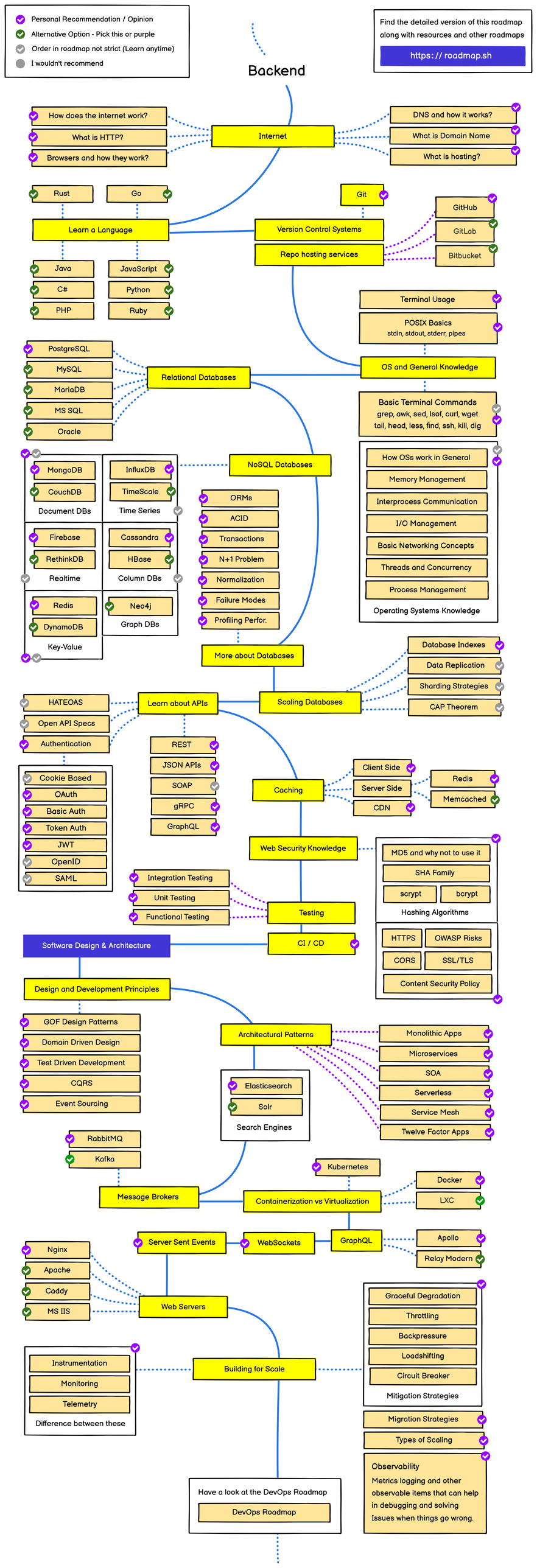

Personal Recommendation / Opinion
Alternative Option - Pick this or purple
Order in roadmap not strict (Learn anytime)
I wouldn't recommend
Find the detailed version of this roadmap along with resources and other roadmaps
https://roadmap.sh
Backend
Internet
How does the internet work?
What is HTTP?
Browsers and how they work?
DNS and how it works?
What is Domain Name
What is hosting?
Learn a Language
Rust
Go
Java
JavaScript
C#
Python
PHP
Ruby
Version Control Systems
Git
Repo hosting services
GitHub
GitLab
Bitbucket
Terminal Usage
POSIX Basics
stdin, stdout, stderr, pipes
OS and General Knowledge
Basic Terminal Commands
grep, awk, sed, lsof, curl, wget
tail, head, less, find, ssh, kill, dig
How OSs work in General
Memory Management
Interprocess Communication
I/O Management
Basic Networking Concepts
Threads and Concurrency
Process Management
Operating Systems Knowledge
Relational Databases
PostgreSQL
MySQL
MariaDB
MS SQL
Oracle
NoSQL Databases
MongoDB
CouchDB
Document DBs
InfluxDB
TimeScale
Time Series
Firebase
RethinkDB
Realtime
Cassandra
HBase
Column DBs
Redis
DynamoDB
Key-Value
Neo4j
Graph DBs
ORMs
ACID
Transactions
N+1 Problem
Normalization
Failure Modes
Profiling Perfor.
More about Databases
Scaling Databases
Database Indexes
Data Replication
Sharding Strategies
CAP Theorem
Learn about APIs
HATEOAS
Open API Specs
Authentication
Cookie Based
OAuth
Basic Auth
Token Auth
JWT
OpenID
SAML
REST
JSON APIs
SOAP
gRPC
GraphQL
Caching
Client Side
Server Side
CDN
Redis
Memcached
Web Security Knowledge
MD5 and why not to use it
SHA Family
scrypt
bcrypt
Hashing Algorithms
HTTPS
OWASP Risks
CORS
SSL/TLS
Content Security Policy
Integration Testing
Unit Testing
Functional Testing
Testing
CI / CD
Software Design & Architecture
Design and Development Principles
GOF Design Patterns
Domain Driven Design
Test Driven Development
CQRS
Event Sourcing
Architectural Patterns
Monolithic Apps
Microservices
SOA
Serverless
Service Mesh
Twelve Factor Apps
Elasticsearch
Solr
Search Engines
RabbitMQ
Kafka
Message Brokers
Kubernetes
Containerization vs Virtualization
Docker
LXC
Server Sent Events
WebSockets
GraphQL
Apollo
Relay Modern
Nginx
Apache
Caddy
MS IIS
Web Servers
Graceful Degradation
Throttling
Backpressure
Loadshifting
Circuit Breaker
Mitigation Strategies
Instrumentation
Monitoring
Telemetry
Difference between these
Building for Scale
Migration Strategies
Types of Scaling
Observability
Metrics logging and other observable items that can help in debugging and solving Issues when things go wrong.
Have a look at the DevOps Roadmap
DevOps Roadmap

Rest API Fundamentals

All that you need to know!

swipe >

API Concept

API allows interactions between systems by following a set of standards and protocols in order to share features, information and data.

Full form of API

A - Application

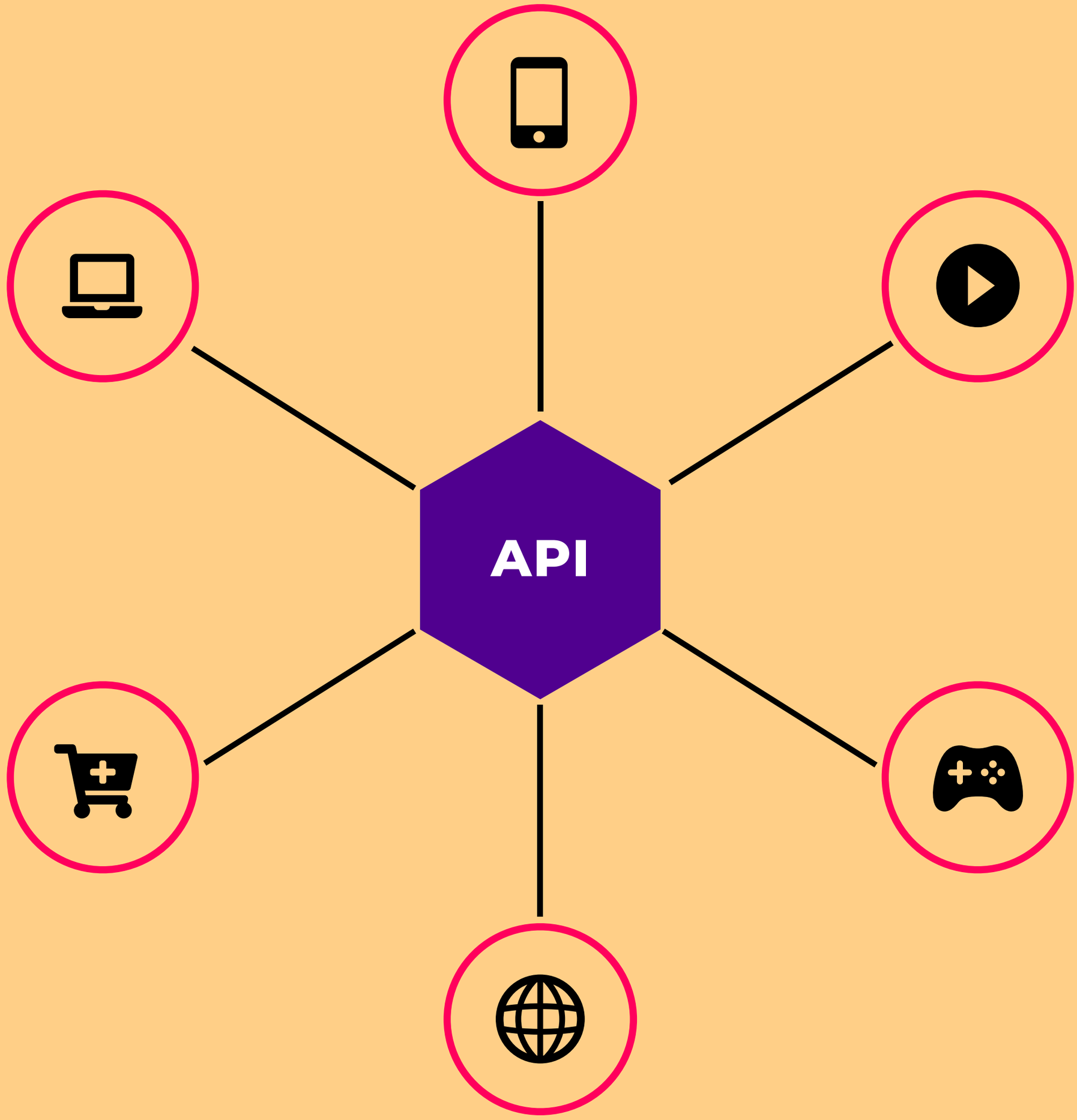
P - Programming

I - Interface

swipe >

API acts as an interface between different applications.

