PYTHON & DATA : API, INTELLIGENCE ARTIFICIELLE

Créer une API REST avec Python et FastAPI

Créez des API REST avec FastAPI : endpoints typés, Pydantic v2, SQLAlchemy et Alembic, architecture en couches, JWT et rôles, gestion des erreurs, tests pytest, asynchrone et déploiement Docker.

DURÉE	TARIF	NIVEAU	LANGUE	GROUPE	FORMAT
5j (35h)	**1 955 € / participant** Inter-entreprise. Intra : sur devis, au plus 1 955 € / participant	**Intermédiaire**	**Français**	**2-8**	**Formation** Présentiel ou à distance

1 PUBLIC VISÉ

- Développeurs Python qui connaissent la programmation orientée objet et veulent créer des API back-end
- Data scientists et ingénieurs qui veulent exposer leurs traitements ou leurs modèles par une API

2 PRÉREQUIS

- Écrire des classes Python, des décorateurs et des annotations de type
- Écrire des requêtes SQL avec des jointures
- Aucune connaissance de FastAPI n’est requise

3 OBJECTIFS PÉDAGOGIQUES

- Créer des endpoints REST typés avec FastAPI : paramètres de chemin et de requête, corps JSON, statuts HTTP et dépendances
- Valider et sérialiser les données avec Pydantic v2, avec des schémas distincts pour la création, la mise à jour et la réponse
- Déclarer des modèles SQLAlchemy reliés et connecter l’API à PostgreSQL
- Écrire les opérations CRUD avec SQLAlchemy et faire évoluer le schéma par des migrations Alembic
- Organiser l’API en couches (routeurs, services, repositories) avec une configuration par environnement
- Sécuriser l’API par jetons JWT, avec mots de passe hachés et accès selon les rôles
- Centraliser la gestion des erreurs et configurer CORS et les middlewares pour une API appelée par un front-end
- Tester les endpoints avec pytest et le client de test, sur une base isolée et avec des dépendances remplacées
- Utiliser l’asynchrone à bon escient : async def, client httpx asynchrone, tâches en arrière-plan
- Déployer l’API avec Docker Compose et PostgreSQL

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) : Premiers endpoints et Pydantic

Module 1 : Routes, paramètres et dépendances (3h30)

FastAPI et l’environnement (1h30)

- FastAPI, Starlette, Pydantic, Uvicorn ; ASGI et WSGI

- Environnement virtuel ; lancer en développement avec fastapi dev
- Structure d'un projet : routeurs, schémas, modèles, services
- La documentation automatique : Swagger UI et ReDoc
- Exercice d'entraînement : premier endpoint et sa documentation

Routes, paramètres et dépendances (2h)

- Décorateurs de route ; paramètres de chemin et de requête ; corps JSON
- status_code, HTTPException, response_model ; tags
- Les dépendances avec Depends
- Envoi de fichier avec UploadFile
- TP évalué : CRUD d'une ressource en mémoire, avec les bons statuts, des filtres, la pagination et une dépendance commune

Module 2 : Pydantic v2 (3h30)

Les bases de Pydantic v2 (1h30)

- BaseModel, Field et contraintes
- EmailStr, HttpUrl, UUID, Annotated
- model_validate et model_dump
- Exercice d'entraînement : schéma d'un produit avec contraintes

Validation avancée et schémas par opération (2h)

- @field_validator et @model_validator
- Schémas imbriqués ; model_config
- Schémas de création, de mise à jour et de réponse : ne jamais exposer les données sensibles
- TP évalué : schémas d'utilisateur distincts (création avec mot de passe, mise à jour partielle, réponse sans données sensibles) avec des validateurs personnalisés

Jour 2 (7h) : SQLAlchemy et Alembic

Module 3 : Modèles SQLAlchemy et connexion (3h30)

Configurer SQLAlchemy (1h30)

- SQLAlchemy 2 et le pilote psycopg 3 pour PostgreSQL
- Moteur et session ; la dépendance get_db
- Connexion par variables d'environnement avec pydantic-settings
- Exercice d'entraînement : connecter l'API à PostgreSQL

Déclarer les modèles et leurs relations (2h)

- DeclarativeBase, Mapped et mapped_column
- ForeignKey, relationship ; relation plusieurs-à-plusieurs
- Dates de création et de modification automatiques
- TP évalué : modèles User, Post et Category avec leurs relations

Module 4 : Opérations CRUD et migrations (3h30)

Opérations CRUD (1h30)

- add et commit ; get, select, where ; limit et offset
- Modifier et supprimer
- Exercice d'entraînement : couche CRUD réutilisable

Migrations avec Alembic (2h)

