API - Application Programming Interface

Node-Express-API

Vraj Suratwala

Department of ICT, VNSGU, Surat

Topics

- What is API ?
- REST API principles and usage
- API Security
- HTTP Methods in REST APIs
- Status Codes and Error Handling
- RESTFUL vs SOAP
- How to Create a REST API in NodeJS? - with full CRUD Example. [API Testing using POSTMAN]
- Advantages and Disadvantages

What is an API ?

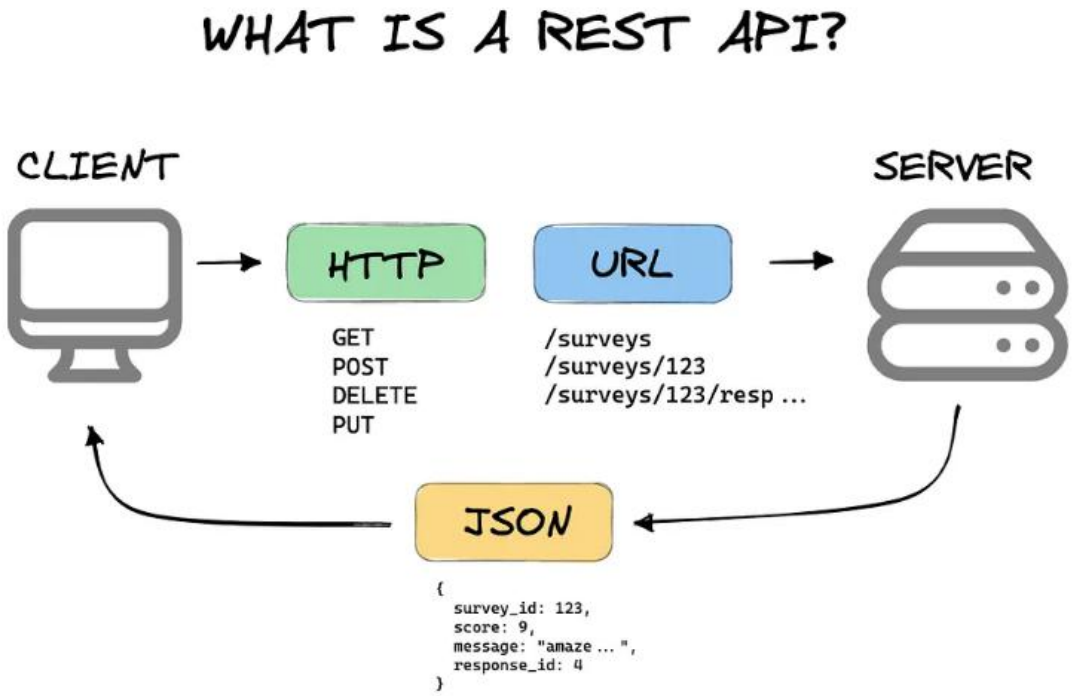
- API refers to Application Programming Language which architectural style for designing networked applications that has become the standard for web services.
- Which transfers the JSON Data to Communicate with different communications!

What is REST API?

