TP 2 API : Créer ma 1ère API

Objectif

Apprendre à créer une API RESTful avec Flask qui permet de gérer une collection de données de type "livres".

Prérequis

- Connaissance de base de Python
- Connaissance de base des requêtes HTTP

Matériel

- Un ordinateur avec Python installé
- Un éditeur de texte (VSCode, PyCharm, etc.)

Modalités pédagogiques :

Travail individuel

Durée : 2 jours

Étapes pour construire l'API

Partie 1 : Installation et Configuration de Flask

1. **Installer Flask**

   ```
   pip install flask
   ```

   Questions :

   - Pourquoi devons-nous installer Flask pour créer une API ?
   - Quelle est la commande pour vérifier que Flask a été installé correctement ?

Aperçu sur la structure du projet :
Voici comment organiser votre projet en utilisant une structure de dossiers appropriée :

```
my_api/ ├── app.py └── books.py
```

Questions :

- Quelle est l'importance d'une structure de projet bien organisée ?
- Pourquoi créons-nous un fichier `app.py` et un fichier `books.py` séparément ?

3. **Créer le dossier du projet :**
   Ouvrez PyCharm et créez un nouveau projet nommé “my_api”.
4. **Créer les fichiers : `app.py` et `books.py`**
   Dans le dossier “my_api”, créez deux fichiers Python nommés `app.py` et `books.py`.

Partie 2 : Contenu du Fichier `books.py`

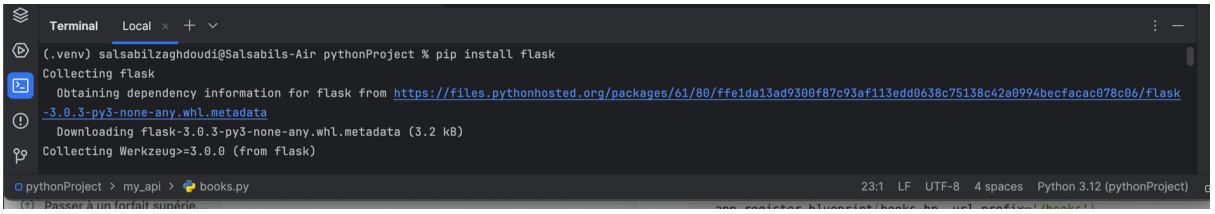
Créez et configurez le fichier `books.py` avec les données des livres et le blueprint :

```
from flask import Blueprint, jsonify

books_bp = Blueprint('books', __name__)

books = [
    {
        'id': 1,
        'title': '1984',
        'author': 'George Orwell',
        'published': 1949
    },
    {
        'id': 2,
        'title': 'To Kill a Mockingbird',
        'author': 'Harper Lee',
        'published': 1960
    }
]
```

```
@books_bp.route('/', methods=['GET'])
def get_books():
    return jsonify(books)
```

NB:

- Le fichier `books.py` doit contenir les données des livres et le blueprint que vous avez défini. Ensuite, vous allez exécuter `app.py`, qui va automatiquement utiliser les données et le blueprint définis dans `books.py`.
- Si PyCharm ne reconnait pas flask vous devez l'installer via le terminal virtuel de PyCharm :