- Configurer Alembic ; générer une migration depuis les modèles
- upgrade et downgrade
- TP évalué : endpoints CRUD des articles branchés sur PostgreSQL, avec filtres et pagination, et migrations Alembic créées et appliquées

Jour 3 (7h) : Architecture et authentification

Module 5 : Architecture en couches et configuration (3h30)

Routeurs, services et repositories (1h30)

- APIRouter : découper l'application
- Couche service : la logique métier ; couche repository : l'accès aux données
- Relier les couches par l'injection de dépendances
- Exercice d'entraînement : extraire le service des articles

Configuration par environnement (2h)

- pydantic-settings : configuration typée et validée
- Développement, test, production ; mise en cache des paramètres avec lru_cache
- lifespan : initialiser et libérer les ressources
- TP évalué : refactoriser l'API en couches, avec une configuration différente selon l'environnement

Module 6 : Authentification JWT et rôles (3h30)

Authentification (1h30)

- Hacher les mots de passe avec pwdlib (Argon2)
- Créer et vérifier un JWT avec PyJWT
- /auth/register et /auth/login ; OAuth2PasswordBearer
- Exercice d'entraînement : inscription et connexion

Routes protégées et rôles (2h)

- La dépendance get_current_user ; protéger un endpoint
- Rôles USER et ADMIN ; dépendances de rôle
- Jetons de rafraîchissement : le principe
- TP évalué : endpoints protégés avec contrôle d'accès par rôle

Jour 4 (7h) : Erreurs, middlewares et tests

Module 7 : Gestion des erreurs, middlewares et CORS (3h30)

Gestion globale des erreurs (1h30)

- Gestionnaires d'exceptions globaux : un format d'erreur unique
- Erreurs de validation 422
- Exercice d'entraînement : erreur 404 au format commun

Middlewares et CORS (2h)

- CORSMiddleware pour une API appelée par un front-end
- Middleware de journalisation et de mesure du temps de réponse
- En-têtes de sécurité ; limitation de débit : le principe
- TP évalué : API prête pour un front-end React : CORS, middleware de journalisation, erreurs au même format

Module 8 : Tests avec pytest (3h30)

Tester une API FastAPI (1h30)

- conftest.py et fixtures ; TestClient
- Base de test isolée ; remplacer une dépendance avec dependency_overrides
- Exercice d'entraînement : tester les endpoints publics

Tests avancés (2h)

- Tester les endpoints protégés avec un jeton de test
- @pytest.mark.parametrize ; cas d'erreur 400, 401, 403, 404, 422
- Mesurer la couverture avec pytest-cov
- TP évalué : suite de tests des endpoints CRUD et d'authentification, cas d'erreur compris

Jour 5 (7h) : Asynchrone, déploiement et projet de synthèse

Module 9 : Asynchrone et déploiement (3h)

L'asynchrone dans FastAPI (1h30)

- async def ou def : quand choisir l'un ou l'autre
- Appels HTTP asynchrones avec httpx.AsyncClient
- Tâches en arrière-plan avec BackgroundTasks ; WebSockets en bref
- Exercice d'entraînement : envoyer un e-mail en arrière-plan sans retarder la réponse

Déployer avec Docker (1h30)

- fastapi run et plusieurs processus en production
- Dockerfile multi-stage ; Docker Compose avec PostgreSQL et Nginx ; healthcheck
- Appliquer les migrations au démarrage ; secrets dans les variables d'environnement
- TP évalué : déployer l'API avec Docker Compose et PostgreSQL (healthcheck, migrations au démarrage), avec un endpoint qui appelle une API externe en asynchrone

Module 10 : Projet de synthèse, évalué (4h)

Réalisation (2h30)

- Projet de synthèse, réalisé individuellement : API de gestion de contenu ou de boutique : couches routeur, service et repository, modèles reliés, migrations Alembic, JWT avec rôles, schémas Pydantic par opération, erreurs au même format, tests des endpoints principaux, documentation Swagger, Docker Compose

Revue et retour (1h30)

- Revue de code collective : architecture, sécurité, tests
- Retour individuel du formateur sur le projet rendu

Le contenu peut être ajusté selon le niveau, les prérequis et les besoins des participants.

5 COMPÉTENCES VISÉES

- Concevoir les endpoints d'une API FastAPI
- Gérer les données avec SQLAlchemy et Alembic
- Structurer et fiabiliser une API
- Sécuriser une API

- Tester une API
- Exploiter une API en production

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou à distance, en classe virtuelle. Le formateur alterne entre méthode démonstrative, méthode interrogative (analyse des choix d'architecture et des compromis de conception) et méthode active (exercices d'entraînement, TP évalués et projet de synthèse).