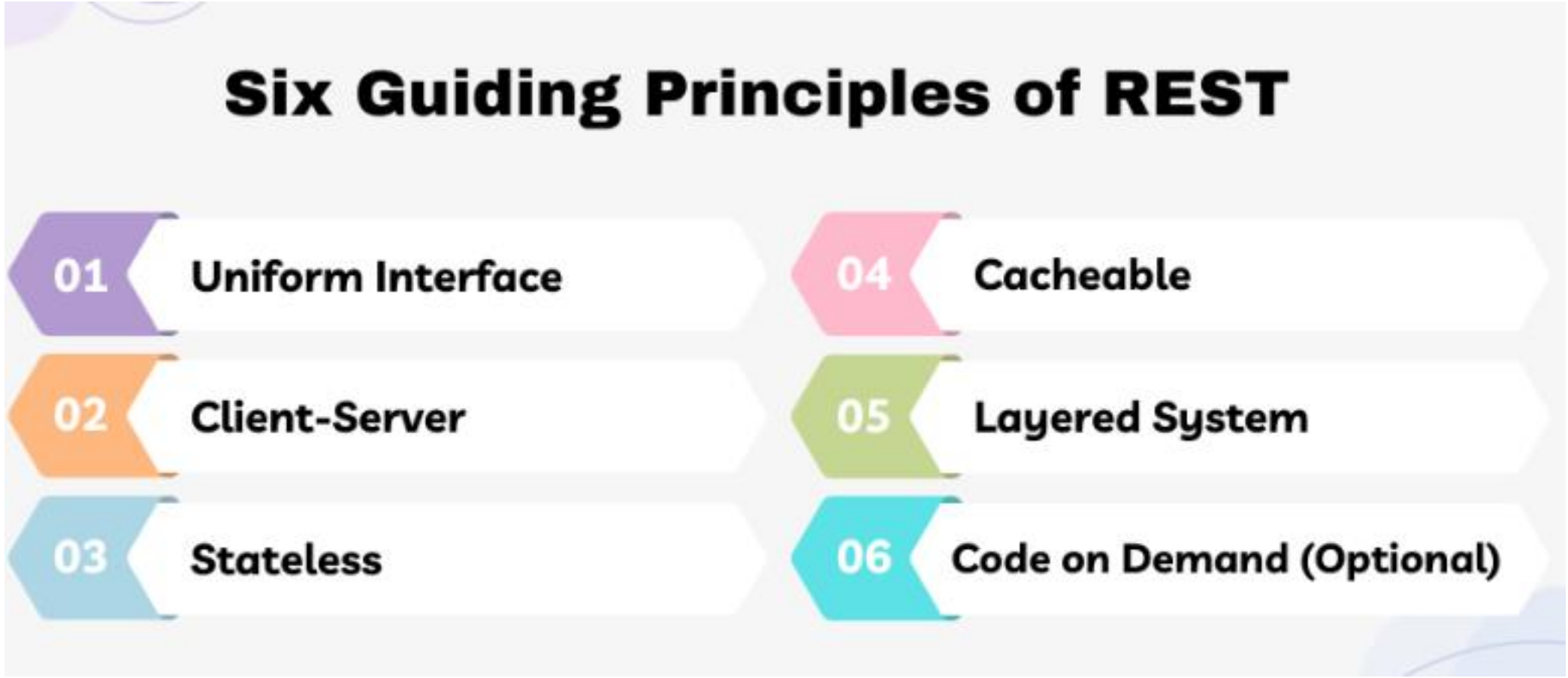
- REST stands for Representational State Transfer and API stands for Application Program Interface.
- REST is a software architectural style that defines the set of rules to be used for creating web services.
- Web services that follow the REST architectural style are known as RESTful web services.
- requesting systems to access and manipulate web resources by using a uniform and predefined set of rules.
- Interaction in REST-based systems happens through the Internet's Hypertext Transfer Protocol (HTTP).

RESTFUL System consists of a :

- A client who requests for the resources.
- server who has the resources.



The Six Principels of REST



1. Uniform Interface

- This is perhaps the most important principle of REST. The uniform interface simplifies and decouples the architecture, allowing each part to evolve independently.
- It consists of four constraints:
  - Identification of Resources : The interface must uniquely identify each resource involved in the interaction between the client and the server.
  - Manipulation of resources through representations
  - Self-descriptive messages
  - Hypermedia as the engine of application state – The client should have only the initial URI of the application. The client application should dynamically drive all other resources and interactions with the use of hyperlinks.

2. Client - Server

- The client-server design pattern enforces the separation of concerns, which helps the client and the server components evolve independently.
- By separating the user interface concerns (client) from the data storage concerns (server), we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

3. Stateless

- Refers to each request from the client to the server must contain all of the information necessary to understand and complete the request.
- The server shouldn’t store any context about the client between requests.
- This means :
  - No session state is stored on the server
  - Each request is independent
  - Authentication information is included with each request
  - This improves scalability and reliability

4. Cacheability

- Clients should be able to cache responses when appropriate. The server must indicate whether a response can be cached and for how long.
- Benefits include:
  - Improved performance
  - Reduced server load
  - Better scalability
  - Network efficiency

5. Layered System

- The architecture can be composed of hierarchical layers, where each layer provides services to the layer above and consumes services from the layer below. Clients cannot ordinarily tell whether they’re connected directly to the end server or an intermediary.
- This enables:
  - Load balancers
  - Caches
  - Security layers
  - Gateways

6. Code on Demand

- It is an optional feature.
- According to this, servers can also provide executable code to the client.
- The examples of code on demand may include the compiled components such as Java Servlets and Server-Side Scripts such as JavaScript.

