\Programmation réseaux\Projet\projet_gestion_v...'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is written in Python and uses the 'requests' library to interact with a web API. It includes functions for displaying a list of flights, getting flight details, and adding a new flight. The code uses Tkinter for the GUI, with labels like 'flight_id_entry', 'airline_entry', 'departure_entry', 'arrival_entry', and 'result_text'. The status bar at the bottom indicates 'Ln: 1 Col: 0'.

```
client_avec_interface.py - G:\Cours\Master 2\Programmation réseaux\Projet\projet_gestion_v...
File Edit Format Run Options Window Help

flights = response.json()
result_text.delete(1.0, tk.END)
flight_id_entry.delete(0, tk.END)
airline_entry.delete(0, tk.END)
departure_entry.delete(0, tk.END)
arrival_entry.delete(0, tk.END)
result_text.insert(tk.END, f"Liste des vols : {flights}")

def get_flight():
#    flightsuite de la réponse :

    flight_id = flight_id_entry.get()
    response = requests.get(f'http://localhost:5000/flights/{flight_id}')
    if response.status_code == 200:
        flight = response.json()
        result_text.delete(1.0, tk.END)
        flight_id_entry.delete(0, tk.END)
        airline_entry.delete(0, tk.END)
        departure_entry.delete(0, tk.END)
        arrival_entry.delete(0, tk.END)
        result_text.insert(tk.END, f"Informations du vol {flight_id} : {flight}")
    else:
        result_text.delete(1.0, tk.END)
        flight_id_entry.delete(0, tk.END)
        airline_entry.delete(0, tk.END)
        departure_entry.delete(0, tk.END)
        arrival_entry.delete(0, tk.END)
        result_text.insert(tk.END, f"Le vol {flight_id} n'existe pas.")

def add_flight():
    flight_data = {
        "id": int(flight_id_entry.get()),
        "compagnie": str(airline_entry.get()),
        "depart": str(departure_entry.get()),
        "arrivée": str(arrival_entry.get())
    }
    response = requests.post('http://localhost:5000/flights', json=flight_data)
    if response.status_code == 200:
```

Ln: 1 Col: 0

Figure 5 IDE de python

Après avoir exécuté on obtient l'interface ci-après :

The screenshot shows a web application window titled "Gestion de vols et de passagers". It features two main sections: "Vols" and "Passagers".

Vols Section:

- Form fields: ID de vol, Compagnie aérienne, Aéroport de départ, Aéroport d'arrivée.
- Buttons: Liste des vols, Ajouter un vol, Supprimer un vol, Informations d'un vol, Modifier un vol.

Passagers Section:

- Form fields: ID de passager, Nom, Email, Téléphone, ID de vol.
- Buttons: Liste des passagers, Ajouter un passager, Supprimer un passager, Informations d'un passager, Modifier un passager.

On a deux parties dans notre application : la partie vols et la partie passagers ; de chaque côté on a 5 boutons :

- **Liste des vols** : qui permet d'afficher la liste de tous les vols disponibles par l'envoi d'une requête GET
- **Information d'un vol** : qui permet d'afficher le résultat d'un vol à partir de son ID toujours grâce à la requête GET
- **Ajouter un vol** : qui permet d'ajouter un vol dans notre table grâce à la requête POST
- **Supprimer un vol** : qui permet la suppression d'un vol par l'envoi d'une requête DELETE
- **Modifier un vol** : qui permet de modifier les informations d'un vol grâce à la requête PUT.

Le même principe est appliqué pour la partie Passagers

- Résultat lorsqu'on appuie sur le bouton liste des vols :

Gestion de vols et de passagers

Vols

ID de vol :

Compagnie aérienne :

Aéroport de départ :

Aéroport d'arrivée :

Liste des vols

Ajouter un vol

Supprimer un vol

Informations d'un vol

Modifier un vol

Passagers

ID de passager :

Nom :

Email :

Téléphone :

ID de vol :

Liste des passagers

Ajouter un passager

Supprimer un passager

Informations d'un passager

Modifier un passager

Liste des vols : {'1': {'arrivée': 'likasi', 'compagnie': 'congo_airways', 'depart': 'kinshasa', 'id': 1}, '2': {'arrivée': 'lubumbashi', 'compagnie': 'caa', 'depart': 'kolwezi', 'id': 2}}

Figure 6 Affichage de tous les vols

- Résultat lorsqu'on appuie sur le bouton *information d'un vol* en entrant sur le champ ID « 2 »

Gestion de vols et de passagers

Vols

ID de vol :

Compagnie aérienne :

Aéroport de départ :

Aéroport d'arrivée :

Liste des vols

Ajouter un vol

Supprimer un vol

Informations d'un vol

Modifier un vol

Passagers

ID de passager :

Nom :

Email :

Téléphone :

ID de vol :

Liste des passagers

Ajouter un passager

Supprimer un passager

Informations d'un passager

Modifier un passager

Informations du vol 2 : {'arrivée': 'lubumbashi', 'compagnie': 'caa', 'depart': 'kolwezi', 'id': 2}

Figure 7 Affiche du vol avec l'ID 2

- On essaye d'ajouter un nouveau vol dans le système :

Gestion de vols et de passagers

Vols

ID de vol :

Compagnie aérienne :

Aéroport de départ :

Aéroport d'arrivée :

Liste des vols

Ajouter un vol

Supprimer un vol

3

ethiopian

ndola

lubumbashi

Informations d'un vol

Modifier un vol

Passagers

ID de passager :

Nom :

Email :

Téléphone :

ID de vol :

Liste des passagers

Ajouter un passager

Supprimer un passager

Informations d'un passager

Modifier un passager

Figure 8 Ajout d'un vol 1

Gestion de vols et de passagers

Vols

ID de vol :

Compagnie aérienne :

Aéroport de départ :

Aéroport d'arrivée :

Liste des vols

Ajouter un vol

Supprimer un vol

Informations d'un vol

Modifier un vol

Passagers

ID de passager :

Nom :

Email :

Téléphone :

ID de vol :

Liste des passagers

Ajouter un passager

Supprimer un passager

Informations d'un passager

Modifier un passager

Le vol a été ajouté avec succès. ID du vol :{3}

Figure 9 Ajout d'un vol

On appuie de nouveau sur liste des vols pour vérifier si la table a été mise à jour :

Gestion de vols et de passagers

Vols

ID de vol :

Compagnie aérienne :

Aéroport de départ :

Aéroport d'arrivée :

Liste des vols

Ajouter un vol

Supprimer un vol

Informations d'un vol

Modifier un vol

Passagers

ID de passager :

Nom :

Email :

Téléphone :

ID de vol :

Liste des passagers

Ajouter un passager

Supprimer un passager

Informations d'un passager

Modifier un passager

Liste des vols : { '1': { 'arrivée': 'likasi', 'compagnie': 'congo_airways', 'depart': 'kinshasa', 'id': 1 }, '2': { 'arrivée': 'lubumbashi', 'compagnie': 'caa', 'depart': 'kolwezi', 'id': 2 }, '3': { 'arrivée': 'lubumbashi', 'compagnie': 'ethiopian', 'depart': 'ndola', 'id': 3 } }

Figure 10 Liste des vols

Le principe est le meme pour les autres boutons ainsi que pour la partie passagers.