# REST API Notes

## Understanding REST APIs

A guide to REST API concepts, components, methods, and Python frameworks.

## Importance of REST APIs

REST APIs (Representational State Transfer APIs) are crucial for modern web and mobile applications. They enable communication between different systems and services.

Key benefits include:

* • Platform Independence
* • Scalability
* • Stateless Communication
* • Ease of Integration

## Stateful vs Stateless APIs

REST APIs are typically stateless, meaning each request is independent, and the server does not store session data.

Comparison:

* • Stateful APIs maintain session data on the server.
* • Stateless APIs do not retain session information between requests, improving scalability.

## Components of a REST API Solution

* • Client & Server
* • Resources & Endpoints
* • HTTP Methods & Request/Response Format
* • Status Codes

## Types of REST API Methods

* • GET: Retrieve data
* • POST: Create a resource
* • PUT: Update a resource
* • DELETE: Remove a resource

## API Workflow (Request/Response)

The typical workflow of a REST API request:

* • Client sends an HTTP request.
* • Server processes the request.
* • Server sends an HTTP response.

Example Request:

GET /users/1

Response: { 'id': 1, 'name': 'Alice' }

## HTTP Status Codes (REST API)

HTTP Status Codes indicate the outcome of an API request. They help clients and developers understand request status and errors.

## Python Web Frameworks for API Development

* • Flask: Simple & Lightweight
* • Django: Scalable & Feature-rich
* • FastAPI: High-performance & Async Support

## Flask Web Framework Guide

Flask is a lightweight Python web framework built on Werkzeug (Engine) and Jinja2 (Templating). It is ideal for building REST APIs.

## JWT (JSON Web Token)

JWT is a compact, URL-safe token format used for authentication and authorization in REST APIs.

* • Stateless: No session storage on the server.
* • Secure: Signed tokens prevent tampering.
* • Compact: Easily transmitted in HTTP headers.
* • Scalable: Works well for microservices and cloud-based applications.

## Developing REST API with Flask

A simple REST API using Flask:

from flask import Flask, jsonify  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/api/data', methods=['GET'])  
def get\_data():  
 return jsonify({'message': 'This is a REST API response'})  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **HTTP Status Code Reference** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Code** | **Meaning** | **Description** | **Probable Causes** | **Solution** |
| 200 | OK | The request was successful, and the response contains the expected data. | None (successful response). | No action needed. |
| 201 | Created | A new resource was successfully created. | No issues; confirmation of creation. | Ensure the response includes the created resource's location if applicable. |
| 400 | Bad Request | The server could not understand the request due to invalid syntax. | Missing required parameters, invalid JSON format, or incorrect data types. | Validate input before sending the request and check API documentation for required fields. |
| 401 | Unauthorized | Authentication failed or was not provided. | Missing or incorrect API token, expired credentials. | Ensure a valid token is included in the Authorization header. Refresh expired tokens. |
| 403 | Forbidden | The user is authenticated but does not have permission to access the resource. | Insufficient privileges, role restrictions, or API rate limiting. | Check user permissions and API access policies. Contact the API provider if needed. |
| 404 | Not Found | The requested resource does not exist on the server. | Incorrect URL, resource ID does not exist, or the API endpoint has changed. | Verify the resource ID and endpoint URL. Ensure the resource exists. |
| 405 | Method Not Allowed | The requested HTTP method is not supported for the resource. | Using POST when only GET is allowed, or vice versa. | Check the API documentation for allowed methods on the endpoint. |
| 409 | Conflict | The request conflicts with the current state of the resource. | Attempting to create a duplicate entry, or concurrent updates to the same resource. | Handle concurrency properly, check for duplicates before inserting data. |
| 500 | Internal Server Error | The server encountered an unexpected error. | Server-side bug, unhandled exception, database issues. | Check server logs for errors, ensure proper error handling is in place. |
| 502 | Bad Gateway | The server received an invalid response from an upstream server. | API gateway misconfiguration, server downtime. | Retry after some time. If persistent, check backend connectivity. |
| 503 | Service Unavailable | The server is temporarily unable to handle the request. | High traffic, maintenance, or overloaded server. | Retry later or check service status for maintenance updates. |