Diagram



REST API

REST

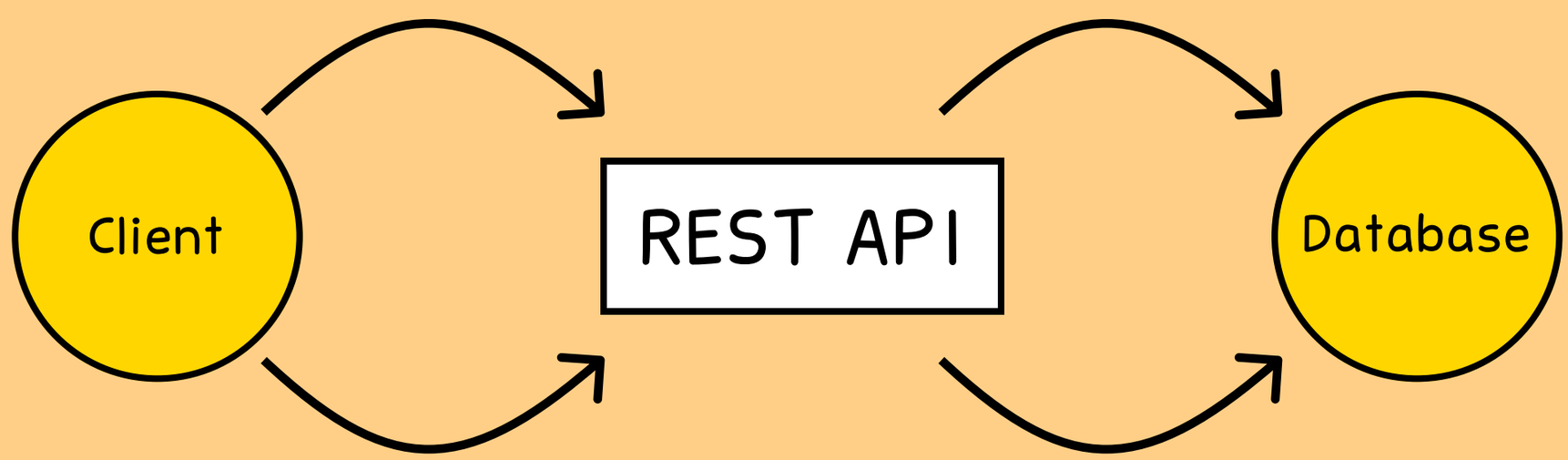
Representational State Transfer

It is an architecture style to develop web applications.

Uses HTTP protocol as a communication interface.

It transfers data through HTTP methods.

Diagram



swipe >

Request Anatomy

URL

Uniform Resource Locator

It is the address to not just identify a resource, but also to specify how to access it.

In an API, the URL can be named as Base URL, which means that is the base address that will be used in every request.

Example : http://api.example.com

swipe >

URI

Uniform Resource Identifier

It is used in the URL to specify which resource the client would like to access in a request.

Example : http://api.example.com/products?name=laptop&available=true

URI
(highlighted in purple)

Here, the client communicates to the server that the request is to retrieve products with name equals laptop, and available equals true.

Every URL is an URI, but not all URIs are URLs.

swipe >

Body Params

Body of the request which contains all the data that the server needs to successfully process the request.

Only used in requests that must send information, such as create or update.

Example

```
{
"name" : "Laptop"
"price" : 1000
"available" : true
}
```

Parameters

Information that can be sent in a request by the client in order to influence the response by the server.

Types

Path Parameter

A variable in the URI path that helps in pointing towards specific resource.

https://food.com/restaurant/1/total-orders

1 is the path parameter (variable) which points to restaurant no 1's total orders.

Query Parameter

A variable in URI path that helps in querying / filtering through a list of resources.

https://food.com/products?name=cake

name=cake is the query parameter (variable) which retrieves products with the name "cake".

swipe >

Headers

It allows sending extra information in a request, such as authentication tokens and content types.

Example

```
Authorization : Bearer token
Accept : application/json
```

In this given example, the client is sending extra data informing not just it's credentials to access a resource, but also a desired response format.

REST API Methods

5 Methods

1 GET

Receive information about an API resource.

2 POST

Create an API resource

3 PUT

Update an API resource

4 DELETE

Delete an API resource

5 PATCH

Modifies an API resource

swipe >

Http Status Codes

With every request made to the server, we get response code /http status codes in return.

Status Codes

2XX : Success

3XX : Redirection

4XX : Problem with the client

5XX : Problem with the server

swipe >

brijpandeyji

API Simplified

The Ultimate 101 Guide

Your Go-To Quick Reference Guide for Effortless API Mastery

swipe >

HTTP Verbs

13 Verbs

1 GET

Retrieve data from the server.

2 POST

Send data to the server to create a resource

3 PUT

Send data to the server to update a resource.

4 PATCH

Send data to the server to update a resource partially.

5 DELETE

Delete a resource from the server.

6 PURGE

Invalidates a cached resource.

swipe >

7 TRACE

Returns the full HTTP request received by the server for debugging and diagnostic purposes.

8 OPTIONS

Returns the HTTP methods supported by the server for the requested URL

9 CONNECT

Converts the request connection to a transparent TCP/IP tunnel for secure communication.

10 LOCK

Locks the resource for exclusive use by the client.

11 UNLOCK

Unlocks the resource previously locked by the client.

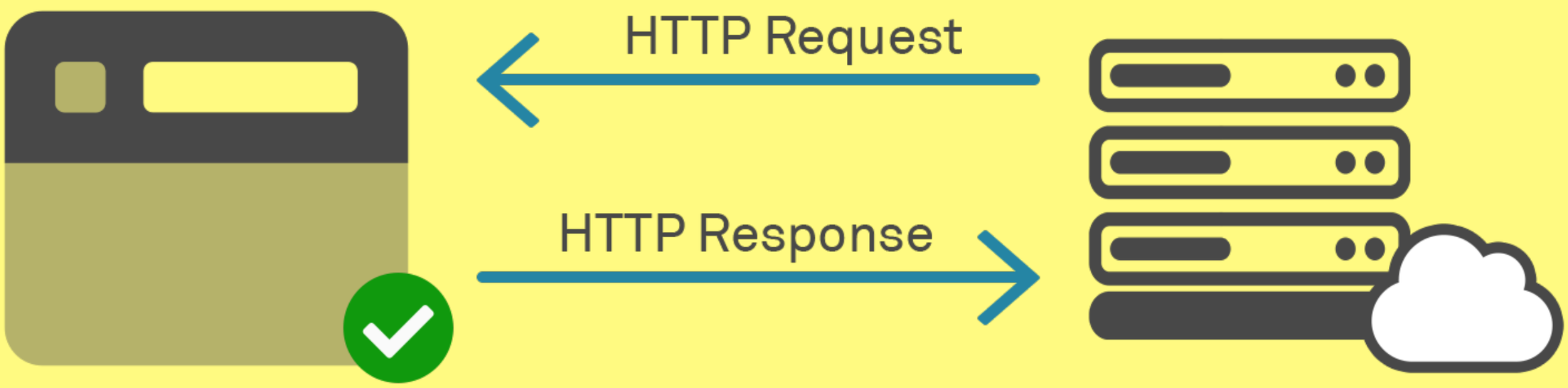
swipe >

12 MKCOL

Creates a new collection resource.

13 COPY

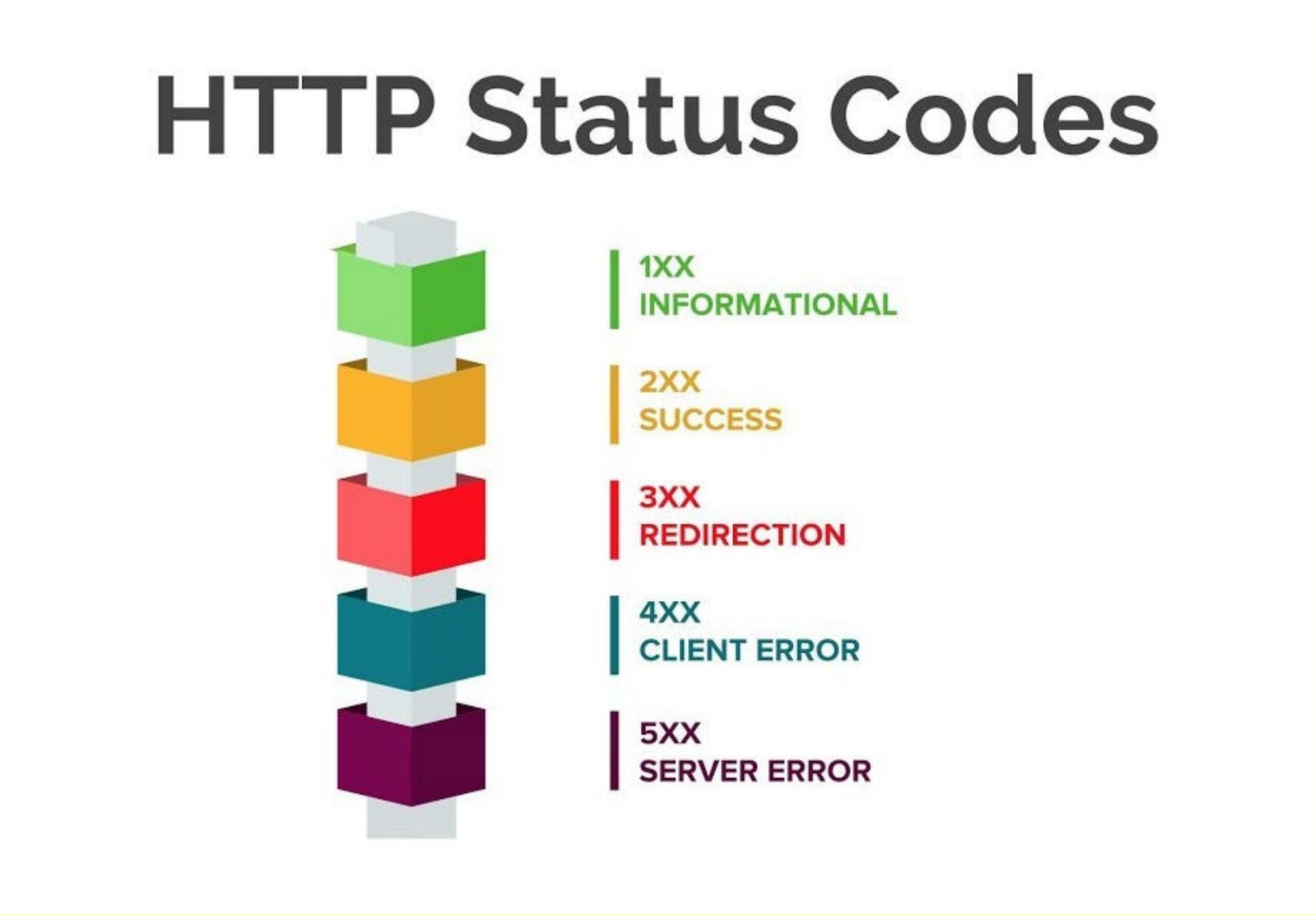
Copies the resource identified by the Request-URI to the destination URI.



