**✅ Core REST API Concepts**

Here are 1-line answers for your REST API interview prep:

1. **What is a REST API?**  
   A REST API allows communication between client and server using standard HTTP methods following REST architecture principles.
2. **What are the main principles of REST?**  
   REST principles include statelessness, client-server separation, cacheability, layered system, uniform interface, and resource-based operations.
3. **What are HTTP methods and how are they used in REST APIs?**  
   HTTP methods like GET, POST, PUT, PATCH, DELETE perform read, create, update, partial update, and delete operations on resources.
4. **Difference between PUT and PATCH?**  
   PUT replaces the entire resource, while PATCH updates only specified fields.
5. **What are status codes? Give examples of commonly used ones.**  
   Status codes indicate response results like 200 OK, 201 Created, 400 Bad Request, 404 Not Found, 500 Internal Server Error.
6. **What is idempotency? Which methods are idempotent?**  
   Idempotency means repeated requests produce the same result; GET, PUT, DELETE are idempotent.
7. **What is the difference between API and web services?**  
   An API is a broad interface for software interaction, while web services are APIs that work over a network using protocols like HTTP.
8. **What is the difference between REST and SOAP?**  
   REST is lightweight and uses HTTP, while SOAP is a protocol with strict rules and XML messaging.
9. **What are URI and URL in context of REST?**  
   URI identifies a resource; URL is a type of URI that also specifies how to locate it on the web.
10. **What is the role of headers in API requests?**  
    Headers provide metadata like content type, authorization, and caching policies in API requests/responses.

Notes:

* + Server – A server is a system that provides services or resources to clients over a network.
  + **Statelessness -** Each request from client to server must contain all the information needed, and the server does not store session state.
  + **Client-Server Separation -** The client and server operate independently—client handles UI, server handles data and logic.
  + **Cacheability -** Responses must define whether they are cacheable or not to improve performance.
  + **Layered System -** A REST API can have multiple layers (e.g., load balancers, proxies) and clients should not know or care.
  + **Uniform Interface -** The API should have a consistent way of accessing resources using standard methods (like GET, POST, etc.).
  + **Resource-Based -** Everything in REST is treated as a resource, identified by a URL, and manipulated using HTTP methods.
  + Layers - **In Django, the layered system includes views (controller layer), serializers (data validation layer), models (ORM/data layer), and middleware (cross-cutting concerns like auth/logging).**

**🖥️ Common Types of Servers:**

| **Type** | **What It Does** | **Example** |
| --- | --- | --- |
| **Web Server** | Serves web pages | Apache, Nginx |
| **Database Server** | Stores and manages data | MySQL, PostgreSQL, MongoDB |
| **Application Server** | Runs application logic | Django app server, Node.js |
| **File Server** | Stores and shares files | Windows File Server |
| **Mail Server** | Sends and receives emails | Microsoft Exchange, Postfix |

🔁 Layers in REST Architecture (with 1-line explanation each):

1. **Client Layer**

Sends requests and displays responses (e.g., browser, mobile app).

1. **Presentation Layer (Frontend)**

Handles UI/UX — displays data fetched from API.

1. **API Gateway / Reverse Proxy**

Acts as an entry point; routes requests to the correct backend service (e.g., NGINX, AWS API Gateway).

1. **Authentication Layer**

Validates user identity using tokens (e.g., JWT, OAuth).

1. **Rate Limiting / Throttling Layer**

Controls how many requests a client can make per time (prevents abuse).

1. **Business Logic Layer (Service Layer)**

Contains core processing rules and application logic.

1. **Data Access Layer (Model/ORM)**

Handles communication with the database (e.g., Django ORM).

1. **Database Layer**

Stores and manages persistent data (e.g., PostgreSQL, MongoDB).

1. **Cache Layer**

Stores frequent data temporarily to speed up response (e.g., Redis, Memcached).

1. **Logging/Monitoring Layer**

Tracks API usage, errors, and performance metrics (e.g., Sentry, Prometheus).

**🔢 Explain by Series (for Interview)**

| **Series** | **Meaning** | **Example Codes** | **Interview Explanation** |
| --- | --- | --- | --- |
| **200 Series** | ✅ **Success** | 200, 201, 204 | The request was successful. |
| **400 Series** | ❌ **Client Error** | 400, 401, 403, 404 | The client sent a wrong request. |
| **500 Series** | ⚠️ **Server Error** | 500, 502, 503 | The server failed to complete a valid request. |

**💡 Detailed with Examples (Optional in Interview):**

**🔹 200 Series – Success**

* 200 OK – Request succeeded.
* 201 Created – Resource created (e.g., after POST).
* 204 No Content – Success, but no content to return.

**🔹 400 Series – Client Errors**

* 400 Bad Request – Invalid syntax in request.
* 401 Unauthorized – No or wrong authentication.
* 403 Forbidden – Authenticated but not allowed.
* 404 Not Found – Resource doesn’t exist.

**🔹 500 Series – Server Errors**

* 500 Internal Server Error – Generic server error.
* 502 Bad Gateway – Invalid response from another server.
* 503 Service Unavailable – Server is down or overloaded.

Notes:

* Broad interface - Broad interface in REST means using a consistent and uniform method set (GET, POST, PUT, DELETE) to access and manage all resources.