Partie 3 Créez `app.py`

Assurez-vous que `app.py` contient le code suivant :

```
from flask import Flask
from books import books_bp

app = Flask(__name__)

# Enregistrer le blueprint des livres avec le préfixe /book
s
app.register_blueprint(books_bp, url_prefix='/books')

@app.route('/')
def home():
    return "Bienvenue à l'API de gestion des livres !"
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Questions :

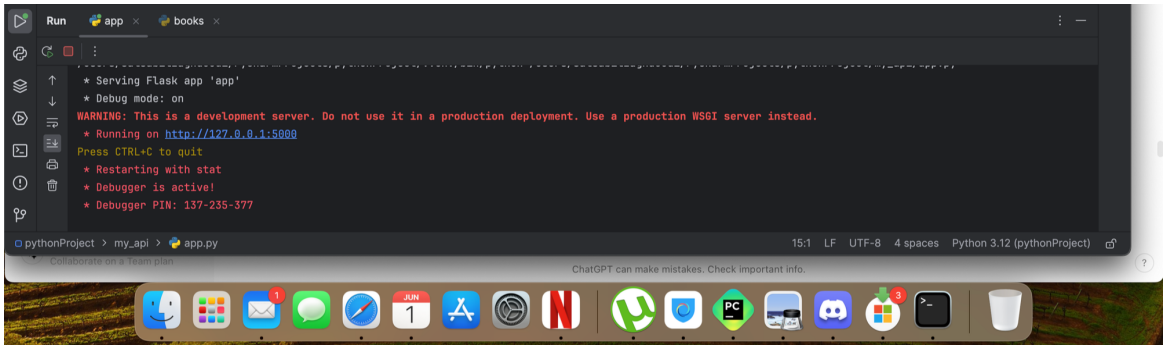
- Expliques les rôles des codes dans books.py et app.py
- Que signifie un blueprint ?
- Que signifie flask ?

Partie 4: Exécuter `app.py` dans PyCharm :

- "Run app.py et analysez les résultats obtenus.

Partie 5: Vérifier que le serveur est en cours d'exécution :

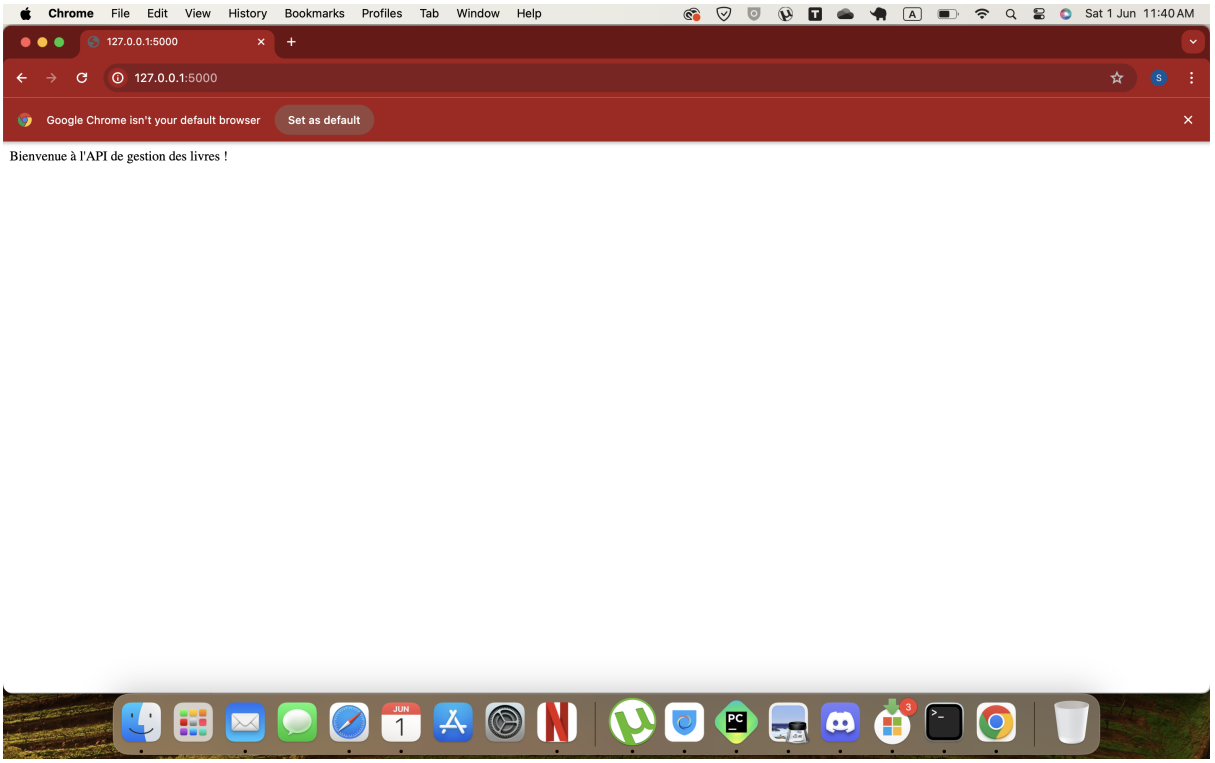
- Une fenêtre de terminal intégrée devrait s'ouvrir au bas de PyCharm, indiquant que le serveur Flask est en cours d'exécution.
- Vous devriez voir un message comme :

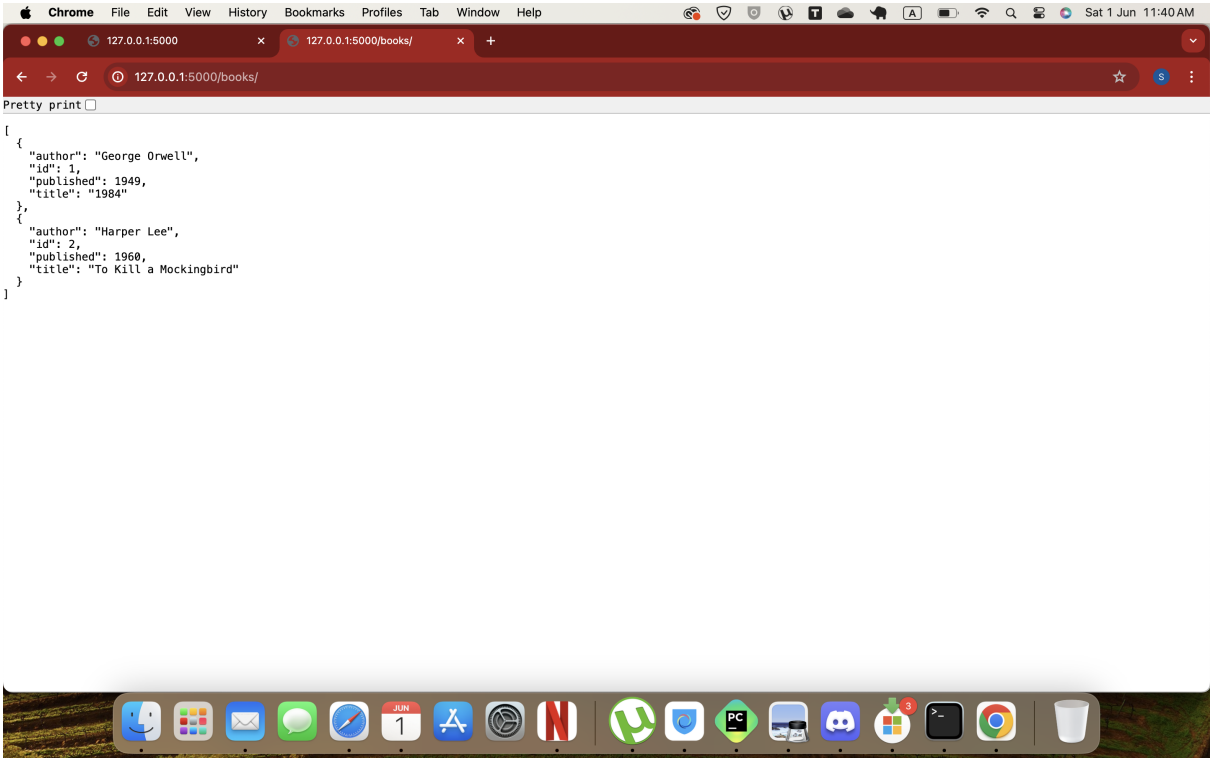


Partie 6: Tester l'API :

- Ouvrez un navigateur web et accédez à `http://127.0.0.1:5000/`. Vous devriez voir le message "Bienvenue à l'API de gestion des livres !".
- Pour accéder à la liste des livres, allez à `http://127.0.0.1:5000/books/`. Vous devriez voir la liste des livres au format JSON.

Résultat attendu :





