**REST API**

**API** stands for Application Programming Interface. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses.

**REST** (Representational State Transfer), is an API that follows a set of rules through which applications and servers communicate. It was specifically designed for working with components like files, objects, and media components.

A REST API is an interface that allows different applications to interact with each other over the web

**The most common operations are**

**GET** : This would be used to get/retrieve data.

**POST** : This would be used to create a new entity.

**PUT** : This would be used to update the data.

**PATCH** : performs a partial update of a resource. The request body specifies the set of changes to apply to the resource.

**DELETE** : This would be used to delete the data.

**REST API Security: Best Practices**

**Always use HTTPS:** You should always use SSL to ensure higher security standards.

**Use password hashing:** Password hashes add another layer of security, protecting the integrity of sensitive data even if a password was compromised.

**Avoid exposing sensitive data in URL strings:** Any data that hackers can potentially use to break into your system, from usernames to session tokens, must not be included in the URL string.

**Implement OAuth:** OAuth is a widely recognized authorization framework allowing data exchange without exposing sensitive information.

*HTTP is used for basic data communication, HTTPS is essential when security and data privacy are concerns.*

* *Data sent using HTTP is transmitted in plain text, making it vulnerable to interception and tampering by attackers.*
* *HTTPS ensures that data is encrypted before being sent and can only be decrypted by the intended recipient.*

**RESTful Principles**

### **Uniform interface**

Fundamental to any RESTFull web service, it indicates the transfer of data in a standard way. It establishes 4 architectural constraints.

### **Stateless technology**

When we speak of “stateless” technology, we’re talking about a communicative method in which the server completes the various requests made by the client, regardless of the previous ones.

1. **Layered system**

In this type of “layered” architecture, clients are able to communicate with other pre-authorized intermediaries while still receiving a response from the server, despite the “additional” communications.

1. **Caching**

This type of architecture can be cached, speeding up the processing of some responses in the client’s cache or in an intermediary, increasing the response time between client requests.

1. **Code on demand**

By passing software programming code to the client, REST-style servers can augment or momentarily alter the client’s functionality.

Take for example:: the forms you find on any website. If you fill it out and there’s a mistake, like the wrong phone number, the server detects it and points it out so that the user corrects it.

## **5 Advantages of REST APIs**

### **1. Interoperability**

REST APIs are based on the same standards used for the web. Same language, same code and architecture. **They are highly interoperable and can easily interact**, synchronize and integrate with other applications.

As a result, APIs are part of the entire web environment, from blogs and social networks to CMSs and development platforms.

### **2. Flexibility**

REST APIs can communicate using any data format. In other words, they can be adapted to almost any application on the web regardless of its format, language or architecture.

### **3. Scalability**

**REST APIs are highly scalable.**They can handle thousands, millions, and billions of data simultaneously without affecting performance.

In other words, they can process huge blocks of information while the two websites, connected through the REST API work seamlessly.

### **4. Security**

REST APIs typically use authentication via access tokens,

Tokens are much more difficult to crack as they are unique. In addition, they can be supplemented with other authentication methods to double the security standards.

### **5. Ease of use**

REST APIs are simpler and easier to use than other APIs.

## **Disadvantages of a REST API?**

### **1. Increased design complexity.**

Although they are easier to use, the design of a REST API can be more complex than other APIs, especially if you are not familiar with web architecture.

### **2. Web connection**

All changes to your REST API must be executed on the web and only on the web. It’s impossible to edit the API from your desktop without an internet connection, unlike HTML web files, for example. You must always connect to make the slightest change.

### **3. Variable performance and flexibility**

On the other hand, REST APIs can have slightly lower performance than other APIs, depending on the servers and their internet speed.

And in some cases, a REST API may be less flexible than other APIs because of the architectural principles it must follow.

**Client request are consist of:**

### **1. Unique number of resources**

Using specific resource identifiers, the server recognizes each resource. It usually uses a uniform resource locator or URL to identify resources in REST services. The path to the resource is specified by the URL.

### **2. Method**

### **3. HTTP headers**

All metadata that is transmitted between the client and the server is contained in the request headers.

One can say that the request header sets out what the request and response formats are, to provide details on the status of the request.

**REST API parameters:**

**1. Header Parameters**

**2. Path Parameters**

**3. Query String Parameters**