HubSpot

How to Use an API

Introduction to APIs

Picture your morning routine: you wake up, grab your phone, check the Weather app, and review how your investments are doing in the the Robinhood app.

Before you leave your bed, you're already using APIs to organize and operate your daily life.

The weather app uses various APIs to provide weather information. Among these APIs, the Channel API is a key component, which grants access to both current and predicted weather data for various locations globally. By using this API, the weather app can request the latest weather information for the user's current location.

APIs also ensure that the stock prices and other financial data are always up-to-date and accurate. This task is handled by IEX Cloud API within the Robinhood app, which relays an accurate, real-time price for stocks for companies like HubSpot and Tesla.

An API, or Application Programming Interface, is an essential tool for modern software development. APIs allow for different software applications to communicate with each other, share data and functionality, and work together to create powerful and efficient systems.

In this ebook, we will cover everything you need to know about APIs, starting with the basics of what APIs are and how they work. We will then explore the different types of APIs and their use cases, including REST APIs, SOAP APIs, and GraphQL APIs.

In addition, we will cover important topics such as API security, authentication, and best practices for designing and consuming APIs. By the end of this ebook, you will have a deep understanding of APIs and be ready to start using them in your own projects.

HubSpot

Table of Contents

05 A History of APIs

06 Kinds of APIs and How They're Used

08 How to Understand API Documentation

09 How to Use an API

11 Common Error Types

15 What are REST APIs?

17 Security and Authentication

What is an API?

An API, or Application Programming Interface, is a set of programming instructions and protocols used to enable communication between software applications. Essentially, an API acts as an intermediary that facilitates interaction between different software programs by defining how they can communicate and exchange information.

APIs have become increasingly important in the modern software landscape, as they allow developers to build complex systems that integrate with other software applications, services, and platforms. APIs allows software developers to leverage the functionality of existing software applications and services, rather than having to build everything from scratch. This can save time and effort, and allow for more rapid development and innovation.

