Introduction to RESTful API Workshop 🚀

A Three-Session Journey

Duration: 3 sessions x 2 hours each

Presenter: *Alberto Camarena*

Date: 12/04/25 📅

Workshop Overview 📝

- **Deep REST Fundamentals**
    - Detailed REST theory and its relationship with HTTP
- **State-of-the-Art Comparison**
    - SOAP vs. REST vs. GraphQL
- **Hands-On Experience**
    - Consuming an existing API
    - Building your own backend API
    - Creating a frontend that integrates with your API

Agenda Overview 📆

1. **Session 1:** REST Fundamentals, HTTP Protocol & API Consumption
2. **Session 2:** Building the Backend
3. **Session 3:** Building the Frontend

Session 1: REST Fundamentals, HTTP Protocol & API Consumption

Introduction & Objectives 🎯

- **Deep Dive into REST:**
  - What is REST?
  - How REST works with HTTP (Application Layer)
- **HTTP Protocol Explored:**
  - Overview of HTTP and its components (methods, status codes, headers)
- **State-of-the-Art API Comparison:**
  - SOAP vs. REST vs. GraphQL
- **Practical API Consumption:**
  - Hands-on demo using a public API

What is REST? 🤔

- **Definition:**
  REST (Representational State Transfer) is an architectural style that uses standard HTTP methods.
- **Constraints:**
  - Statelessness
  - Cacheability
  - Layered system
  - Uniform interface
- **Key Focus:**
  - Resources & URIs
  - CRUD operations via HTTP methods

REST and HTTP Protocol 🌐

- **REST at the Application Layer:**
    - Utilizes HTTP methods: GET, POST, PUT, DELETE
- **Key HTTP Elements:**
    - **Methods:**
        - GET: Retrieve data
        - POST: Create resource
        - PUT: Update resource
        - DELETE: Remove resource
    - **Status Codes:**
        - 200 OK, 201 Created, 404 Not Found, 500 Server Error
    - **Headers & Payloads:**
        - Describe metadata and data format (typically JSON)

State-of-the-Art API Architectures ⚖️

- **SOAP:**
    - Heavyweight, XML-based
    - Strict contract with WSDL
- **REST:**
    - Lightweight, uses standard HTTP
    - Flexible and easier to implement
- **GraphQL:**
    - Query language for your API
    - Solves over-fetching/under-fetching challenges
- **Discussion:**
    - Trade-offs, use cases, and evolution of API design

Hands-On Demo: API Consumption 💻

- **Objective:**
  - Consume a public API (e.g., JSONPlaceholder) using code.
- **Example Code (Python):**

```python
import requests

url = "https://jsonplaceholder.typicode.com/posts/1"
response = requests.get(url)
if response.status_code == 200:
    data = response.json()
    print("Title:", data.get("title"))
else:
    print("Error:", response.status_code)
```

- **Example Code (JavaScript):**

```javascript
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then(response => {
    if (response.ok) {
      return response.json();
    } else {
      throw new Error(response.status);
    }
  })
  .then(data => {
    console.log("Title:", data.title);
  })
  .catch(error => {
    console.error("Error:", error.message);
  });
```

- **Live Interaction:**
  - Experiment with GET requests
  - Modify parameters, discuss error handling

Session 1 Wrap-Up & Q&A ❓

- Recap REST fundamentals and HTTP protocol
- Compare SOAP, REST, and GraphQL
- Q&A: Answer your questions and troubleshoot common issues