HTTP Status Codes

5 Status Codes

1. 1xx
   Informational.
2. 2xx
   Success.
3. 3xx
   Redirection.
4. 4xx
   Client Errors.
5. 5xx
   Server Errors.



swipe >

Response Headers

1 Content-Type

Specifies the MIME type of the data in the response body.

2 Content-Length

Specifies the length of the response body in bytes.

3 Cache-Control

Specifies the caching behavior of the response.

4 Location

Specifies the URI of a resource that can be used to retrieve the requested resource.

swipe >

5 Server

Specifies the name and version of the server software that generated the response.

6 Access-Control-Allow-Origin

Specifies which origins are allowed to access the resource.

7 Set-Cookie

Specifies a cookie that should be stored by the client and sent back to the server with future requests.

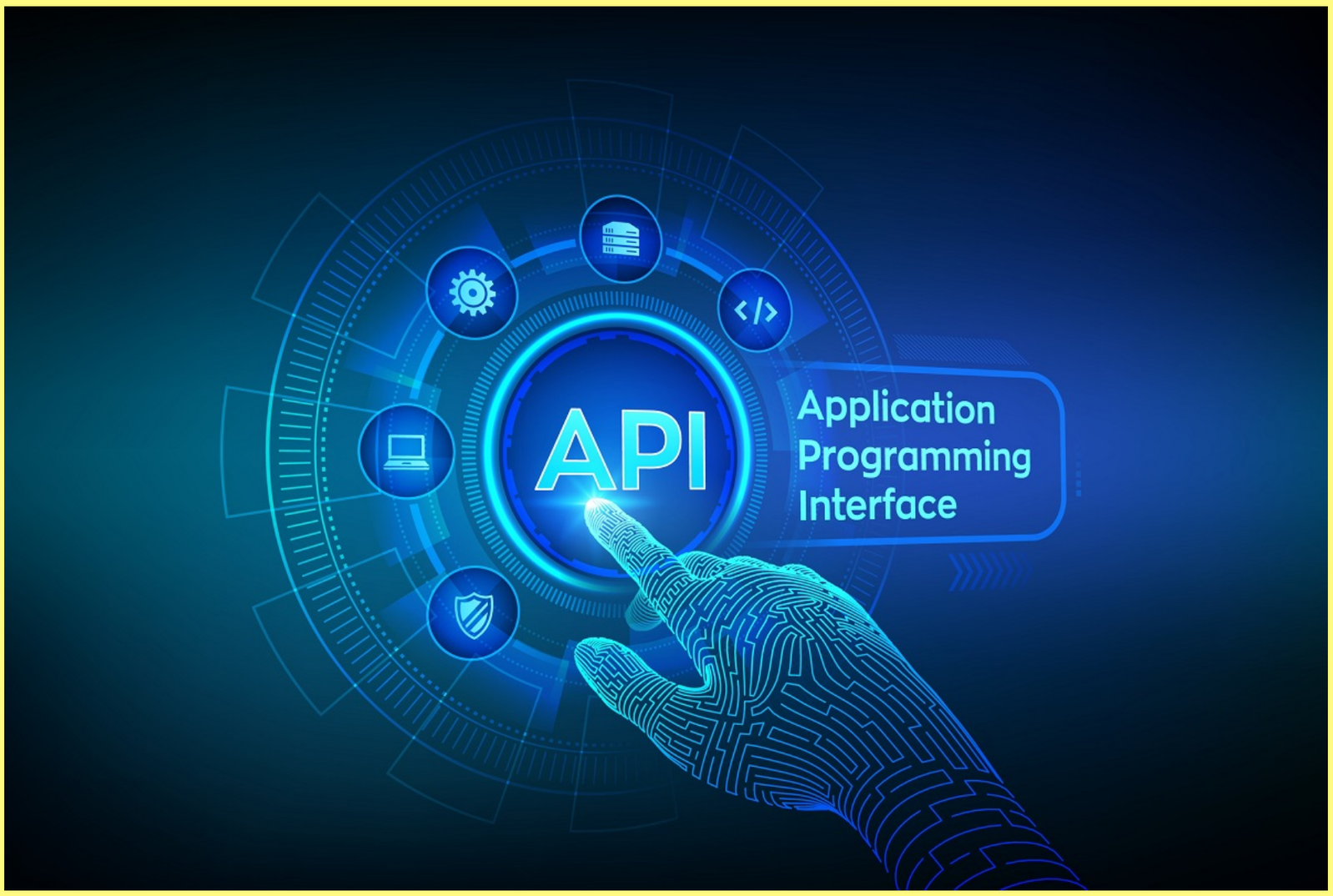
swipe >

8 Expires

Specifies the date and time after which the response is considered stale.

9 Last-Modified

Specifies the date and time the resource was last modified.



API Design

1 REST

Representational State Transfer, a design pattern for building web services.

2 SOAP

Simple Object Access Protocol, a messaging protocol for exchanging structured data.

3 GraphQL

A query language and runtime for building APIs.

4 API Gateway

A service that manages, protects, and scales APIs.

swipe >

API Architectures

1 Microservices

An architectural style for building complex applications as a suite of small, independent services.

2 Serverless

A cloud computing execution model where the cloud provider manages the infrastructure and automatically allocates resources as needed.

3 SOA

Service-Oriented Architecture, an architectural style for building distributed systems.

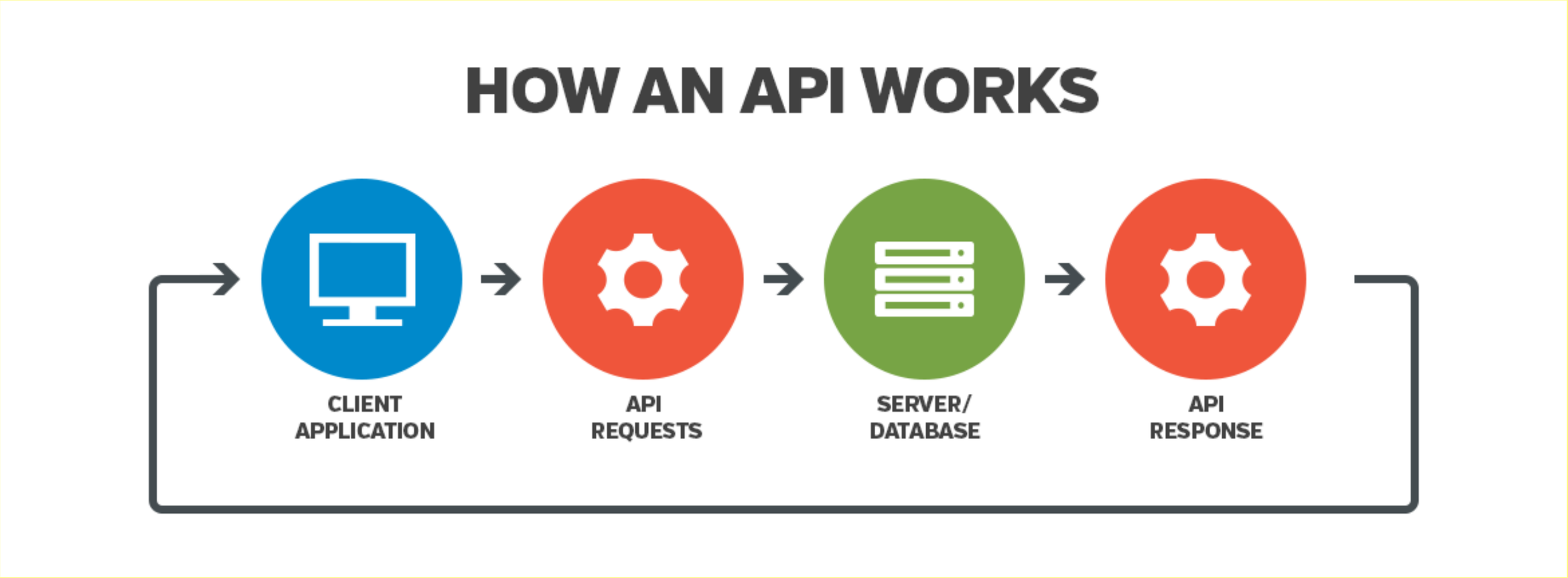
swipe >

4 Event-Driven

An architectural style where the flow of data between components is triggered by events.

5 RESTful API

An architectural style that uses HTTP requests to GET, POST, PUT, and DELETE data.



API Design Patterns

1 Adapter Pattern

A pattern that converts the interface of a class into another interface that clients expect.

2 Decorator Pattern

A pattern that adds behaviour to an individual object dynamically.

3 Proxy Pattern

A pattern that provides a surrogate or placeholder for another object to control access to it.