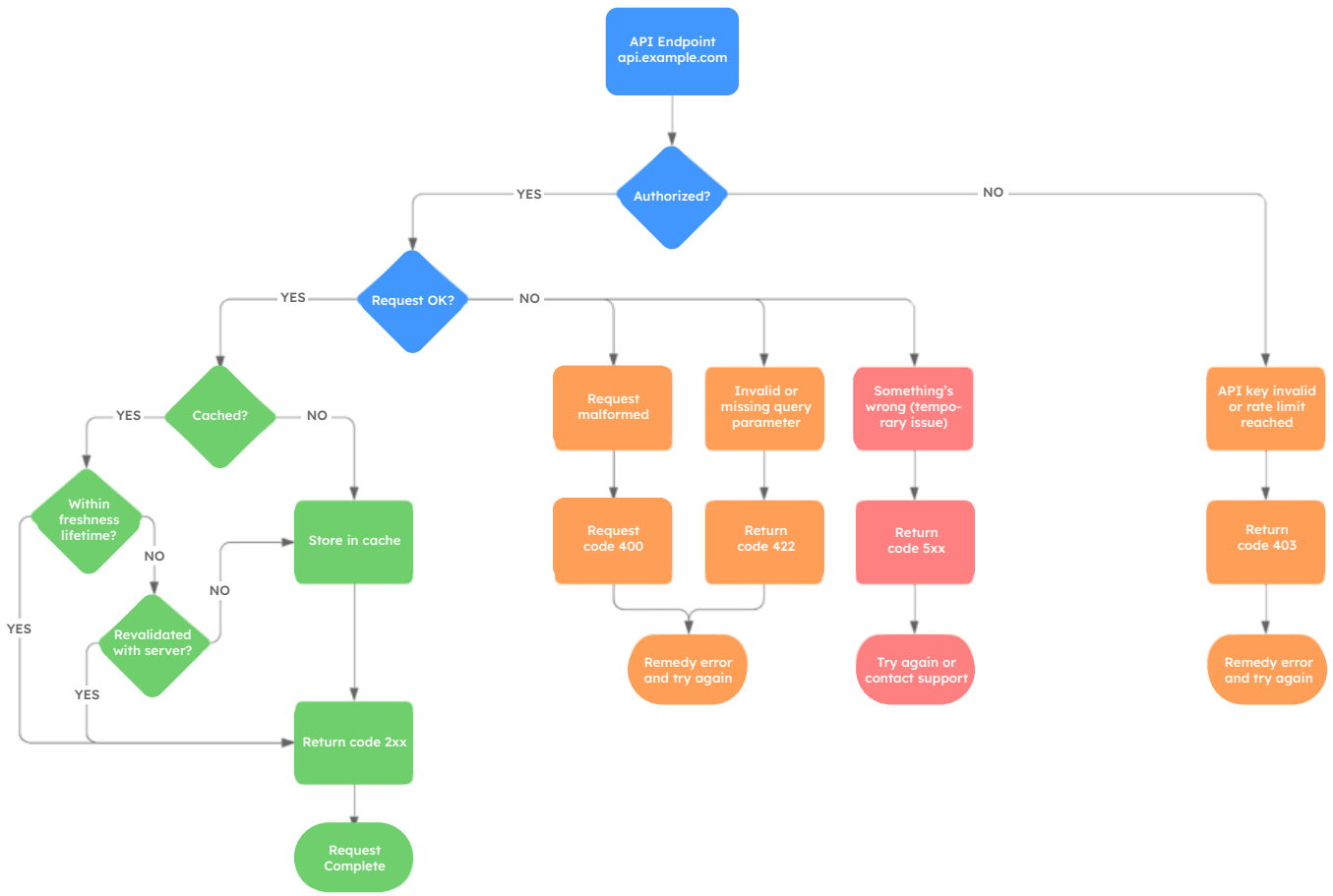


Image Source

This API diagram includes several key components. The first is the API itself, which acts as an interface between different applications. The API may consist of several sub-components, such as authentication and authorization mechanisms, data models, and request/response formats.Overall, APIs are crucial for modern software development, enabling integration between different applications, platforms, and services to create more powerful and interconnected systems.

A History of APIs

APIs, or Application Programming Interfaces, have a long history that dates back to the early days of computer programming.

1960s-70s: During this era, the first APIs were used by software developers to communicate between different systems — mostly centralized computing systems used by organizations for critical data processing and transactional operations.

1980s-90s: The introduction of personal computers and the rise of local area networks (**LANs**) saw the development of APIs for operating systems like Windows and Macintosh. These APIs enabled software developers to create applications that could run on these platforms.

2000s: With the explosion of the internet and the rise of web applications, eBay, Google, and Amazon, APIs took on a new role. They were used to allow websites and applications to communicate with each other and exchange data. For example, the Google Maps API allowed developers to embed Google Maps into their own applications and websites. The Amazon API was another powerful tool for third-party advertisers and sellers allowing them to attract the right traffic for their products and advertisements.

2010s: With the growth of mobile computing and the rise of mobile apps, APIs became an essential component of mobile app development. Mobile APIs allowed developers to access the capabilities of a device, such as its camera or accelerometer, and to exchange data with servers.

2020s: APIs continue to play a critical role in the development of modern applications. They are used to integrate systems, provide access to data and services, and enable collaboration between different parties. With the growth of cloud computing and Web3, APIs are becoming even more important as they allow for the creation of highly scalable, secure and flexible applications and smart contracts.

APIs will continue to evolve and remain the highways of modern software development, allowing systems to communicate and exchange data in a standardized, secure and efficient manner.

Kinds of APIs and How They're Used

Here is a list of different types of APIs and how they are used:

- **Rest APIs (Representational State Transfer):**

  These APIs are commonly used because of their simplicity and reduced cache load, which can improve performance and reduce the load on a server.

- **SOAP API (Simple Object Access Protocol):**

  A structured protocol critical to the introduction of web services, a SOAP API uses XML for the format of sending and receiving data, designed to be more secure than REST APIs.

- **GraphQL API:**

  Developed by Meta in 2012 and open sourced in 2015, a GraphQL API is defined as a query language for APIs. GraphQL allows for requesting clients to receive exact data rather than a fixed structure of data from requests performed with a traditional REST API.

