PROG2 - API | TD FastAPI + Postman

Objectifs académiques :

- Prendre en main le framework Fast API pour la création d'un webservice (API REST)
- Savoir manipuler les requêtes HTTP et les réponses HTTP à travers le service

TD1 : Prise en main de Fast API | 4 & 7 juillet 2025 - Groupe K1-K2-K3-K4-K5

Sur votre IDE (IntelliJ) :

1. Cloner le repository : https://github.com/hei-ryan/hello-world-api-python.git

2. Installer les dépendances nécessaires (recommandées automatiquement par votre IDE normalement).
   Si ce n'est pas le cas, faire manuellement l'installation des dépendances avec :
   pip install uvicorn fastapi

3. Lancer le serveur, soit à travers la commande :
   a. Si sur Linux ou MacOS : "./venv/bin/uvicorn main:app --reload" soit à travers la commande "uvicorn main:app --reload".

   b. Si sur Windows : "python .\venv\bin\uvicorn main:app –reload"

      En cas d'erreur, vous pouvez essayer de créer l'environnement virtuel avec la commande "python -m venv venv"

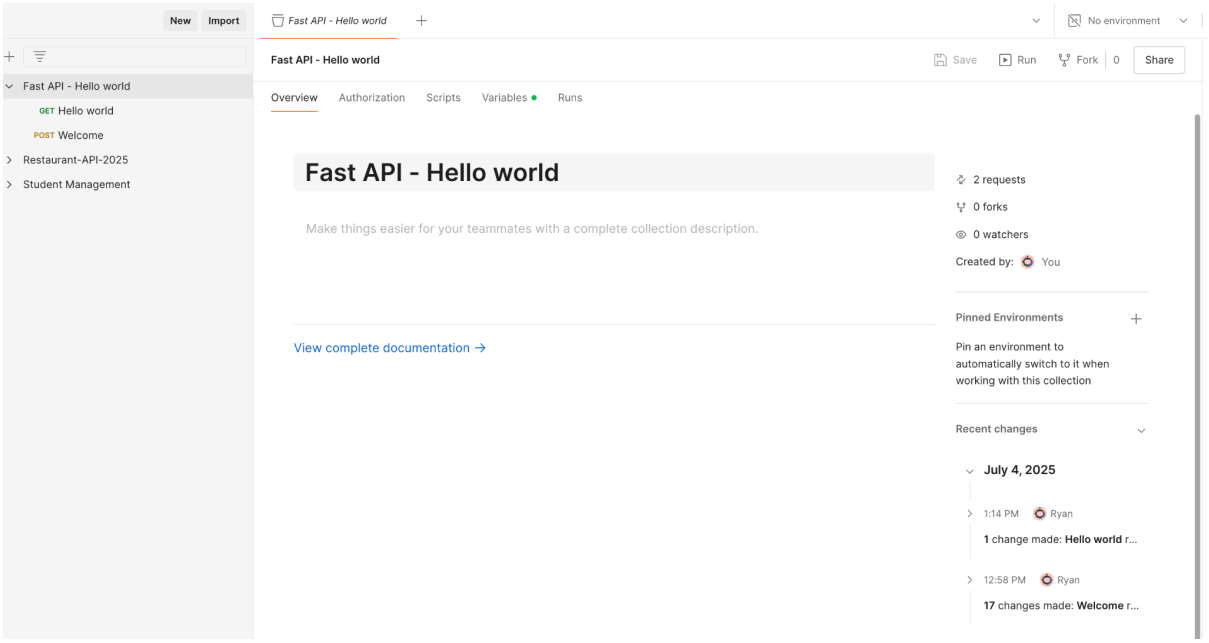
      Si la commande a été exécuté avec succès, votre terminal devrait afficher les messages suivant (ou similaire) :

```
(.venv) afryan@MacBook-Pro-de-Ryan hello-word-api-python % ./venv/bin/uvicorn main:app --reload

INFO:     Will watch for changes in these directories: ['/Users/afryan/IdeaProjects/hello-word-api-python']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [28245] using StatReload
INFO:     Started server process [28247]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Sur Postman :

1. Importer le fichier se trouvant dans /docs du projet cloné intitulé **Fast API - Hello world.postman_collection.json** en tant que collection Postman, et suivre les étapes. Vous devez tomber sur un écran similaire une fois que l'opération a été effectuée avec succès :



2. Lancez la requête GET Hello World. Quel résultat avez-vous obtenu ?

3. Lancez la requête POST Welcome. Quel résultat avez-vous obtenu ?

TD2 : Manipulation des requêtes avec Fast API | 9 & 10 juillet 2025 - Groupe K1-K2-K3-K4-K5

Documentation sur FastAPI : https://devdocs.io/fastapi/

L'objectif est de pouvoir manipuler la récupération des informations données par le client (comme Postman ou Navigateur ou tout autre application) et retourner cela dans le corps de la réponse retournée par le serveur (FastAPI).

a) **Récupération des données depuis l'URL :**

Il faut mettre en argument de la fonction les paramètres qu'on veut récupérer à travers l'URL.
Par exemple, pour l'URL suivant :
http://localhost:8000/hello?name=Ryan&is_teacher=true
On peut constater par définition que **name** c'est le **premier paramètre** car il se trouve après le caractère "?", et sa valeur ici est de type chaîne de caractère "Ryan". Et on retrouve le second paramètre **isTeacher** avec la valeur booléenne **true.**

Dans la fonction **def read_hello(request: Request)** qui permet d'exécuter les instructions lorsque le client invoque GET /hello, il suffit de mettre les deux paramètres de requête ainsi que leur type pour récupérer leurs valeurs.

Travail à faire 1 :

Tout d'abord, modifier la fonction **def read_hello** en conséquence afin de pouvoir récupérer ces données et modifier la réponse de sorte à ce que le message retourné comme ceci :
Si name=Ryan et is_teacher=true, alors le message retourné est "Hello Teacher Ryan !"
Si name=Rakoto et is_teacher=false, alors le message retourné est "Hello Rakoto !"

Indice pour la récupération, la fonction doit être modifiée comme suit mais avec les bons paramètres : **def read_hello(param_name: type)**