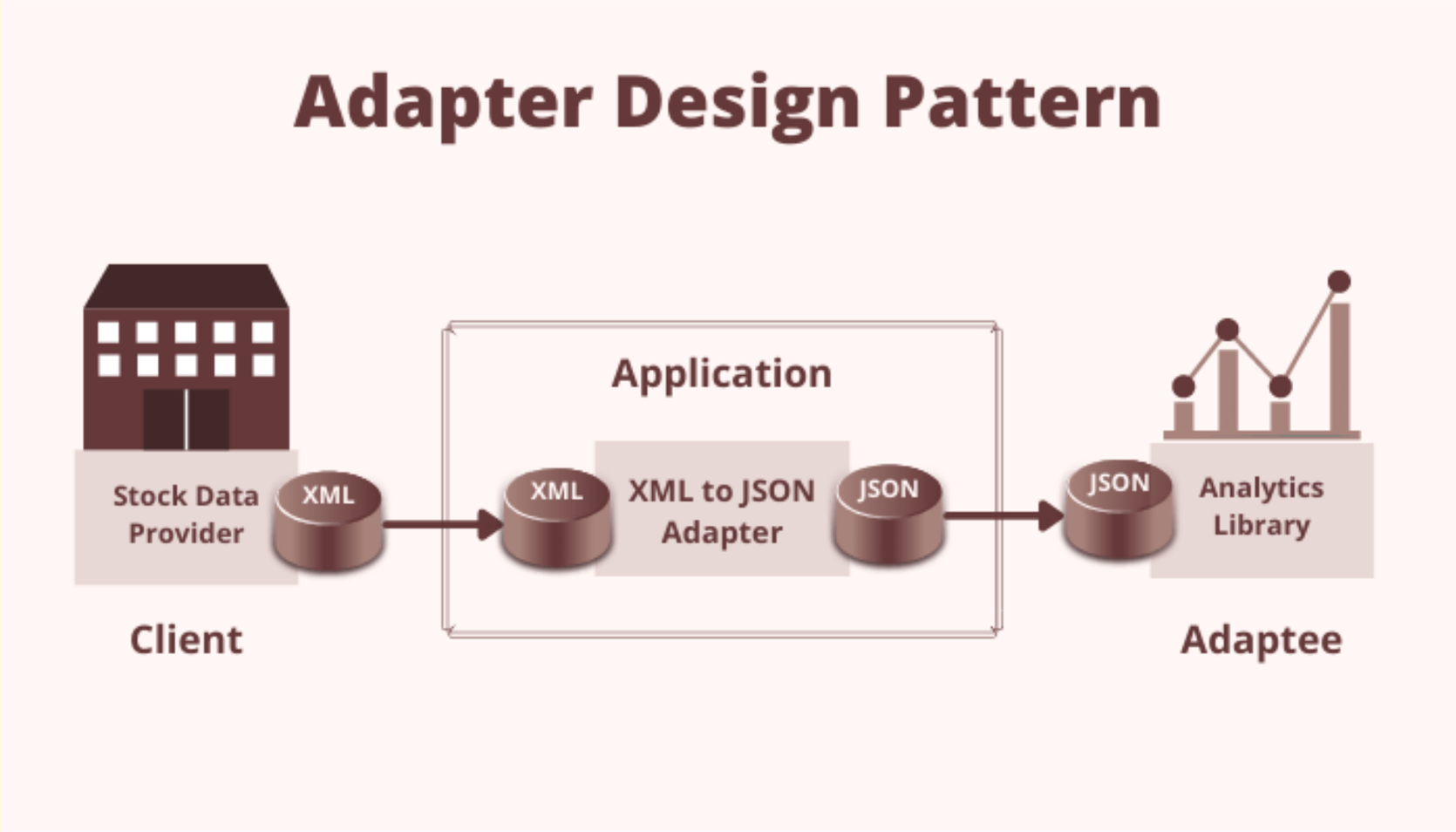
swipe >

4 Chain of Responsibility Pattern

A pattern that delegates commands to a chain of processing objects.

5 Observer Pattern

A pattern that defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.



swipe >

API Security

1 OAuth

An open standard for authorization used for protecting APIs.

2 JWT

JSON Web Tokens, a standard for securely transmitting information between parties as a JSON object.

3 SSL/TLS

Secure Sockets Layer/ Transport Layer Security, a protocol for establishing a secure connection between a client and a server.

swipe >

4 API Key

A secret token used to authenticate API requests.

5 Rate Limiting

A technique used to limit the number of requests that can be made to an API over a specific period of time.

6 OpenID Connect

An authentication layer built on top of OAuth that allows users to be authenticated across multiple domains.

swipe >

7 Cross-Origin Resource Sharing (CORS)

A mechanism that allows many resources (e.g., fonts, JavaScript, etc.) on a web page to be requested from another domain outside the domain from which the resource originated.

API Testing

1 Postman

A popular tool for testing and debugging APIs.

2 SoapUI

A tool for testing SOAP and REST web services.

3 Swagger

A tool for designing, building, and testing APIs.

4 JMeter

A tool for testing the performance of APIs.

swipe >

5 TestRail

A test management tool for planning, executing, and tracking API tests.

6 Dredd

A command-line tool for testing API documentation against its backend implementation

7 REST Assured

A Java-based library for testing RESTful APIs.

8 Karate DSL

A testing framework for API testing using Gherkin syntax.

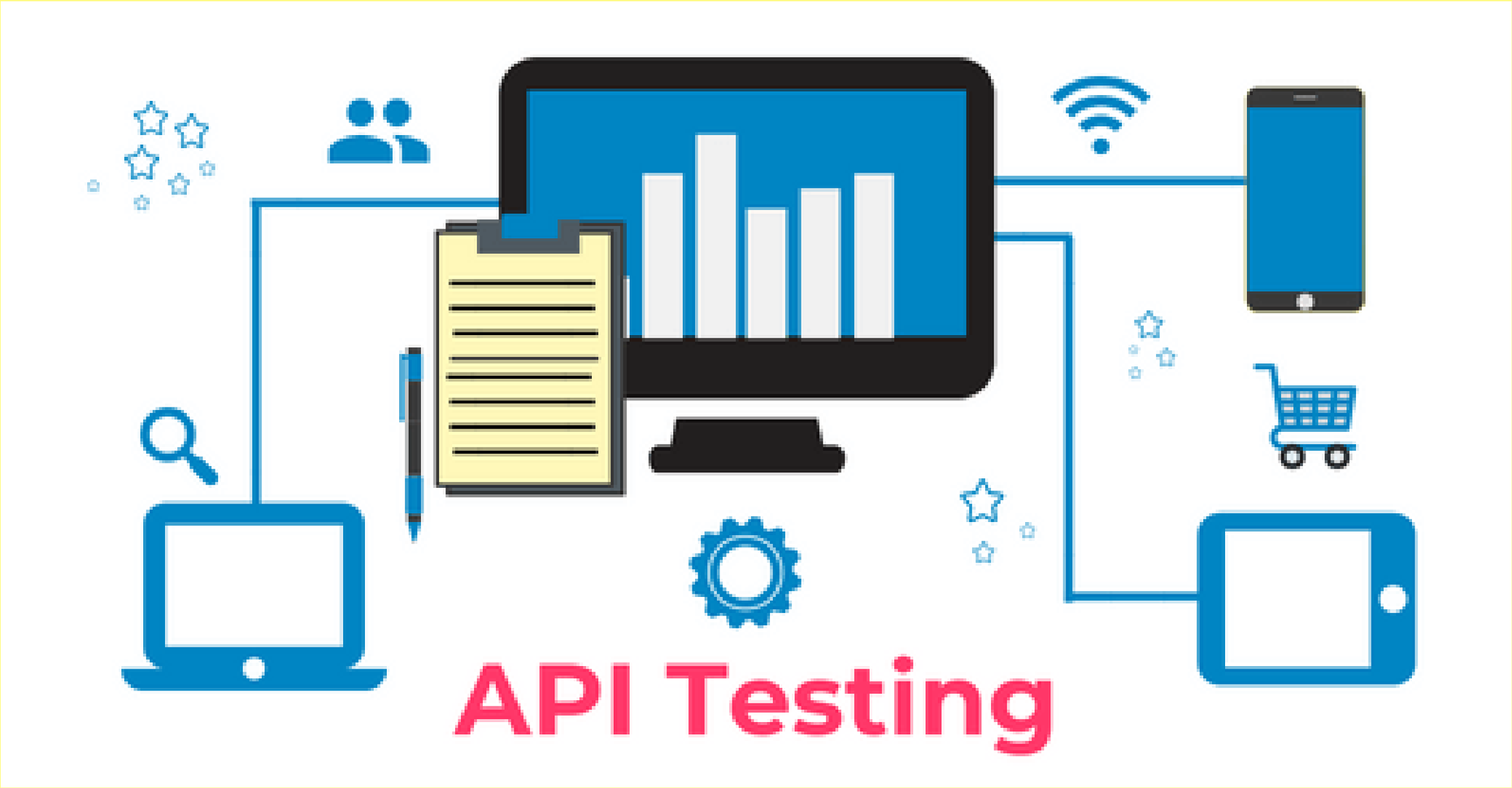
swipe >

9 HttpMaster

A tool for testing and debugging APIs

10 Assertible

A tool for testing and monitoring APIs with automated tests.



API Development

1 Node.js

A JavaScript runtime for building server-side applications.

2 Express

A popular framework for building web applications and APIs with Node.js.

3 Django

A Python web framework for building web applications and APIs.

swipe >

4 Flask

A lightweight Python web framework for building web applications and APIs.

5 Spring

A Java framework for building enterprise-level web applications and APIs.

6 Swagger Editor

A tool for designing and documenting APIs using the OpenAPI specification.

7 Postman

A tool for testing and debugging APIs.

swipe >

8 Insomnia

A tool for designing, testing, and debugging APIs.

9 Paw

A tool for designing and testing APIs on Mac OS.

10 API Blueprint

A high-level API description language for building RESTful APIs.

API Implementation Platforms

1 Firebase

A mobile and web application development platform developed by Google

2 Backendless

A mobile and web application development platform that allows developers to build and deploy applications without backend coding.

3 Parse Server

An open-source version of the Parse backend that can be deployed to any infrastructure.

swipe >

4 Amazon API Gateway

A fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs.

5 Microsoft Azure API Management

A fully managed service that enables users to publish, secure, transform, maintain, and monitor APIs.

Firebase

Amazon API Gateway

Azure

swipe >

API Performance

1 Caching

A technique for improving API performance by storing responses in a cache.

2 Throttling

A technique for limiting the rate of requests to an API to prevent overload.

3 Load Balancing

A technique for distributing traffic evenly across multiple servers to improve API performance.

