PROGRAMME DE FORMATION

CRÉER DES API AVEC PYTHON (FASTAPI) — EXPOSER SES MODÈLES ET SCRIPTS

FICHE SYNTHÉTIQUE DE LA FORMATION

Intitulé	Créer des API avec Python (FastAPI) pour exposer ses modèles et scripts
Durée	3 jours (21 heures)
Référence	API-FASTAPI-3J
Tarif	1 950 € HT / personne
Modalité	Présentiel ou distanciel
Public	Développeurs Python, data scientists, ML engineers, profils ESN
Prérequis	Python intermédiaire (fonctions, classes, typage), notions de HTTP
Effectif	8 participants maximum
Attestation	Attestation de fin de formation Pythonia
Lieu	255 boulevard Pereire, 75017 Paris ou à distance (Zoom)

1. Public visé

Cette formation s'adresse aux développeurs Python, data scientists, ML engineers et profils techniques en ESN qui souhaitent concevoir et exposer des API robustes : exposer un modèle de machine learning, créer un backend pour une application front, fournir des services à d'autres équipes, bâtir une architecture microservices. La formation est centrée sur FastAPI, devenu le framework de référence pour les API Python modernes, avec comparaisons ponctuelles vers Flask et Django REST Framework.

2. Prérequis

- Python intermédiaire : fonctions, classes, décorateurs, gestion d'exceptions
- Bases du typage Python (type hints, Optional, Union)
- Notions de HTTP : méthodes, codes de statut, headers
- Notions de JSON et de structures de données
- Aisance avec Git et la ligne de commande

Un test de positionnement technique est proposé avant la formation pour valider le niveau Python et les acquis en HTTP.

3. Objectifs pédagogiques

À l'issue de cette formation, le stagiaire sera capable de :

- Comprendre les principes REST et les bonnes pratiques de design d'API (O1)
- Créer une API FastAPI complète : endpoints, paramètres, corps de requête (O2)
- Utiliser Pydantic pour la validation et la sérialisation des données (O3)
- Gérer les erreurs de manière cohérente avec les standards HTTP (O4)

- Documenter automatiquement son API (Swagger / OpenAPI) (O5)
- Sécuriser une API : authentification, JWT, OAuth2, clés API (O6)
- Connecter une API à une base de données (SQLAlchemy, ORM, async) (O7)
- Exposer un modèle de machine learning en production via une API (O8)
- Maîtriser le mode asynchrone de FastAPI pour la performance (O9)
- Tester une API : pytest, httpx, tests unitaires et d'intégration (O10)
- Connecter son API à un front (CORS, client JavaScript, outils low-code) (O11)
- Déployer une API FastAPI avec un serveur ASGI en production (O12)

4. Compétences visées

Code	Compétence
O1 – O2	Design REST et création d'API FastAPI
O3 – O4	Validation Pydantic et gestion d'erreurs
O5 – O6	Documentation et sécurité
O7	Connexion à une base de données
O8 – O9	Exposer des modèles ML et optimiser les performances
O10	Tests automatisés
O11 – O12	Intégration front et déploiement en production

5. Programme détaillé

JOUR 1 — Fondamentaux REST et premiers endpoints (7h)

Objectifs pédagogiques visés : O1, O2, O3, O4, O5

Matin (3h30) — REST et premier API FastAPI

- Rappels REST : ressources, méthodes HTTP, codes de statut (O1)
- Bonnes pratiques de design d'API : nommage, versioning, pagination
- Pourquoi FastAPI : comparaison avec Flask et Django REST Framework
- Installation et structure d'un projet FastAPI
- Premier endpoint GET : route, path parameters, query parameters (O2)
- Endpoints POST, PUT, DELETE : corps de requête
- Organisation du code : routers, modules, structure recommandée

Méthodes : exposé interactif, live coding progressif avec les participants.

Après-midi (3h30) — Pydantic, validation et documentation

- Introduction à Pydantic v2 : modèles, types, validation (O3)
- Validation des entrées : automatique avec type hints
- Validateurs personnalisés : field_validator, model_validator
- Réponses typées avec response_model
- Gestion des erreurs : HTTPException, handlers personnalisés (O4)
- Codes de statut HTTP : les utiliser correctement (201, 204, 400, 404, 422, 500)
- Documentation automatique OpenAPI : Swagger UI et ReDoc (O5)
- Enrichir la doc : summary, description, tags, examples

Travaux pratiques :

- TP1 : Construire une API CRUD complète pour une ressource métier (ex. catalogue produits)
- TP2 : Ajouter validation Pydantic stricte et gestion d'erreurs cohérente
- TP3 : Enrichir la documentation OpenAPI avec exemples et descriptions métier

Évaluation formative : correction collective, partage des bonnes pratiques repérées.

JOUR 2 — Base de données, sécurité et async (7h)

Objectifs pédagogiques visés : O6, O7, O9

Matin (3h30) — Connexion à une base de données

- Panorama : SQLAlchemy 2.0, SQLModel, Tortoise ORM (O7)
- Configuration d'une base PostgreSQL locale et cloud
- Mise en place de SQLAlchemy 2.0 avec FastAPI
- Dependency injection : Depends pour la session de base de données
- Opérations CRUD complètes avec une base relationnelle
- Migrations de schéma avec Alembic : principes et premier cas
- Mode asynchrone avec asyncpg et SQLAlchemy async
- Transactions et gestion d'erreurs au niveau de la base

Méthodes : live coding, construction progressive d'une API persistée.

