# Formation Python Avancé : FastAPI + ORM

| | |
|---|---|
| **Durée :** | 5 jours |
| **Public :** | Développeurs Python |
| **Pré-requis :** | Avoir suivi le stage "Python : Initiation + Approfondissement" ou notions équivalentes |
| **Objectifs :** | Construire une API en Python avec FastAPI et implémenter une couche d'accès aux données avec un ORM |
| **Sanction :** | Attestation de fin de stage mentionnant le résultat des acquis |
| **Taux de retour à l'emploi:** | Aucune donnée disponible |
| **Référence:** | PYT101931-F |
| **Note de satisfaction des participants:** | Pas de données disponibles |

## Découvrir FastAPI

Présentation des Web Services (WS) : fonctionnement, intérêt, interopérabilité
Architecture orientée services (SOA) vs microservices : composantes, technologies
FastAPI : présentation, cas d'usage, architecture
FastAPI vs autres frameworks (Flask, Django)
Design et documentation : OpenApi Specification (Swagger)
Outils de test de services web : Postman

**Atelier : Installation de l'environnement de développement (VS Code + Interpréteur Python) - Création d'un projet FastAPI (structure, point d'entrée, dépendances)**

## Implémenter et interroger des services web REST

Architecture REST : composantes, méthodes d'appel (GET, POST, PUT, DELETE)
Définition de routes
Gestion des paramètres de la requête
Validation des entrées : typing, pydantic
Types de réponses, format (json, xml, texte, binaire)
Gestion des erreurs
Traitements asynchrones
Déploiement d'un service RESTful
Interrogation de web services REST (Python/Javascript)
Implémentation de tests unitaires et fonctionnels (TestClient, PyTest)
Déploiement et configuration d'une application FastAPI sur un serveur (Uvicorn, Hypercorn)

**Atelier : Création et interrogation d'une API REST avec FastAPI**

## Sécuriser une application FastAPI

Niveaux de sécurité
Gestion de l'authentification dans un web service (JWT, OpenID Connect)
Gestion des droits (OAuth2)
Multiples configurations : CORS, HTTPS, ...

**Atelier : sécurisation globale de l'application FastAPI**

### Réaliser un mapping relationnel objet (ORM)

Pattern DAO (Data Access Object)
Frameworks ORM : fonctionnalités, intérêt
ORMs Python : SQLAlchemy, Django ORM, PonyORM, SQLObject, Peewee, ...
Mapping des tables et gestion des clés primaires (simples, composées)
Mapping des types de bases, propriétés des colonnes
Gestion de la concurrence : optimistic (versioning), pessimistic
Gestion des relations : OneToMany/ManyToOne, OneToOne, ManyToMany
Paramétrage des cascades
Gestion des collections
Mapping de l'héritage
Stratégies de chargement : Lazy ou Eager

**Atelier : Réalisation d'un schéma global de mapping d'une base de données**

### Ecrire des requêtes avec un ORM

Langage de requêtes objet
Sélections de base, filtres
Jointures complexes
Fonctions d'agrégation, de chaine, ...
Gestion des chargements Lazy/Eager

**Atelier : Réalisation d'opérations CRUD (Create Read Update Delete) - requêtes complexes**

### Découvrir des fonctionnalités avancées

Cycle de vie des entités et validation
Intercepteurs, Event-listeners
Configuration avancée : performance et fonctionnalités
Utilisation du cache
Serveurs Websockets en Python

**Atelier : Implémentation d'intercepteurs et gestion du cache.**