Testez sur Postman en ajoutant les paramètres de requête sur votre URL. Vous allez devoir dupliquer la requête GET Hello World, car vous avez deux exemples à tester, sur la première requête, vous allez ajouter les paramètres de requête name=Ryan et is_teacher=true et dans la deuxième requête que vous aurez dupliquée, vous allez ajouter les paramètres name=Rakoto et is_teacher=false.

Travail à faire 2 :

A - Veuillez noter que si vous supprimez les paramètres de requêtes préalablement ajoutées (name et is_teacher), une réponse 422 sera retournée indiquant que ces paramètres sont obligatoires, le travail consiste ainsi à les rendre facultatifs. Dans le cas où aucun des deux paramètres ne sont fournis, on reste sur le GET /hello initial,

modifiez le comportement de la fonction de sorte à retourner le message initial “Hello world”. Faites en sorte de ne pas casser le comportement lorsque les deux paramètres sont fournis.

B - Pour le cas où un seul paramètre est fourni, il faut faire en sorte d’ajouter des valeurs par défaut pour ceux qui n’ont pas été fournis.

Indice pour la récupération, la fonction doit être modifiée comme suit mais avec les bons paramètres : **def read_hello (name: str = “valeur par défaut”)**
“name” est ici le paramètre de type chaîne de caractère, (comme un String en java) et “valeur par défaut” est la valeur par défaut.

Premier exemple, si le client ne fournit que le paramètre name, alors il faut par défaut mettre la valeur de is_teacher à false, (et gérer le comportement équivalent) même si ce n’est pas présent dans l’URL. L’URL en question va être par exemple GET http://localhost:8000/hello?name=Ryan

Deuxième exemple, si le client ne fournit ne que le paramètre is_teacher, alors il faut par défaut mettre la valeur de name à “Non fourni”, (et gérer le comportement équivalent) même si ce n’est pas présent dans l’URL. L’URL en question va être par exemple GET http://localhost:8000/hello?is_teacher=true

Attention ! Il faut maintenir le comportement indiqué dans **A,** c’est à dire on ne considère les valeurs par défaut, uniquement lorsque l’un des deux paramètres sont fournis, car dans le cas où aucun n’est fourni, il ne faut pas

Testez sur Postman en ajoutant de nouvelles requêtes afin de vérifier que tout est bon.

b) **Récupération des données depuis les entêtes et le corps de la requête :**

Récupération en-tête :

Les entêtes de la requête peuvent être récupérées à travers request.headers.get(“<nom de l’entête>”), où request est un argument de type Request de la fonction invoquée lors de l’appel du chemin.

Par exemple, comme indiqué dans le code fourni, request.headers.get(“Accept”), permet de récupérer l’entête Accept envoyé par le client.

Récupération du corps de la requête :

Regarder l’exemple fourni selon **@app.post(“/welcome”)**, et essayer d’interpréter le code en vous appuyant sur la documentation présente ici : https://devdocs.io/fastapi/

Travail à faire :

A - Créer un nouveau chemin accessible à travers GET /top-secret. Une fois créé, vérifiez que les en-têtes envoyés par le client, comportent l'en-tête "Authorization" avec comme valeur "my-secret-key". Si aucune valeur n'est fournie, ou encore, si la valeur fournie ne correspond pas à "my-secret-key", alors il faut retourner une réponse avec un code de statut 403 FORBIDDEN et dans le corps de la réponse, indiquer la valeur qui a été fournie, mais qui n'a pas a été attendue.

B - Ajouter un corps de requête qui attend un JSON contenant les attributs "secret_code" de type int, qui doit comporter 4 chiffres. Si ce qui est fourni par le client est inférieur ou supérieur à 4 chiffres, il faut retourner une réponse 400 BAD_REQUEST avec comme message indiquant que le code fourni n'est pas à 4 chiffres.