HTTP methods used in API

- 1. GET - Retrieve Information.

```
GET /users                # Get a list of users
GET /users/123            # Get a specific user
GET /users/123/posts      # Get posts by a specific user
```

- 2. POST - Create Resources.

```
POST /users
Content-Type: application/json
{
  "name": "John Doe",
  "email": "john@example.com"
}
```

HTTP methods used in API

- 3. PUT - Create or Replace.

```
PUT /users/123
Content-Type: application/json
{
  "id": 123,
  "name": "John Doe",
  "email": "john.doe@example.com"
}
```

- 4. PATCH -Partial Update

```
PATCH /users/123
Content-Type: application/json
{
  "email": "john.doe@example.com"
}
```

HTTP methods used in API

- 5. DELETE - Remove Resources.

```
DELETE /users/123
```

- 6. HEAD - Similar to GET but retrieves only headers, not the body
- 7. OPTIONS - Get information about the communication options available.

Important Status Code

reference : https://status-code.medium.com/rest-api-principles-3859e7699fd4

- Success Code : (200)
  - 200 - OK (GET,PETCH)
  - 201 - Created (POST)
  - 204 - No Content (DELETE)

- Client Errors Codes : (400)
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found

- Server Error Code : (500)
  - 500 Internal server code
  - 502 Bad Gatway
  - 503 Service Unavailable

API Security

- API security refers to the protection of Application Programming Interfaces (APIs) from malicious attacks, misuse, and unauthorized access.
- As APIs serve as bridges between systems, especially in web and mobile apps, they become prime targets for hackers.

Key points of API Security

- 1. Authentication & Authorization
- 2. Rate Limiting
- 3. Input Validation
- 4. HTTPS Enforcement
- 5. Access Control
- 6. Logging & Monitoring
- 7. Security Testing

SOAP vs RESTful

Feature	SOAP	RESTful
Protocol or Style	Protocol	Architectural Style
Format	XML only	JSON, XML, HTML, etc.
Speed	Slower (heavy)	Faster (lightweight)
Flexibility	Less	High
Security	WS-Security	HTTPS, OAuth, JWT
Error Handling	SOAP Faults	HTTP Status Codes
Use Case	Enterprise apps, banking	Web/mobile apps, IoT

How to Create and Use of API with Mongoose ?

Full Example

How to Create API With Node.js

- **db.js - config file.**
- const mongoose = require('mongoose');
- mongoose.connect("mongodb://0.0.0.0:27017/db1");
- const db = mongoose.connection;
- //mongoose.set('useFindAndModify', false);
- db.on('error', console.error.bind(console, 'connection error:'));

- module.exports = mongoose;

books.js - model

```
//SCHEMA
const mongoose = require('../config/db.js');


const BookSchema = mongoose.Schema({
    name: String,
    price: Number,
    quantity: Number
 });
const Book = mongoose.model('Book', BookSchema, "books"); //MODEL


module.exports  = Book;
```

BookRoutes.js - Router file

```
const app = require('express').Router();
const Book = require('../models/book.js');

app.get("/",(req,res)=>{
    //Get all
        Book.find({})
            .then(( books)=> {
                res.send(books);
            })
            .catch((err)=>{
                res.status(400).send("ERR")
            })
});
```

server.js - mail file

```
const express = require('express');
const app = express();
app.use(express.urlencoded({ extended: true }));
const Book   = require('./models/book.js');
const mongoose = require('./config/db.js');
const router = require('./routes/BookRoutes.js');

app.use('/books',router);
if(mongoose)
{
    console.log("Database Connected");
}
else{
    console.log("Database is not Connected!");
}

const PORT = process.env.PORT ||8000; // Application can Ported by this kind of approach.
app.listen(PORT,(err)=>{
    console.log(`App is running on port no ${PORT}.`)
});
```

API TESTING ON POSTMAN

GET http://localhost:8000/books/

	_id	name	price	quantity
0	688c4ed8501c8736333ca00b	Vraj	500	18

Advantages of API

- Easy Integration
- Automation and Efficiency
- Faster Development
- Scalability
- Customization
- Exhance UI

Disadvantages of an API

- Security Risks - Solved by JWT
- Dependency on Third-Party Services - Relay only on Authenticated Source
- Versioning Issues
- Cost
- Complexity in Management

References

- 1. https://restfulapi.net/
- 2. https://www.restapi.guru/rest-api-fundamentals
- 3. https://status-code.medium.com/rest-api-principles-3859e7699fd4
- 4. https://www.geeksforgeeks.org/node-js/what-is-rest-api-in-node-js/
- 5. https://www.postman.com/api-platform/api-testing/