Here are different ways APIs are used throughout some of the industries that power the world we live in.

Finance:

Financial institutions as we know today couldn't exist without the use of APIs. APIs power the infrastructure of pushing and pulling real-time data, which is a necessity for accuracy and security for financial transactions and price stability within the stock and cryptocurrency market, as well as providing account information for users and businesses alike.

Transportation:

Transportation services manage traffic, real-time ride request, cargo information, and countless other services efficiently with the use of APIs integrated with other software. For example, Google Maps API allows developers to access maps and route information for their own applications.

Ecommerce:

Ecommerce companies use APIs to maximize functionality of applications and websites, pushing and pulling customer information, order requests, printing of labels, and even managing shipping arrangements — all while allowing payment gateways and choosing advertisements to persuade your next likely purchase.

Social Media:

Social media companies APIs allow for developers to leverage functionality and data within the applications to track user analytics and behavior. These analytics are then used to generate a better user experience and label each user for demographic based content and advertisements. This seamless integration between the website and Facebook's API allows for easy sharing of content across different platforms.

Marketers vs. Developers

Marketers most commonly use APIs to embed third-party software applications to manage and connect an array of services provided by APIs. Examples include gaining insights about customers and creating a marketing campaign that targets the audience with effective advertising. Developers are the ones who assemble software applications that allow for the communication and transferring of data between applications and services. These unique applications can leverage functionality and services of APIs to enhance third-party applications. Both marketers and developers use APIs to automate tasks and increase efficiency within their tasks.

How to Understand API Documentation

Documentation of APIs are a set of guidelines and instructions derived from software developers to assist additional developers to onboard the functionality of their API software. Documentation sets the boundaries and limitations of accessing data and interacting with services, devices, and systems.

To understand each API and its requirements, you need to consider the following:

- **API documentation:**

  API documentation needs to understood to begin the process of putting it to use, give a thorough overview of the documentation and understand the provided details of methods, parameters, and responses from the API.

- **Authentication and Authorization:**

  Authentication and Authorization are essential for accessing many APIs, but not all. Typically, keys or tokens are used to authenticate and authorize API requests. To obtain these keys or tokens, you must register with the API provider.

- **Data Format:**

  To extract data from APIs, developers must understand that the data may be provided in various formats, including JSON (JavaScript Object Notation) and XML (Extensible Markup Language). These formats are commonly used, and it is crucial to comprehend them to begin the internal formatting process and extract the necessary data.

- **API Endpoints:**

  An **API endpoint** is the URL that it uses to access the API. Endpoints set boundaries and limitations on what can be requested from the API ensuring your requests are answered with the correct data.

- **Rate Limits:**

  A rate limit is an imposed request limit, also known as transactions per second. These limits are set in place to prevent throttling an abuse of the API services. Having a good metric of rate limits will prevent failed requests and denied access.

- **Error Handling:**

  Proper error handling documentation should provide details about the error codes that the API returns, along with explanations of what each error code means. It should also provide guidance on how to handle each error code, including what actions the developer should take to resolve the error. Understanding error handling can prepare developers for unexpected errors.

Understanding these error codes and understanding each API and its requirements includes reading the documentation, understanding authentication and authorization, knowing the correct data format, and accessing the correct endpoint. These are the skills required to effectively have APIs access necessary data and gain access to service functionality.

How to Use an API

Now that you have a basic understanding of what an API is, let's talk about how to actually use one.

The randomuser.me API is a free service that generates random user data, including names, email addresses, phone numbers, addresses, and more. This data is intended to be used for testing and prototyping applications that require user data. The API is available for anyone to use, and it is a popular choice for developers and testers who need to generate large amounts of user data quickly and easily.

Using the randomuser.me API is simple. To get started, you need to make a request to the API with the desired parameters. For example, you can specify the number of users you want to generate, the nationality of the users, and the gender of the users. The API will then generate the requested number of random users with the specified parameters.

Understanding Documentation:

Before you can use any API, it's important to read and understand the documentation provided by the API provider. The documentation should contain information on how to use the API, including its endpoints, parameters, and data formats. Take the time to carefully read through the documentation to ensure that you understand how to use the API effectively.