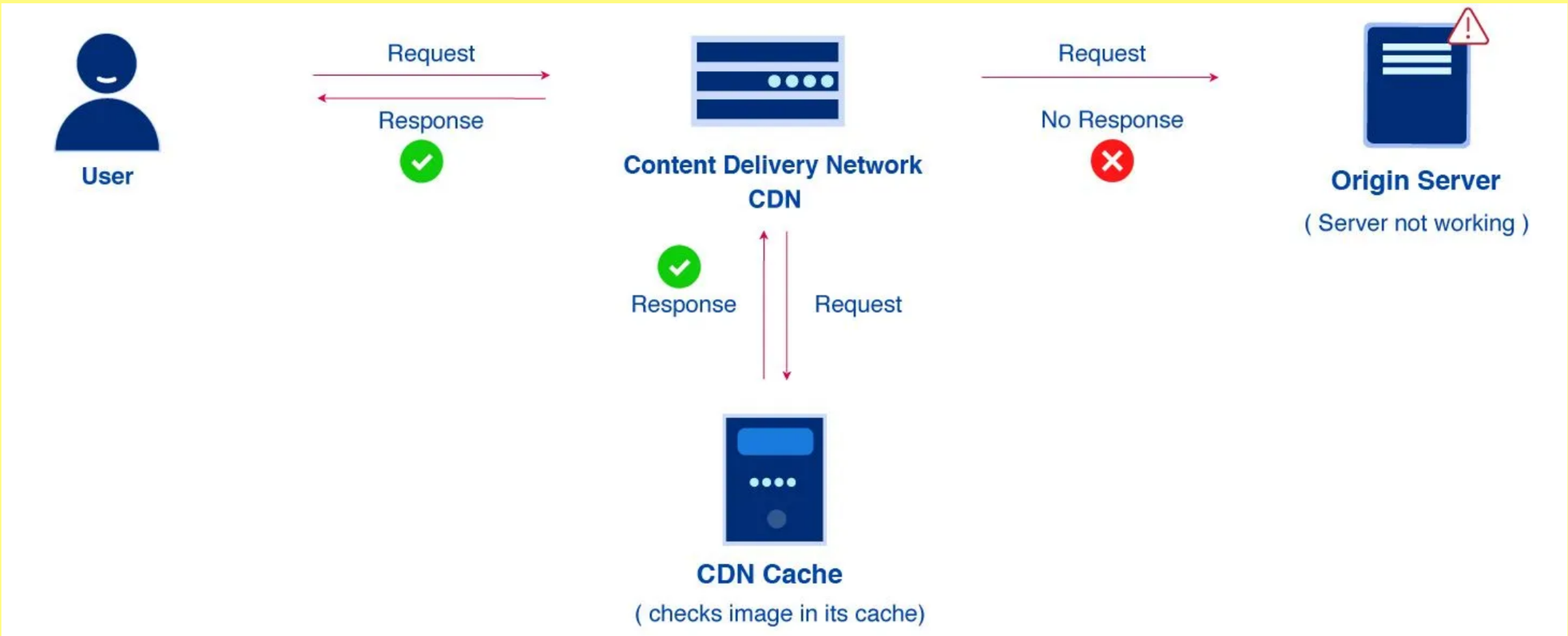
swipe >

4 Content Delivery Network (CDN)

A distributed system of servers that delivers content to users based on their geographic location to improve API performance.

5 Edge Computing

A computing paradigm that brings computation and data storage closer to the location where it is needed to reduce latency and improve API performance.



API Monitoring

1 Pingdom

A tool for monitoring the uptime and performance of APIs.

2 New Relic

A tool for monitoring the performance of APIs and other web applications.

3 Datadog

A monitoring and analytics platform for cloud-scale applications and APIs.

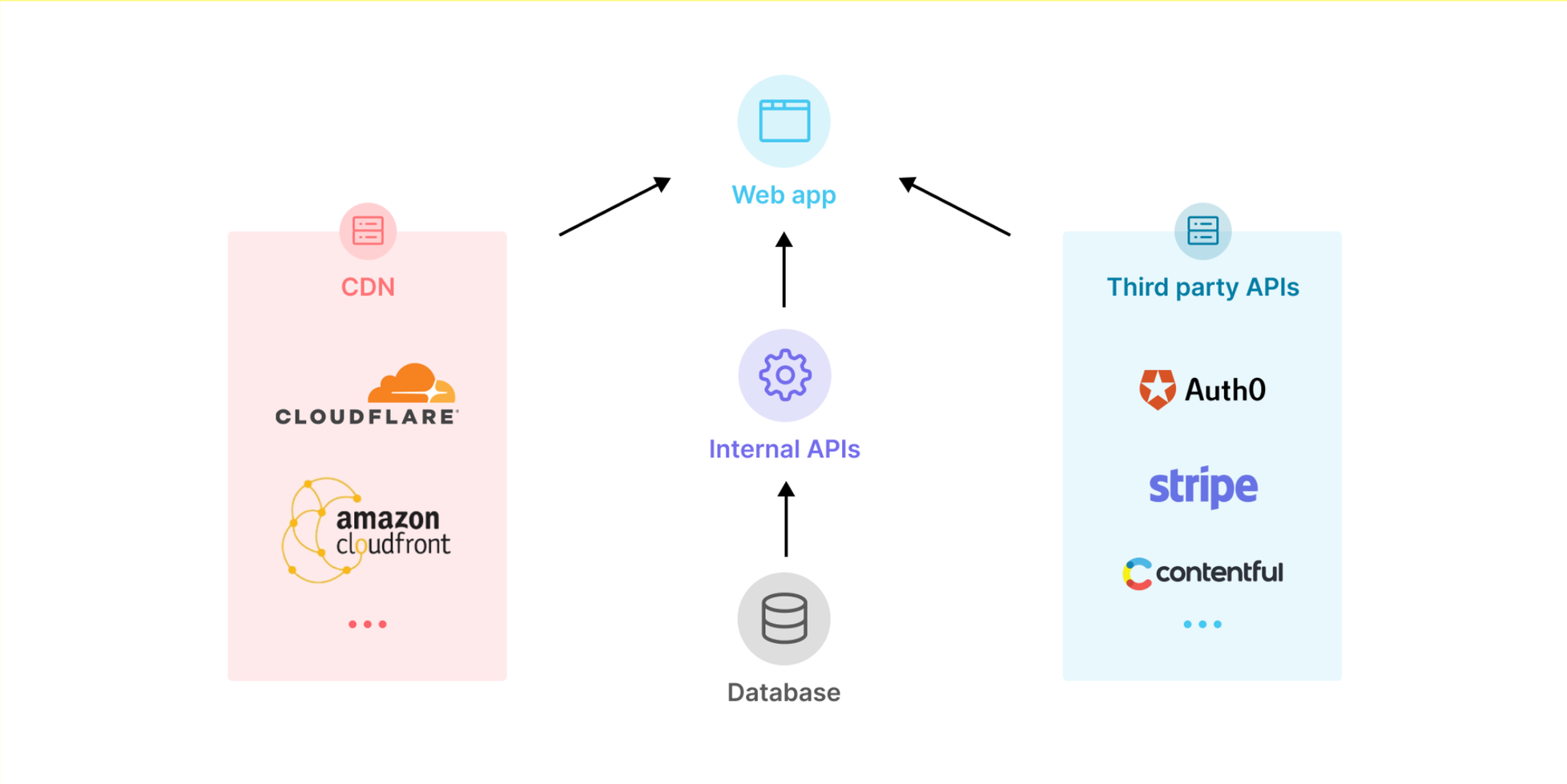
swipe >

4 Sumo Logic

A cloud-based log management and analytics platform for APIs and other applications.

5 Loggly

A cloud-based log management platform for monitoring APIs and other applications.



API Standards

1 JSON API

A specification for building APIs that use JSON as the data format.

2 HAL

Hypertext Application Language, a standard for building hypermedia-driven APIs.

3 JSON-LD

A format for representing linked data on the web.

4 OData

Open Data Protocol, a standard for building and consuming RESTful APIs.

5 AsyncAPI

A specification for building event-driven APIs.

swipe >

API Standards Organizations

1 W3C

The World Wide Web Consortium, an international community that develops web standards.

2 IETF

The Internet Engineering Task Force, an open standards organization that develops and promotes Internet standards.

3 OASIS

Organization for the Advancement of Structured Information Standards, a nonprofit consortium that drives the development, convergence, and adoption of open standards for the global information society.

swipe >

4 RESTful API Modeling Language (RAML)

A YAML-based language for describing RESTful APIs developed by MuleSoft.

5 JSON API

A specification for building APIs that use JSON as the data format.

W3C WORLD WIDE WEB consortium

IETF

RAML

Write once. Use many. Creative laziness encouraged.

DESIGN BUILD TEST DOCUMENT SHARE

API Infrastructure

1 Kubernetes

An open-source platform for managing containerized workloads and services.

2 OpenShift

A container application platform that builds on top of Kubernetes

3 Docker Swarm

A native clustering and orchestration solution for Docker

swipe >

4 Consul

A service mesh solution that provides service discovery, configuration, and segmentation capabilities

5 Istio

A service mesh solution that provides traffic management, security, and observability capabilities.

kubernetes

OPENSHIFT

Consul

swipe >

API Governance

1 API Management

The process of creating, publishing, and monitoring APIs in a secure and scalable way

