DAWAN

Plan de la formation

www.dawan.fr

Formation Python Avancé : FastAPI + ORM

Durée :	5 jours (35 heures)
Tarifs inter-entreprise :	3 475,00 € HT (standard) 2 780,00 € HT (remisé)
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
Modalités pédagogiques, techniques et d'encadrement :	• Formation synchrone en présentiel et distanciel. • Méthodologie basée sur l'Active Learning : 75 % de pratique minimum. • Un PC par participant en présentiel, possibilité de mettre à disposition en bureau à distance un PC et l'environnement adéquat. • Un formateur expert.
Modalités d'évaluation :	• Définition des besoins et attentes des apprenants en amont de la formation. • Auto-positionnement à l'entrée et la sortie de la formation. • Suivi continu par les formateurs durant les ateliers pratiques. • Évaluation à chaud de l'adéquation au besoin professionnel des apprenants le dernier jour de formation.
Sanction :	Attestation de fin de formation mentionnant le résultat des acquis
Référence :	PYT101931-F
Note de satisfaction des participants:	Pas de données disponibles
Contacts :	commercial@dawan.fr - 09 72 37 73 73

Modalités d'accès :	Possibilité de faire un devis en ligne (www.dawan.fr, moncompteformation.gouv.fr, maformation.fr, etc.) ou en appelant au standard.
Délais d'accès :	Variable selon le type de financement.
Accessibilité :	Si vous êtes en situation de handicap, nous sommes en mesure de vous accueillir, n'hésitez pas à nous contacter à referenthandicap@dawan.fr, nous étudierons ensemble vos besoins

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.