Formation API Rest avec Python et Flask

Informations

Durée : 3 jours (21h.)

Tarif* : 1790 € HT
Intra: Nous consulter

Réf : APIR

Niveau : Moyen

inter à distance / intra

Mise à jour le 02/09/24

*tarif valable jusqu'au 31/12/2025

Prochaines sessions

22 septembre - 24 septembre
(à distance)

17 novembre - 19 novembre
(à distance)

Pré-requis

- Notions de base en développement web
- Compréhension des concepts fondamentaux de Python
- Notions de base en gestion de versions avec Git (optionnel mais recommandé)

Objectifs

Objectifs pédagogiques :

- Comprendre les concepts fondamentaux des API REST et se familiariser avec l'environnement Python
- Apprendre à utiliser Flask pour créer une API REST
- Savoir manipuler les données et gérer les erreurs dans une API
- Apprendre les bonnes pratiques de sécurité pour les API REST
- Savoir tester et documenter une API REST
- Apprendre à déployer une API et adopter des bonnes pratiques de développement

Objectifs opérationnels :

- Concevoir, développer et tester des API REST en utilisant Python

Programme

Jour 1 : Introduction et bases des API REST

Matin : Introduction aux API REST et à Python

Présentation des API et des API REST
Qu'est-ce qu'une API ?
Différences entre API REST et autres types d'API
Avantages des API REST
Principes des RESTful services
Méthodes HTTP (GET, POST, PUT, DELETE)
Ressources et endpoints
Statelessness, cacheabilité, uniformité de l'interface
Concepts de CRUD (Create, Read, Update, Delete)
Introduction à Python
Variables, types de données, structures de contrôle
Fonctions et modules
Mise en place de l'environnement de développement
Installation de Python et de pip
Configuration de l'IDE (PyCharm, VSCode, etc.)
Installation des bibliothèques nécessaires (Flask, requests)
Exercice pratique : Installation de Python et création d'un premier script Python simple.

Après-midi : Flask et création d'une API simple

Introduction à Flask
Qu'est-ce que Flask ?
Installation et configuration de Flask
Création d'une application Flask basique
Structure d'un projet Flask
Création du fichier de base (app.py)
Routes et gestion des requêtes HTTP
Définir des routes et des méthodes HTTP associées
Gestion des paramètres d'URL et des requêtes
Retourner des réponses JSON
Utilisation de jsonify pour retourner des données JSON
Exercice pratique : Développer une petite API avec Flask qui gère des

opérations de base (GET, POST).

Jour 2 : Fonctionnalités avancées et sécurisation des API

Matin : Manipulation des données et gestion des erreurs

Utilisation de Flask avec des bases de données (SQLAlchemy)
Introduction à SQLAlchemy
Configuration de la base de données avec Flask-SQLAlchemy
Modèles de données et ORM
CRUD : création, lecture, mise à jour, suppression des ressources
Implémentation des opérations CRUD avec SQLAlchemy
Routes associées aux opérations CRUD
Gestion des erreurs et réponses personnalisées
Gestion des exceptions
Personnalisation des messages d'erreur
Middleware et hooks
Introduction aux middlewares
Utilisation de before_request et after_request hooks
Exercice pratique : Implémenter une API CRUD complète avec gestion des erreurs

Après-midi : Sécurisation des API

Authentification et autorisation (JWT, OAuth2)
Introduction aux JWT (JSON Web Tokens)
Implémentation de JWT avec Flask-JWT-Extended
Concepts de OAuth2
Sécurisation des endpoints
Utilisation des décorateurs pour sécuriser les routes
Gestion des rôles et permissions
Protection contre les attaques courantes
Protection contre les attaques XSS et CSRF
Prévention des injections SQL
Exercice pratique : Ajouter une couche d'authentification JWT à l'API et sécuriser les endpoints

Jour 3 : Tests, documentation et déploiement

Matin : Tests et documentation des API

Introduction aux tests unitaires et d'intégration
Concepts des tests unitaires
Introduction à unittest et pytest
Utilisation de Postman pour tester les API
Installation et configuration de Postman
Création de collections et de tests automatisés
Documentation des API avec Swagger/OpenAPI
Introduction à Swagger et OpenAPI
Utilisation de Flask-RESTPlus pour générer une documentation Swagger
Exercice pratique : Écrire des tests unitaires et d'intégration pour l'API, et générer une documentation Swagger

Après-midi : Déploiement et bonnes pratiques

Déploiement sur Heroku/Clouds/Docker
Introduction à Heroku et les Clouds
Déploiement d'une application Flask sur Heroku
Introduction à Docker et création d'une image Docker pour l'API
Gestion de la configuration et des variables d'environnement
Utilisation de fichiers .env
Configuration de Flask pour utiliser les variables d'environnement
Bonnes pratiques de développement et maintenance des API
Versionnement des API
Gestion des logs et monitoring
Documentation continue et mise à jour
Exercice pratique : Déployer l'API développée sur une plateforme cloud (Heroku ou autre)