


FastAPI (v0.52.0)

 FastAPI	Python-based web framework	
	Système d'exploitation :	Multiplateforme
	Développé par :	Sebastián Ramírez (auteur)
Open source, Licence MIT	Personne de contact :	katy.fokou@smals.be

Fonctionnalités

[FastAPI](#) est un outil de développement web et plus particulièrement un (*micro*)*framework* API qui permet d'implémenter rapidement et aisément une API (Application Program Interface) web en langage Python.

Traitement asynchrone : FastAPI s'appuie sur Starlette, un *framework* web [ASGI](#) (Asynchronous Standard Gateway Interface*) et sur la librairie Pydantic pour la validation des données, la sérialisation des données et la documentation. Starlette confère à FlaskAPI un avantage sur un autre *framework web* largement utilisé pour le développement d'API, Flask, qui utilise le protocole [WSGI](#) (Web Server Gateway Interface). FastAPI utilise les fonctionnalités de la librairie [asyncio](#) de Python qui permet une communication asynchrone entre serveur et application. Ceci en fait un *framework* beaucoup plus rapide que Flask. En outre, FastAPI intègre les fonctionnalités suivantes en plus de celles héritées de Flask telle que le *routing* URL :

- le protocole web WebSocket pour une communication bidirectionnelle entre serveur et application,
- les requêtes GraphQL en plus des requêtes REST,
- la sécurité et l'authentification (OAuth2).

Génération automatique de la documentation: FastAPI a été développé selon les standards [OpenAPI Specification](#) (Swagger) et intègre par défaut [SwaggerUI](#) et [ReDoc](#). SwaggerUI permet de visualiser l'API et de tester les requêtes de façon interactive ; ReDoc permet la génération automatique de la documentation. Tout outil de documentation API basé sur OpenAPI peut être utilisé avec FastAPI.

* Spécifications d'interface standard qui permettent au serveur et à l'application (Python) de communiquer

Conclusions & Recommandations

FastAPI est un *framework* intéressant pour la mise en production et l'opérabilité des applications développées en Python. Il présente de bonnes performances, est facile d'utilisation et bien documenté.

Tests et Résultats

Notre test de la librairie FastAPI consiste au développement d'une application Python qui permet d'extraire des entités (NER) d'un texte donné. Ce test a pu mettre en évidence la facilité avec laquelle il est possible de construire un API avec des notions très basiques de développement web.

```

from utils import classify

class EntityLabel(str, Enum):
    person = "PERSON"
    date = "DATE"
    gpe = "GPE"

class Entities(BaseModel):
    label: str
    entities: List[str] = []

app = FastAPI(
    title="Entity Extractor",
    version="1.0",
    description="Extract entities from a text"
)

@app.post("/ner/{entity_label}", response_model=Entities, tags=["extracted entities"])
def extract_entities(entity_label: EntityLabel = Path(..., title="Entity label"), 3
                    q: str = Query(None, max_length=200), regex="^[A-Za-z0-9]*$"):
    if q:
        doc = nlp(q)
        results = [e.text for e in doc.ents if e.label == entity_label]
        return {"label": entity_label, "entities": results}

1 @app.post("/domain/", tags=["domain"], deprecated=True) 2 deprecated path operation
1 async def classify_doc(q: str = Query(None, max_length=200), regex="^[A-Za-z0-9]*$"):
    if q:
        label = await classify(q)
        return {"entities": label}
  
```

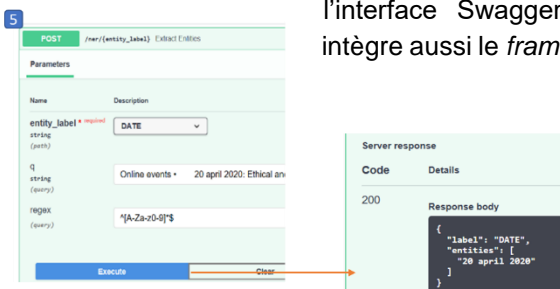
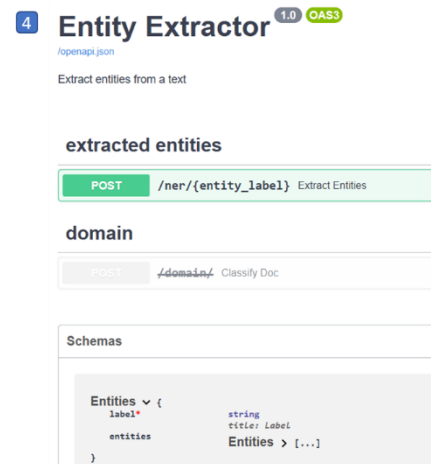
Routing : On retrouve le routing de la librairie Flask c.-à-d. des décorateurs qui permettent d'assigner un URL à une fonction ainsi que de définir la méthode de communication HTTP. La syntaxe "async def" permet l'exécution asynchrone d'une fonction [1]. De plus, il est facile de désactiver une fonction sans supprimer le chemin correspondant grâce au paramètre « deprecated » [2].

Validation des données: Il est possible de déclarer, dans la fonction, le type des paramètres du chemin. Dans l'exemple

ci-contre [3], on assigne le type *string* au paramètre *entity_label*. On définit en outre une liste de valeurs possibles pour ce paramètre. Des contraintes peuvent être ajoutées au paramètre de requête *q*. Dans cet exemple, un texte excédant le nombre maximal de caractères autorisé retournera un message d'erreur.

Documentation : L'interface interactive de Swagger permet de visualiser la documentation automatiquement générée selon les standard OpenAPI et JSON [4]. La documentation reprend les méthodes HTTP, les noms des décorateurs (call name), les paramètres, les schémas JSON de données, etc.

Testing: Le test de l'API se fait de façon interactive et intuitive via l'interface Swagger UI [5]. FastAPI intègre aussi le *framework* de test *pytest*.

Conditions d'utilisation & Budget

FlaskAPI est une open source et sous licence MIT, à utiliser dans un environnement Python (>=3.6) et requiert l'installation d'un serveur ASGI (Uvicorn).