2 API Monetization

The process of generating revenue from APIs by charging developers for usage

3 API Versioning

The process of managing changes to APIs over time

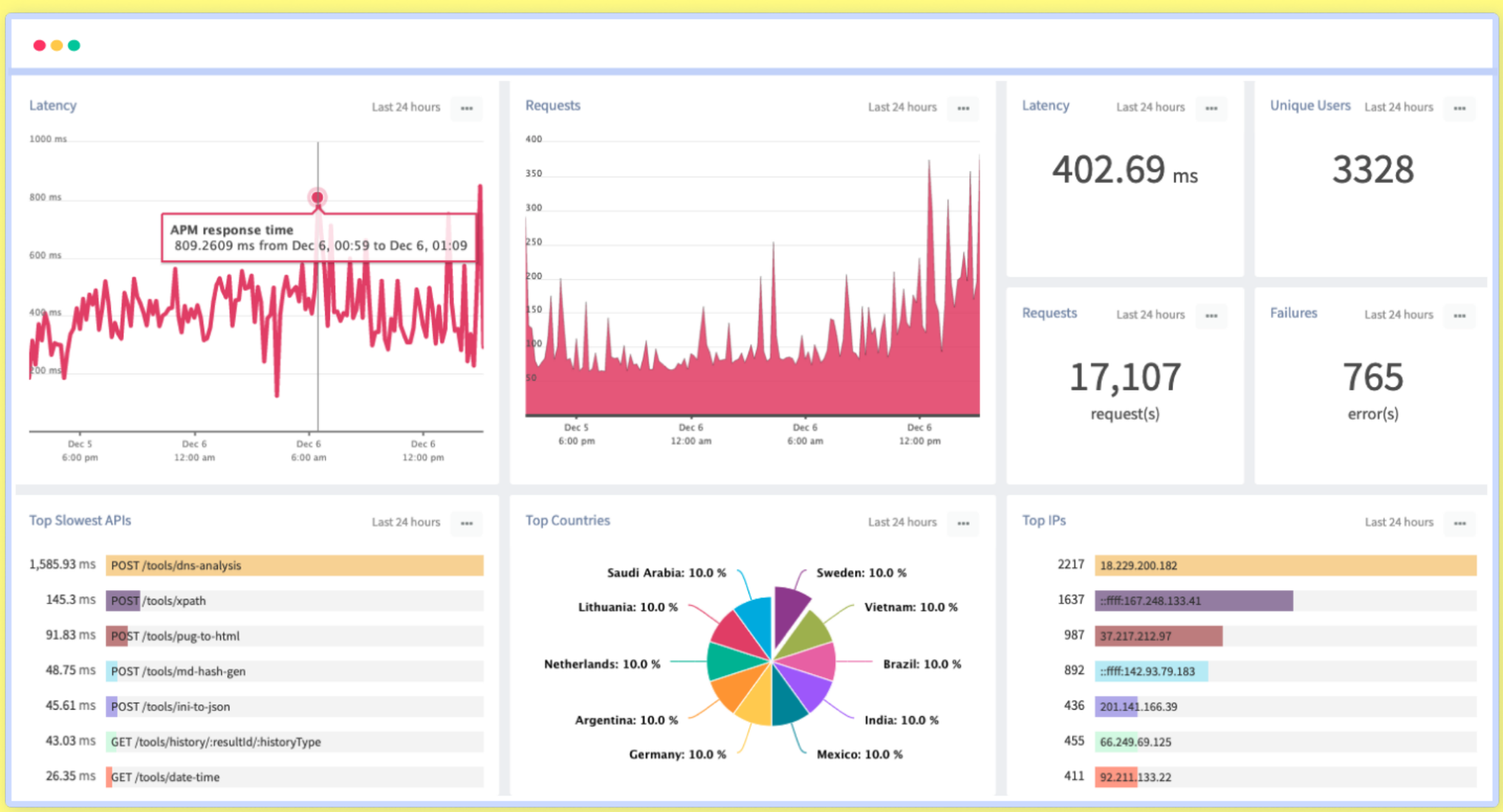
swipe >

4 API Analytics

The process of collecting and analyzing data on API usage and performance

5 API Gateway

A service that manages, protects, and scales APIs



swipe >

API Documentation

1 OpenAPI

A specification for building APIs in YAML or JSON format

2 API Blueprint

A high-level API description language for building RESTful APIs.

3 RAML

A YAML-based language for describing RESTful APIs.

swipe >

4 Swagger UI

A tool for visualizing and interacting with APIs that have been described using the OpenAPI specification

5 Slate

A tool for generating beautiful, responsive API documentation.

Swagger™

SLATE
API DOCS GENERATOR

swipe >

API Deployment

1 Heroku

A cloud platform for deploying, managing, and scaling web applications and APIs.

2 AWS Elastic Beanstalk

A service for deploying and scaling web applications and APIs on AWS.

3 Azure App Service

A service for deploying and scaling web applications and APIs on Azure.

swipe >

4 Google App Engine

A service for deploying and scaling web applications and APIs on GCP.

5 Docker

A containerization platform used for packaging and deploying applications.

6 AWS Lambda

A serverless compute service for running code in response to events.

swipe >

7 Azure Functions

A serverless compute service for running code in response to events.

8 Google Cloud Functions

A serverless compute service for running code in response to events.

9 Netlify

A cloud platform for deploying and managing static websites and APIs

swipe >

10 Vercel

A cloud platform for deploying and managing static websites and APIs.

HEROKU

Vercel

netlify

swipe >

API Best Practices

1 Versioning

A technique for managing changes to APIs over time.

2 Pagination

A technique for breaking up large API responses into smaller, more manageable chunks.

3 Caching

A technique for improving API performance by storing responses in a cache.

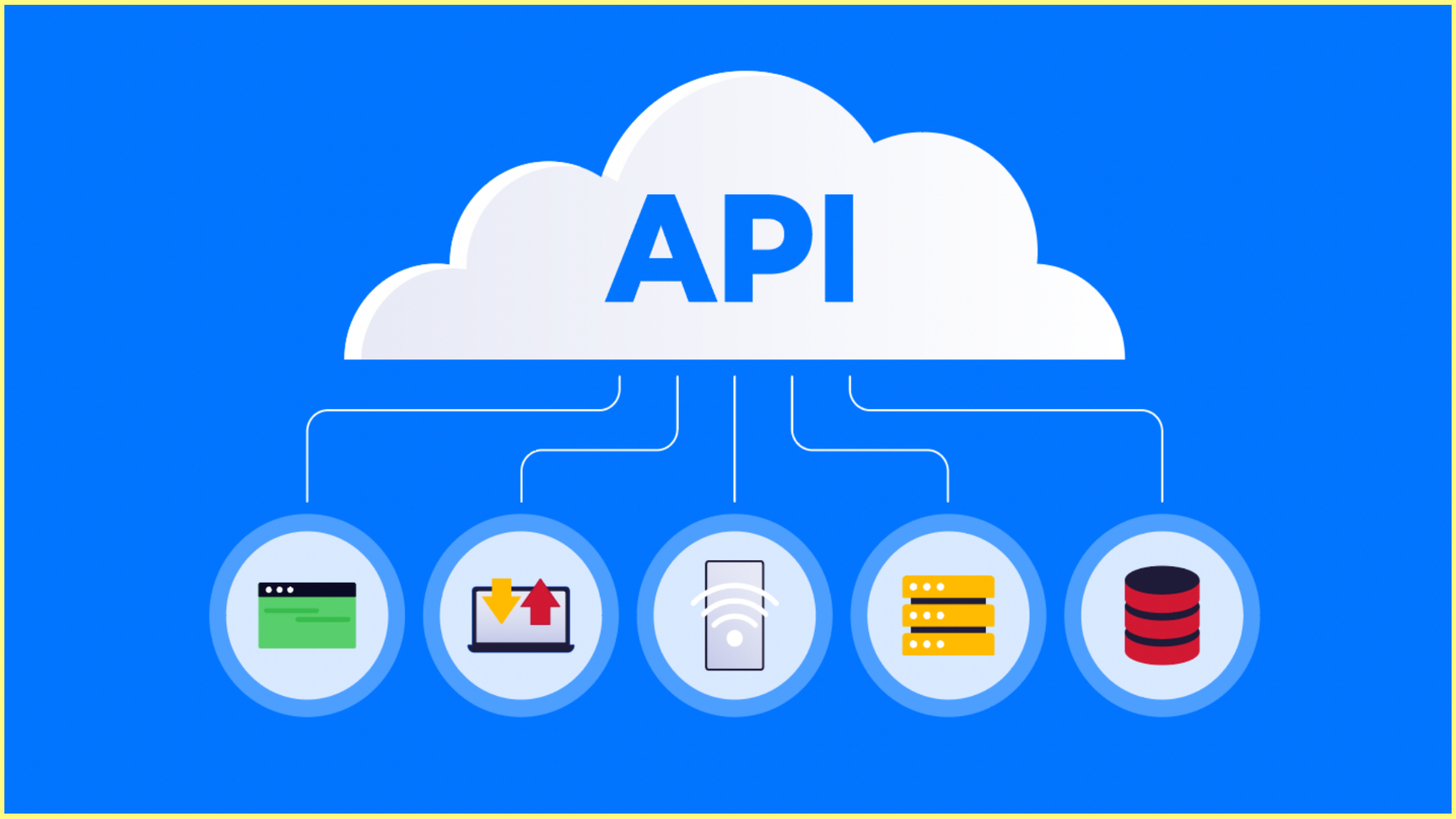
swipe >

4 Error Handling

A technique for returning meaningful error messages to API clients.

5 HATEOAS

Hypermedia as the Engine of Application State, a constraint of RESTful APIs that requires the API to provide links to related resources.



swipe >

API Tutorials

1. Getting Started with RESTful APIs by Tania Rascia
2. API Design Best Practices by Martin Fowler
3. Testing RESTful Web Services Made Easy Using the REST Assured Framework by Dinesh Rajput
4. API Gateway Concepts and Options by AWS
5. Building Secure APIs by Auth0
6. RESTful API Designing guidelines – The best practices by Mahesh Haldar

swipe >

API Guides

1. REST API Tutorial by Guru99
2. A Beginner´s Guide to HTTP and REST by Linode
3. REST API Design: Resource Modeling by Oracle
4. API Security Best Practices by Google Cloud
5. API Governance Handbook by WSO2

swipe >

API Tools

1 API Studio

A web-based IDE for designing and testing APIs.

2 Stoplight

A collaborative platform for designing, documenting, and testing APIs.

3 Apigee

A full lifecycle API management platform that allows developers to design, secure, deploy, and analyze APIs.

swipe >

4 Azure API Management

A fully managed service that enables users to publish, secure, transform, maintain, and monitor APIs.

5 Postman Learning Center

A hub for learning how to use Postman to design, develop, and test APIs.

Stoplight

apigee

swipe >

brijpandeyji

For More Interesting Content

Brij Kishore Pandey

Follow Me On LinkedIn

https://www.linkedin.com/in/brijpandeyji/

9 HTTP Methods

Stay ahead of the game with these essential HTTP methods!

swipe >

HTTP Protocol

HTTP stands for Hyper Text Transfer Protocol. Communication between client and web servers is done by sending HTTP Requests and receiving HTTP Responses.

HTTP Protocol is used to perform

CRUD operations (Create, Read, Update, Delete) by sending HTTP requests with different HTTP methods.

HTTP methods are also called as HTTP verbs.

HTTP Request Structure

Every HTTP request should have a URL or URI address and a method.

(Highlighted in purple)

Example URI

http://api.example.com/products?name=laptop&available=true

The format of a request message includes

- A request-line
- Zero or more header field(s) followed by CRLF (Carriage Return, Line Feed)
- An empty line
- A message-body (optional)

HTTP Methods in REST

REST

REpresentational State Transfer

It is a set of architectural principles for designing web services.

Web services based on REST Architecture are known as RESTful web services.

REST makes it easy to share data between clients and servers.

REST applications use HTTP methods like GET, POST, DELETE, PUT, etc., to do CRUD operations.