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Dépôt GitHub de démonstration avec le projet fil rouge versionné par étape
- Environnement de développement : VS Code + Python 3.12 ou plus récent + Docker, installés sur l'ordinateur du participant
- Pour la classe virtuelle : visioconférence (Microsoft Teams ou équivalent), partage d'écran, chat en direct
- Supports remis par e-mail et sur la plateforme pédagogique LaPolaris (ressources, émargement)

En formation intra-entreprise sur le site du client, celui-ci s'assure et s'engage à disposer de toutes les ressources matérielles nécessaires à son bon déroulement (salle, postes de travail, logiciels, accès à Internet), conformément aux prérequis indiqués dans le programme et aux moyens prévus dans la convention.

8 MODALITÉS D'ÉVALUATION

- Questionnaire de positionnement individuel renseigné par chaque participant avant le démarrage de la formation, analysé par le formateur et donnant lieu, le cas échéant, à un ajustement du déroulé
- Exercices d'entraînement réalisés par chaque participant sur son poste, corrigés en séance
- Évaluation des acquis par les TP évalués, réalisés individuellement sur poste : un par module, puis le projet de synthèse décrit dans le dernier module du programme ; chaque objectif est évalué dans au moins un TP
- Grille de suivi renseignée par le formateur pour chaque participant et pour chacun des objectifs de l'action, au regard des critères d'évaluation ci-dessous
- Questionnaire d'appréciation renseigné par les participants en clôture de formation, et enquête à froid dans un délai de trois à six mois

9 CRITÈRES D'ÉVALUATION

- Endpoints avec les bons verbes, statuts et paramètres typés ; dépendance partagée par plusieurs routes ; documentation Swagger générée
- Schémas distincts pour la création, la mise à jour et la réponse ; validateur personnalisé ; aucune donnée sensible dans les réponses
- Modèles reliés (un-à-plusieurs, plusieurs-à-plusieurs) ; connexion configurée par variables d'environnement ; session injectée par dépendance

- Opérations CRUD, filtres et pagination justes ; migrations Alembic générées, appliquées et annulables
- Routeurs, services et repositories séparés ; aucune requête SQL dans les routes ; configuration différente selon l'environnement
- Mots de passe hachés ; connexion qui renvoie un JWT ; routes protégées refusées sans jeton ; accès selon le rôle
- Toutes les erreurs renvoient un JSON au même format ; CORS limité aux origines prévues ; middleware de journalisation actif
- Tests qui passent sur une base isolée ; dépendances remplacées ; cas d'erreur testés ; couverture mesurée
- async def seulement là où l'opération attend une ressource ; appel externe asynchrone ; tâche lancée en arrière-plan sans retarder la réponse
- API lancée avec Docker Compose et PostgreSQL ; healthcheck ; migrations appliquées au démarrage ; secrets dans des variables d'environnement

10 INDICATEURS DE RÉSULTATS

Les indicateurs de résultats de cette formation (taux de satisfaction et taux de réussite) seront publiés sur sa fiche en ligne à l'issue des premières sessions réalisées.

11 MODALITÉS DE VALIDATION

Une attestation de fin de formation est remise à chaque participant à l'issue du parcours. Elle mentionne **l'intitulé** et la **nature de l'action**, sa **durée** en heures, sa **période de réalisation**, **les objectifs de l'action** et le **niveau d'atteinte** de chacun d'eux, exprimé en trois valeurs : **atteint**, **partiellement atteint**, **non atteint**. **Une attestation d'assiduité**, établie à partir des émargements, est délivrée dans les mêmes conditions.

Formation non certifiante : elle ne délivre pas de certification enregistrée au RNCP ou au Répertoire spécifique.

12 SUIVI ET ACCOMPAGNEMENT

- Émargement électronique signé par chaque participant et par le formateur à chaque demi-journée, en présentiel comme à distance
- Traçabilité des activités pédagogiques réalisées
- Attestation d'assiduité délivrée en fin de formation
- Suivi individuel via les QCM de fin de journée, les exercices corrigés et les TP évalués

13 CONDITIONS D'ACCÈS

Formation accessible sans prérequis administratif particulier. Cette formation n'est pas éligible aux financements publics ou mutualisés, notamment au compte personnel de formation et à une prise en charge par un opérateur de compétences.

14 DÉLAIS D'ACCÈS

Inscription possible jusqu'à 15 jours ouvrés avant le début de la session pour permettre la préparation logistique et, si besoin, l'adaptation du contenu au contexte client. Pour toute demande spécifique, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations peuvent être accessibles aux personnes en situation de handicap. La question d'un besoin d'aménagement (matériel, temporel ou pédagogique) est posée lors de l'analyse du besoin et du questionnaire de positionnement ; vous pouvez aussi nous le signaler à tout moment, y compris pendant la formation, afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris

TÉL. +33762584798

EMAIL contact@lapolaris.fr

WEB lapolaris.fr