Faysal Aziz

FastAPI CRUD API Development

Building Production-Ready Database APIs for MariaDB and Oracle using python - 3 days

FA-0206 · 3 days

Software Development · Data & Analytics

Introduction

This hands-on course focuses on teaching beginners how to build REST API endpoints that perform CRUD (Create, Read, Update, Delete) operations using Python and FastAPI with enterprise databases such as MariaDB and Oracle. CRUD operations form the foundation of almost every data-driven application and represent the essential operations for managing persistent data.

Participants will start from the fundamentals of Python programming and REST API concepts, then gradually progress toward designing and implementing fully functional API endpoints connected to relational databases. The course will also introduce modern best practices such as data validation using Pydantic, database abstraction with SQLAlchemy, project structuring, and API testing through interactive documentation tools.

The instructor for this course brings over **30 years of industry experience**, ensuring that the material reflects **real industry-demanded development practices rather than purely academic concepts**.

Learning outcomes

By the end of this training, participants will be able to:

- Understand how APIs work and how RESTful services expose endpoints
- Install and configure a Python backend development environment
- Build web APIs using the FastAPI framework
- Design and implement CRUD endpoints for database-driven applications
- Connect FastAPI applications to relational databases such as MariaDB and Oracle

- Use SQLAlchemy ORM for database interaction
- Create and validate request/response models using Pydantic
- Structure backend API projects using maintainable architecture
- Test APIs using FastAPI's built-in interactive documentation
- Understand best practices for database sessions, connection pooling, and error handling

---

Prerequisites

Participants should have:

- Basic computer literacy
- Basic understanding of programming concepts (variables, logic, loops)
- Basic understanding of databases or SQL (helpful but not mandatory)

  A laptop with:
  - Python 3.10+ (or ssh capability to access the labs)
  - A code editor: VS Code
  - Access to MariaDB and Oracle database instances (provided as external Labs)
- Basic command-line familiarity

Training outline

1. Foundations of Web APIs and Backend Development

1. Understanding modern backend architectures
2. Role of APIs in enterprise systems
3. Overview of REST architecture
4. HTTP protocol fundamentals
5. HTTP request and response structure
6. REST endpoint concepts

   API verbs and semantics
   1. GET
   2. POST
   3. PUT
   4. PATCH
   5. DELETE

8. JSON as a data interchange format
9. API status codes and error responses

2. Python Foundations for API Development

1. Overview of Python in backend development
2. Installing Python development environment
3. Python package management using pip
4. Virtual environments for dependency isolation

   Python scripting fundamentals

   1. variables and data types
   2. control structures
   3. functions and modules
6. Python package structure
7. Importing and managing modules
8. Introduction to asynchronous programming concepts
9. Understanding async and await in Python

3. Development Environment Setup

1. Installing required development tools
2. Installing FastAPI framework
3. Installing Uvicorn ASGI server

   Installing database drivers

   1. MariaDB connector
   2. Oracle database driver

   Installing ORM libraries

   1. SQLAlchemy
6. Creating a Python project workspace
7. Running the first FastAPI application
8. Understanding FastAPI automatic documentation endpoints
9. Running API servers with Uvicorn

4. FastAPI Fundamentals

1. Overview of the FastAPI framework architecture
2. Understanding ASGI servers

3. Creating a FastAPI application object
4. Creating basic endpoints
5. Handling request parameters
6. Path parameters
7. Query parameters
8. Request body handling
9. Response models and serialization
10. API documentation using OpenAPI and Swagger UI
11. FastAPI request lifecycle

5. Data Modeling with Pydantic

1. Understanding data validation in APIs
2. Introduction to Pydantic models
3. Creating request schemas
4. Creating response schemas
5. Data validation rules
6. Field types and constraints
7. Nested schemas
8. Serialization and deserialization
9. Automatic schema generation in OpenAPI documentation

6. Introduction to Relational Databases

1. Overview of relational database concepts
2. Tables, rows, and columns
3. Primary keys and foreign keys
4. SQL query fundamentals
5. Data manipulation operations
6. Schema design considerations
7. Understanding transactional behavior
8. Differences between MariaDB and Oracle environments

7. Database Integration with Python

1. Overview of ORM concepts
2. Introduction to SQLAlchemy ORM

3. Benefits of ORM vs raw SQL
4. Installing and configuring SQLAlchemy
5. Creating database engine connections
6. Session management
7. Database transaction handling
8. Database connection pooling strategies

8. Connecting FastAPI to MariaDB

1. Configuring MariaDB database connection
2. Installing MariaDB Python connector
3. Creating SQLAlchemy engine for MariaDB
4. Managing database sessions
5. Defining database models
6. Mapping Python classes to database tables
7. Performing database queries through ORM
8. Handling connection pooling and performance considerations

9. Connecting FastAPI to Oracle Database

1. Oracle database driver installation
2. Creating Oracle connection strings
3. Configuring SQLAlchemy with Oracle
4. Handling enterprise database environments
5. Session lifecycle management
6. Transaction handling with Oracle
7. Cross-database compatibility strategies

10. Desigining CRUD APIs

1. Understanding CRUD patterns in API design
2. Mapping CRUD operations to HTTP methods
3. Designing resource-oriented endpoints
4. Endpoint naming conventions
5. Request and response design patterns
6. Validation and error handling strategies

11. Implementing CRUD Endpoints in FastAPI

1. Creating resources with POST endpoints
2. Retrieving records with GET endpoints
3. Fetching collections of resources
4. Fetching individual resources by ID
5. Updating records with PUT and PATCH endpoints
6. Deleting records with DELETE endpoints
7. Database commit and rollback strategies
8. Handling missing records and validation errors

12. Structuring a Scalable FastAPI Project

1. Organizing application folders
2. Separation of concerns in API design
3. Modular routing using APIRouter
4. Organizing models, schemas, and services
5. Configuration management
6. Environment variables and settings management
7. Dependency injection patterns

13. API Testing and Debugging

1. Using Swagger UI for API testing
2. Using ReDoc documentation interface
3. Testing endpoints with HTTP clients
4. Inspecting request and response payloads
5. Logging and debugging API issues
6. Handling validation errors and exceptions

14. Performance and Best Practices

1. Async vs synchronous database access
2. Connection pooling strategies
3. Handling database session lifecycle
4. Preventing common API errors
5. Input validation and secure coding practices

6. API versioning strategies

15. Preparing APIs for Real-World Deployment

1. Running APIs with ASGI servers
2. Uvicorn configuration
3. Basic production deployment considerations
4. API documentation publishing
5. Monitoring and logging considerations
6. Scaling APIs for enterprise workloads

© 2026 Faysal Aziz · Kuala Lumpur