Categories of HTTP Methods

You can divide HTTP methods into two main categories :

- Safe HTTP Methods
- Idempotent Methods

Safe HTTP Methods

It doesn´t change data on the server. It always returns the same response, no matter how many times it gets called.

Example - GET, HEAD

Idempotent Methods

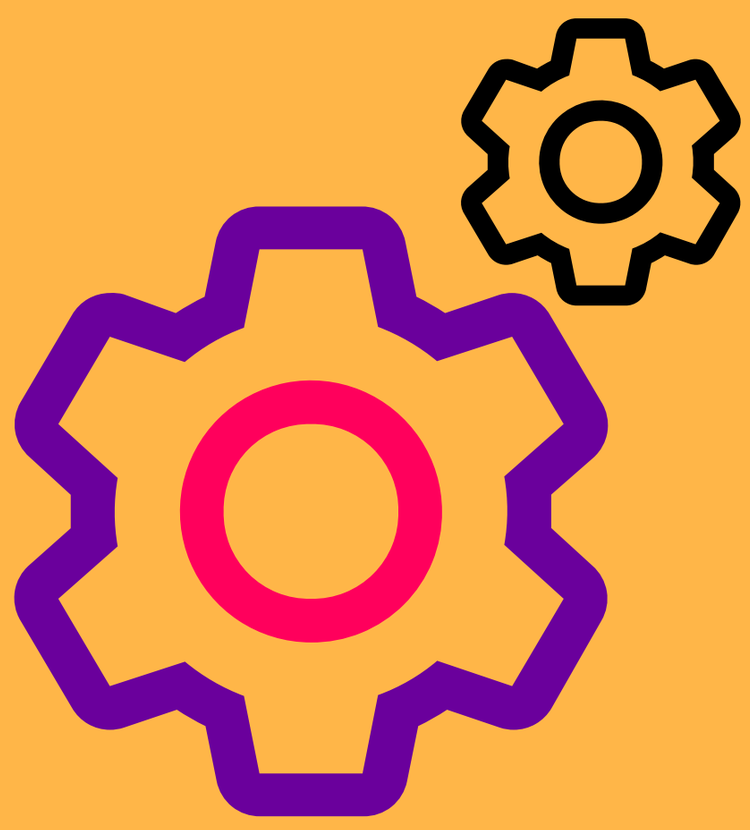
Example - GET, HEAD, PUT, DELETE, TRACE

It may change data on the server. It always returns the same response, no matter how many times it gets called.

All safe methods are also idempotent, but not all idempotent methods are safe.

swipe >

The 9 HTTP Methods



- GET Method
- POST Method
- PUT Method
- PATCH Method
- DELETE Method
- HEAD Method
- OPTIONS Method
- TRACE Method
- CONNECT Method

Let us now see them in detail.

GET Method

A GET Request is used to request information from a resource such as a website, a server, or an API.

Example

GET /api/employees/{employee-id}

Returns a specific employee by employee id.

GET /api/employees

Returns a list of all employees.

Since the GET method should never change the data on the resources and just read them(read-only), it is considered a safe method.

The GET method is also idempotent.

Test an API with a GET Method

When we want to test an API, the most popular method that we would use is the GET method.

Therefore, We expect the following to happen

- If the resource is accessible, the API returns the 200 Status Code, which means OK.
- Along with the 200 Status Code, the server usually returns a response body in XML or JSON format. So, for example, we expect the [GET] /members endpoint to return a list of members in XML or JSON.
- If the server does not support the endpoint, the server returns the 404 Status Code, which means Not Found.
- If we send the request in the wrong syntax, the server returns the 400 Status Code, which means Bad Request.

swipe >

POST Method

It creates a new resource on the backend (server). We send data to the server in the request body.

Example

POST /api/employees/department

Creates a department resource.

POST /api/employees/232/department/114/department-items

Creates a department item using the employee id and department id.

Two identical POST requests will create two new equivalent resources with the same data and different resource ids. We don´t get the same result every time.

It is neither a safe nor an idempotent method

Testing a POST Endpoint

Since the POST method creates data, we must be cautious about changing the data. Testing all the POST methods in APIs is highly recommended.

Here are some suggestions that we can do for testing APIs with POST methods

- Create a resource with the POST method, and it should return the 201 Status Code.
- Perform the GET method to check if it created the resource was successfully created. You should get the 200 status code, and the response should contain the created resource.
- Perform the POST method with incorrect or wrong formatted data to check if the operation fails.

PUT Method

Using this, we can update an existing resource by sending the updated data as the content of the request body to the server.

Example

PUT /api/employees/123

Update employee by employee id

If it applies to a collection of resources, it replaces the whole collection, so be careful using it. The server will return the 200 or 204 status codes after updating.

The PUT method is idempotent but not safe.

Test an API with a PUT Method

The PUT method is idempotent, and it modifies the entire resources.

Make sure to do the following operations

- Send a PUT request to the server many times, and it should always return the same result.
- When the server completes the PUT request and updates the resource, the response should come with 200 or 204 status codes.
- After the server completes the PUT request, make a GET request to check if the data is updated correctly on the resource.
- If the input is invalid or has the wrong format, the resource must not be updated.

PATCH Method

Similar to PUT, PATCH updates a resource, but it updates data partially and not entirely.

Example

```
PATCH /api/employees/123
{
"name" : "Brij"
}
```

Updates name for employee id 123.

The PATCH method updates the provided fields of the employee entity. In general, this modification should be in a standard format like JSON or XML.

It is neither a safe nor an idempotent method

Test an API with a PATCH Method

To test an API with the PATCH method, follow the steps for the testing API with the PUT and the POST methods.

Consider the following results

- Send a PATCH request to the server. The server will return the 2xx HTTP status code, which means, the request is successfully received, understood, and accepted.
- Perform the GET request and verify that the content is updated correctly.
- If the request payload is incorrect or ill-formatted, the operation must fail.

DELETE Method

The DELETE method deletes a resource. Regardless of the number of calls, it returns the same result.

Example

DELETE /api/employees/235

Delete employee by employee id.

Most APIs always return the 200 status code even if we try to delete a deleted resource but in some APIs, If the target data no longer exists, the method call would return a 404 status code.

The DELETE method is idempotent but not safe.

Testing a DELETE Endpoint

When it comes to deleting something on the server, we should be cautious. We are deleting data, and it is critical.

Then perform the following actions

- Call the POST method to create a new resource. Never test DELETE with actual data. For example, first, create a new employee and then try to delete the employee you just created.
- Make the DELETE request for a specific resource. For example, the request [DELETE] /employees/{employee-id} deletes a employee with the specified employee id.
- Call the GET method for the deleted employee, which should return 404, as the resource no longer exists.

HEAD Method

The HEAD method is similar to the GET method. But it doesn't have any response body, so if it mistakenly returns the response body, it must be ignored.

Example

HEAD /api/employees

Similar to GET, but it does not return the list of emloyees.

Before requesting the GET endpoint, we can make a HEAD request to determine the size (Content-length) of the file or data that we are downloading.

The HEAD method is safe and idempotent.

Testing a HEAD Endpoint

One of the advantages of the HEAD method is that we can test the server if it is available and accessible as long as the API supports it.

The API can be tested as follows

- It is much faster than the GET method because it has no response body.
- The status code we expect to get from the API is 200.
- Before every other HTTP method, we can first test API with the HEAD method.

OPTIONS Method

This method is used to get information about the possible communication options (permitted HTTP methods) for the given URL or an asterisk to refer to the entire server.

Example

OPTIONS /api/main.html/1.1

Returns permitted HTTP method in this URL

OPTIONS * HTTP/1.1

Returns all permitted methods

Various browsers widely use the OPTIONS method to check whether the CORS (Cross-Origin resource sharing) operation is restricted on the targeted API or not.

The OPTIONS method is safe and idempotent

Testing an OPTIONS Endpoint

Depending on whether the server supports the OPTIONS method, we can test the server for the times of FATAL failure with the OPTIONS method.

To try it, consider the following

- Make an OPTIONS request and check the header and the status code that returns.
- Test the case of failure with a resource that doesn't support the OPTIONS method.

TRACE Method

The TRACE method is for diagnosis purposes. It creates a loop-back test with the same request body that the client sent to the server before, and the successful response code is 200 OK.

Example

TRACE /api/main.html

Responds the exact request that client sent.

The TRACE method could be dangerous because it could reveal credentials. A hacker could steal credentials, including internal authentication headers, using a client-side attack.

The TRACE method is safe and idempotent

Test an API with a TRACE Method

- Make a standard HTTP request like a GET request to /api/status
- Replace GET with the TRACE and send it again.
- Check what the server returns. If the response has the same information as the original request, the TRACE ability is enabled in the server and works correctly.

CONNECT Method

The CONNECT method is for making end-to-end connections between a client and a server. It makes a two-way connection like a tunnel between them.

Example

CONNECT www.example.com:443 HTTP/1.1

Connects to the URL provided.

For example, we can use this method to safely transfer a large file between the client and the server.

It is neither a safe nor an idempotent method

These are the 9 HTTP methods, their uses and a guide on how to test them. Hope you learned about them in detail.

swipe >

brijpandeyji

For More Interesting Content

brijpandeyji

Follow Me On LinkedIn

https://www.linkedin.com/in/brijpandeyji/

brijpandeyji

API Terminology Handbook

The Ultimate Guide to Rest API Terms and Glossary

swipe >

Contents

- API
- API Call
- API Economy
- API Endpoint
- API Integration
- API Gateway
- API Lifecycle
- API Request
- API Keys
- API Layer
- API Portal
- API Security
- Apigee
- APIsec
- Application
- Framework
- Burp Suite
- CI/CD
- CRUD
- Cache
- Client
- DDoS
- Resource
- Request
- Response
- Response Code
- Payload
- Pagination
- Method
- Query Parameters
- Authentication
- Rate Limiting
- API Documentation
- Logic Flaw

swipe >

Contents

- JSON
- Microservices
- Monetization
- OWASP
- ZAP
- Parameters
- Penetration Testing
- Production Environment
- REST
- Red Teams
- SDK
- SDLC
- SOAP
- SQL Injection
- Webhook
- Over-Permissioned Container

END

API

Application Programming Interface is what API stands for. API is a set of definitions and protocols that allow technology products and services to communicate via the internet.

API Call

The API call is simply the process of sending a request to your API after setting up the right endpoints. Upon receiving your information, it is processed, and you receive feedback.