Après-midi (3h30) — Sécurité et asynchronisme

- Authentification : pourquoi, quand, comment (O6)

- API Key : simple et efficace pour les usages internes
- OAuth2 Password Bearer : flux standard pour API publiques
- JWT (JSON Web Token) : génération, validation, refresh
- Dépendances de sécurité : patterns réutilisables
- CORS : comprendre et configurer correctement
- Async en Python : principes, quand c'est utile (O9)
- async / await dans FastAPI : gains de performance mesurables
- Tests de charge : wrk, Locust, mesure de concurrence

Travaux pratiques :

- TP1 : Connecter l'API CRUD à une base PostgreSQL avec SQLAlchemy async
- TP2 : Ajouter l'authentification JWT avec 2 rôles (utilisateur, admin)
- TP3 : Mesurer l'impact du mode async sur la performance avec Locust

Évaluation formative : QCM mi-parcours de 25 questions, revue des implémentations de sécurité.

JOUR 3 — Exposer des modèles IA, tests et déploiement (7h)

Objectifs pédagogiques visés : O8, O10, O11, O12

Matin (3h30) — Exposer un modèle IA et tests automatisés

- Exposer un modèle de ML via une API FastAPI (O8)
- Chargement du modèle au démarrage : lifespan events
- Endpoint de prédiction : validation d'entrée, réponse typée
- Cas pratique : exposer un modèle scikit-learn ou HuggingFace
- Intégration d'une API LLM externe (OpenAI, Claude) via FastAPI
- Streaming des réponses : Server-Sent Events avec FastAPI
- Tests unitaires avec pytest et TestClient (O10)
- Tests d'intégration avec httpx AsyncClient
- Fixtures, mocks, base de données de test
- Couverture de code et CI : GitHub Actions

Méthodes : construction d'une API IA complète, avec batterie de tests.

Après-midi (3h30) — Intégration front et déploiement

- Consommer l'API depuis un front JavaScript : fetch, axios (O11)
- Consommer l'API depuis un outil no-code : Make, Zapier, n8n
- Génération automatique d'un client TypeScript à partir de l'OpenAPI
- Déploiement de l'API : uvicorn, gunicorn, workers (O12)
- Dockeriser son API : Dockerfile minimaliste
- Déploiement sur Render, Railway et comparaison Azure App Service
- Configuration production : logging, gestion des secrets, health check
- Observabilité : métriques Prometheus, traces OpenTelemetry (survol)

Projet final :

- Chaque participant finalise une API FastAPI complète avec :
- — authentification JWT, base de données, au moins 5 endpoints,
- — tests automatisés couvrant les cas principaux,
- — API déployée et accessible publiquement,
- — documentation OpenAPI enrichie.
- Soutenance (15 min par participant) : présentation, démonstration, choix techniques

Évaluation sommative : QCM final de 30 questions et présentation du projet final avec démonstration live.

6. Méthodes et moyens pédagogiques

MÉTHODES MOBILISÉES

- **Apports théoriques :** exposés concis avec schémas et cas concrets
- **Live coding :** construction progressive d'une API réelle tout au long des 3 jours
- **Pédagogie par la pratique :** 70 % du temps consacré à la mise en pratique
- **Pédagogie par l'erreur :** analyse des anti-patterns fréquents en production
- **Projet fil rouge :** chaque participant construit une API complète et déployée
- **Retours personnalisés :** revue de code individuelle pendant les TP

MOYENS TECHNIQUES

- Salle de formation équipée : vidéoprojecteur, paperboard, connexion Wi-Fi haut débit
- Poste informatique par stagiaire ou ordinateur personnel avec droits admin
- Base de données PostgreSQL cloud mise à disposition pendant la formation
- Comptes Render et Railway fournis pour les déploiements
- Accès API OpenAI et Claude pour les TP d'exposition de modèles IA
- En distanciel : Zoom avec partage d'écran, salons de groupe pour le debug
- Code source, notebooks et template FastAPI production-ready remis aux stagiaires

7. Modalités d'évaluation

Avant l'entrée en formation, chaque participant complète un questionnaire de positionnement permettant d'évaluer son niveau et ses attentes.

POSITIONNEMENT À L'ENTRÉE

- Questionnaire de positionnement en ligne avant la formation
- Entretien téléphonique avec le formateur si nécessaire

ÉVALUATIONS EN COURS DE FORMATION

- **Évaluation formative :** exercices et mises en situation tout au long de la formation, corrections collectives
- **Évaluation sommative :** QCM final et présentation d'un cas pratique en dernière journée
- **Critères de réussite :** obtenir 60 % au QCM final et présenter un cas pratique fonctionnel
- **Attestation de fin de formation :** remise à l'issue de la formation, mentionnant les objectifs pédagogiques, la durée et les résultats

SUIVI POST-FORMATION

- Accès aux supports de cours et aux ressources complémentaires
- Invitation à la communauté Pythonia des anciens stagiaires

SATISFACTION

- Questionnaire de satisfaction en fin de formation (formateur, contenu, logistique, atteinte des objectifs)
- Questionnaire à froid à 3 mois sur l'impact en situation professionnelle

8. Accessibilité aux personnes en situation de handicap

Pythonia s'engage à rendre ses formations accessibles aux personnes en situation de handicap. Notre référent handicap est disponible pour étudier les aménagements nécessaires :

- Adaptation des supports pédagogiques (taille de police, contraste, format audio)
- Aménagement des conditions de passage (temps majoré, pauses adaptées)
- Orientation vers nos partenaires spécialisés : Agefiph, Cap Emploi, MDPH

Contact référent handicap : contact@pythonia.fr – 06 66 06 38 59

9. Engagement et suivi

- **Suivi de l'assiduité :** feuille d'émargement signée par demi-journée
- **Prévention des abandons :** suivi individualisé, adaptation du rythme, relance en cas d'absence
- **Réclamations :** toute réclamation peut être adressée par email à contact@pythonia.fr. Elle sera traitée sous 48h ouvrées.