Questions supplémentaires pour l'API des livres :

1. **Ajout de livres :**
    - Quelle méthode HTTP utilisons-nous pour ajouter un nouveau livre à la collection ?
    - Écrivez un exemple de route et de fonction Flask pour ajouter un nouveau livre à la collection.
    - C’est quoi une route ? Que signifie la fonction flask ?

2. **Suppression de livres :**
    - Quelle méthode HTTP utilisons-nous pour supprimer un livre de la collection ?
    - Écrivez un exemple de route et de fonction Flask pour supprimer un livre par son ID.

3. **Mise à jour des livres :**
    - Quelle méthode HTTP utilisons-nous pour mettre à jour les informations d'un livre existant ?
    - Écrivez un exemple de route et de fonction Flask pour mettre à jour les informations d'un livre par son ID.

3. **Recherche de livres :**
    - Comment pouvons-nous implémenter une recherche pour trouver des livres par titre ou par auteur ?
    - Écrivez un exemple de route et de fonction Flask pour rechercher des livres par titre.

4. **Gestion des erreurs :**
    - Comment gérez-vous les erreurs, comme lorsqu'un livre avec un ID spécifique n'existe pas ?
    - Écrivez un exemple de gestion d'erreur pour une tentative de suppression d'un livre inexistant.

Livrables :

Réponses aux questions avec des screenshots