By entering your login and password into a website and hitting `enter´, you made an API call.

swipe >

API Economy

The API economy is a term to describe the exchange of value between a user and an organization.

It enables businesses to leverage APIs from other providers such as Google to power their own apps, allowing an ecosystem that makes it possible for users to get value from a platform without having to build the APIs from scratch.

For Example: Uber uses API calls to connect with Google Maps.

API Endpoint

An endpoint is the end of a communication channel. When APIs interact with other systems, each touchpoint of interaction is considered an endpoint.

For example, it could be a server, a service, or a database where a resource lives.

API Integration

In simple terms, API integration connects two or more applications to exchange data between them and connect to the outside world.

API Gateway

An API gateway is an API management tool that serves as an intermediary between the client and a set of different backend services.

API gateways act as gatekeepers and proxies that moderate all your API calls, aggregate the data you need, and return the correct result.

Gateways are used to handle common tasks such as API identification, rate limiting, and usage metrics.

API Lifecycle

The API lifecycle is an approach to API management and development that aims at providing a holistic view of how to manage APIs across its different life stages, from creation to retirement.

The API lifecycle is often divided into three stages, the creation stage, the control stage, and the consumption stage.

API Request

APIs are everywhere and are part of every aspect of the web. An API request happens when a developer adds an endpoint to a URL and uses that endpoint to call the server or the database.

API Keys

An API key is a unique identifier that enables other software to authenticate a user, developer, or API calling software to an API to ensure that this person or software is who it says it is.

API keys authenticate the API instead of a user and offer a certain degree of security to API calls.

API Layer

An API layer is a proxy that joins together all your service offerings using a graphic UI to provide greater user interactivity. API layers are language-agnostic ways of interacting with apps and help describe the services and data types used to exchange information.

API Portal

An API portal is a bridge between the API provider and the API consumer.

API portals serve to make APIs public and offer content to educate developers about them, their use, and how to make the most of them.

An API portal provides information about the APIs at every stage of the API lifecycle.

API Security

API security is an umbrella term that defines a set of practices that aim to prevent malicious attacks, misuse, and exploit APIs.

API security includes basic authentication and authorization, tokens, multi-factor authentication, and other advanced security measures.

The ubiquitous nature of APIs makes them one of the favorite targets for hackers.

swipe >

Apigee

Apigee is an API gateway management tool offered by Google to exchange data across cloud services and applications.

As a proxy layer, Apigee enables you to expose your backend APIs in abstraction or facade and helps protect your APIs, limit their rate, and provide analytics and other services.

It enables developers to build and manage APIs.

APIsec

APIsec is an API security company. It leverages automated testing tools to find logic flaws before your code hits the production stage.

APIsec addresses the business need to secure APIs before they reach production and provides the industry's only automated and continuous API testing platform that uncovers security vulnerabilities in APIs.

Application

Application software is commonly defined as a program or a bundle of different programs designed for end-users.

Every program can be called an application, and often the terms are used interchangeably.

Framework

A framework contains libraries of code, instructions, and APIs from which developers and API consumers can obtain information from an app.

Burp Suite

Burp -also called Burp Suite- is a set of tools used for penetration testing of web apps.

Burp is an all-in-one penetration testing suite that offers users a one-stop shop for all their pen testing needs.

BurpSuite contains an intercepting proxy that lets the user see and modify the contents of requests and responses while they are in transit for granular control of your APIs.

swipe >

CI/CD

Continuous integration (CI) and continuous deployment (CD) are a set of operating principles and a collection of practices and agile methodologies that enable development teams to deliver better and faster changes to their code.

CI/CD is one of the most important DevOps practices as it gives teams the tools to focus on meeting their business requirements, code quality, and security needs.

CRUD

CRUD is an acronym for create, read, update and delete. It refers to the necessary functions to implement a storage application, such as a hard drive.

Unlike random access memory and internal caching, CRUD data is typically stored and organized into a database, which is simply a collection of data that can be viewed electronically.

Cache

The cache is a software or hardware component that stores data so users can access and retrieve that data faster. Cached data might be the result of a copy of certain data stored elsewhere.

Cache reads data and retrieves it faster than you would otherwise.

Client

A client is a device that communicates with a server. A client can be a desktop computer, a laptop, a smartphone, or an IoT-powered device. Most networks allow communication between clients and servers as it flows through a router or switch.

DDoS

A distributed denial of service (DDoS) attack is a malicious attack that aims at disrupting the target's traffic.

It usually overwhelms the target's infrastructure with a flurry of internet traffic aimed at saturating the servers and causing them to shut the page down.

Resource

An entity that can be represented by a URI and can be accessed through an API. Resources can be anything from data (such as a list of users or a single user´s profile) to operations (such as creating or updating a resource).

Request

An HTTP request sent by a client to a server to retrieve or modify data. A request typically includes a method, a URI, and a set of headers and/or a body.

Response

An HTTP response sent by a server to a client in response to a request.

Response Code

A numerical status code returned in an API response to indicate the success or failure of a request. Common response codes include 200 (OK), 404 (Not Found), and 500 (Internal Server Error).

Payload

The data sent in an API request or response, often in the form of a JSON object.

Pagination

A technique used in APIs to divide a large dataset into smaller, more manageable chunks or pages. This allows a client to request a specific page of data rather than receiving the entire dataset all at once.

Method

The HTTP verb used in an API request, such as GET, POST, PUT, or DELETE.

Query Parameters

Key-value pairs that are added to the end of an API endpoint URL to specify certain criteria or filters for the data being requested.

Authentication

The process of verifying the identity of a client or user before allowing them to access an API. This is often done using an API key or other form of credentials.

Rate Limiting

The process of limiting the number of API requests that a client can make within a certain timeframe to prevent abuse or overuse of the API.

API Documentation

Detailed documentation or reference material provided by the creator of an API, explaining how to use the API and its various endpoints and parameters.

Logic Flaw

Business logic flaws result from faulty application logic. In simple terms, a logic flaw happens when an application behaves unexpectedly. A logic flaw allows attackers to misuse an application and circumvent its rules to change how it performs.

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format based on a subset of JavaScript programming language standards.

JSON has the advantage that it is both easy for humans to read and write and for machines to parse and generate.

It is a format that is completely agnostic to languages and uses conventions that are familiar to programmers of C-family languages.

Microservices

Microservices are also known as microservices architecture. It is a software architecture style that structures apps as a collection of loosely coupled, independent.

Microservices are highly maintainable services that are organized to enhance an app, website, or platform's business capabilities.

Monetization

API monetization is a process by which a business can create revenue from its APIs.

Since APIs enable users to access and integrate data from different sources, they can be used by different developers to integrate relevant services within their products, digital services, or applications, which could, in turn, become a source of revenue for both public and private services and applications.

OWASP

OWASP (Open Web Application Security Project®) is a nonprofit organization dedicated to enhancing software security.

OWASP offers a range of tools to help developers and programmers secure the web through open-source software projects, hundreds of local chapters worldwide, and educational and training events.

Over-Permissioned Container

An over-permissioned container is a container that has all the root capabilities of a host machine.

That means that it can access resources that aren´t accessible to ordinary containers and users.

The problem with over-permissioning is that it gives malicious actors a point where they can attack your infrastructure and compromise your implementation.

Parameters

Parameters are special types of variables used in computer programming to pass information between procedures and functions.

An argument to a function is referred to as a parameter. Adding three numbers, for example, may require three parameters.

Penetration Testing

Also called pen testing or ethical hacking, penetration testing simulates attacks on your computer system to identify exploitable vulnerabilities.

Pen testing identifies, tests, and highlights vulnerabilities in an organization's security posture.

Web application firewalls (WAF) are generally augmented by penetration testing in the context of web application security.

Production Environment

In a production environment, software and other products are actually put into operation in how their intended users intend them to be used.

Developers generally use this term to refer to the setting where end-users will actually use the products.

In a production environment, software programs and hardware are run in real-time, and they are relied on daily by organizations and companies for their daily operations.

REST

Created by Roy Fielding, a computer scientist, REST, which stands for REpresentational State Transfer, is an application programming interface that conforms to the constraints of REST architectural style and enables a quicker interaction between different RESTful web services.

A stateless Web service must be able to read and modify its resources using a predefined set of operations and a textual representation.

Red Teams

Red teams are cybersecurity professionals trained in attacking systems and breaking into them by finding compromised entry points or exploitable logic flaws.

The objective of the red team is to improve a company's cybersecurity standing by showing it how they managed to gain access and exploit their system vulnerabilities.

SDK

SDK stands for software development kit and is a set of instructions, integrated practices, pieces, code samples, and documentation that enables developers to create software applications on a specific software platform.

SDKs can be seen as workshops with everything developers need to build specific software for a determined platform.

SDLC

SDLC -also called software development lifecycle- is the process for planning, creating, testing, and deploying an information system.

SDLC aims at producing quality software at the lowest cost in the shortest time possible.

SDLC gives developers a structured flow divided into phases to help companies produce high-quality software.

swipe >

SOAP

Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information to implement web services.

SOAP leverages XML Information Set for message format and other application-layer protocols, such as HTTP or SMTP for message transmission. The messaging services provided by SOAP are exclusively XML-based.

Microsoft originally developed the SOAP protocol to replace old technologies such as Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA) that cannot work over the internet.

SQL Injection

An SQL injection technique is a way to inject code into a database that may damage it.

SQL injections are one of the most common web hacking techniques and rely on the placement of malicious SQL code in SQL statements via web input using forms or other editable fields.

Webhook

A webhook (also called a web callback or HTTP push API) is a way for an app to provide other applications with real-time information.

Webhooks deliver data directly to other applications, so data is available immediately instead of standard APIs requiring frequent polling for real-time data.

Webhooks are beneficial to both consumers and providers in this way, but the only drawback is the difficulty of setting them up at first.

swipe >

ZAP

Also called OWASP Zed Attack Proxy (ZAP) is one of the world's most popular free security tools, which lets you automatically find security vulnerabilities in your applications.

By automating penetration testing and security regression testing, developers can automate an application's security testing during the CI/CD process.

With ZAP, you can also do nearly everything you can do with the desktop interface using its powerful API.

brijpandeyji

For More Interesting Content

Brij Kishore Pandey

Follow Me On LinkedIn

https://www.linkedin.com/in/brijpandeyji/