Choosing an Endpoint:

Once you have a good understanding of the API's documentation, you can choose an appropriate endpoint for your desired requests. Endpoints are URLs that represent specific functions of the API. For example, the randomuser.me API has endpoints for generating random users, retrieving user data, and more. The documentation should list all available endpoints and their respective parameters.

Send a Request:

After identifying the correct endpoint and parameters for your request, you can use HTTP (Hypertext Transfer Protocol) to build and send a request to the server's endpoint. You can use various programming languages and libraries to send HTTP requests.

Parse the API Response:

Once you've sent a request to the server, it will respond with data that needs to be parsed before it can be used in your application. In the case of the randomuser.me API, the response is sent in JSON (JavaScript Object Notation) format, a format that is easy for humans to read and write, and easy for machines to parse and generate.

Parse the JSON Response:

Use a JSON parse library in your programming language. The JSON data can then be accessed as a regular object, with each key representing a piece of data in the response.

Synchronizing these steps should be a basic intro of understanding of how to generate random user data using the Randomuser.me API services.

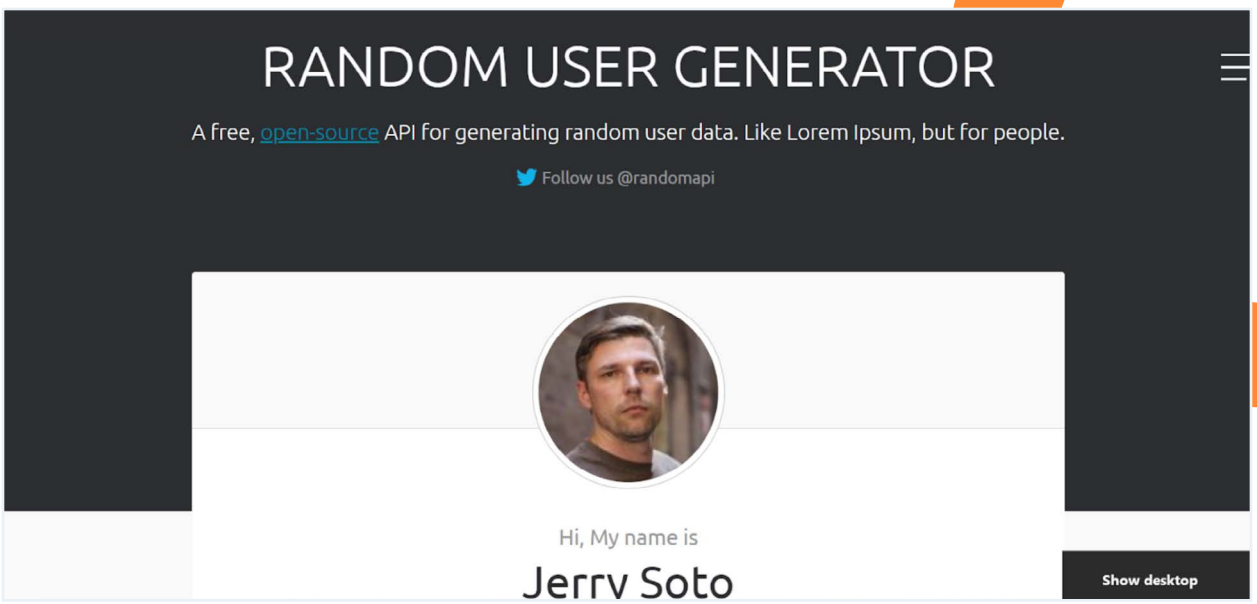


Image Source

Common Error Types

Web developers may experience a variety of issues when developing web applications. Here are some common errors and ways to address them.

400 Bad Request

Image Source

The 400 Bad Request error is an HTTP status code that indicates that the server could not interpret the request sent by the client due to invalid syntax. Handling this issue will be determined by your understanding of the error and being able to make the necessary adjustments. Careful examination of the client side code should conclude the issue of 400 Bad Request Error.

401 Unauthorized Error:

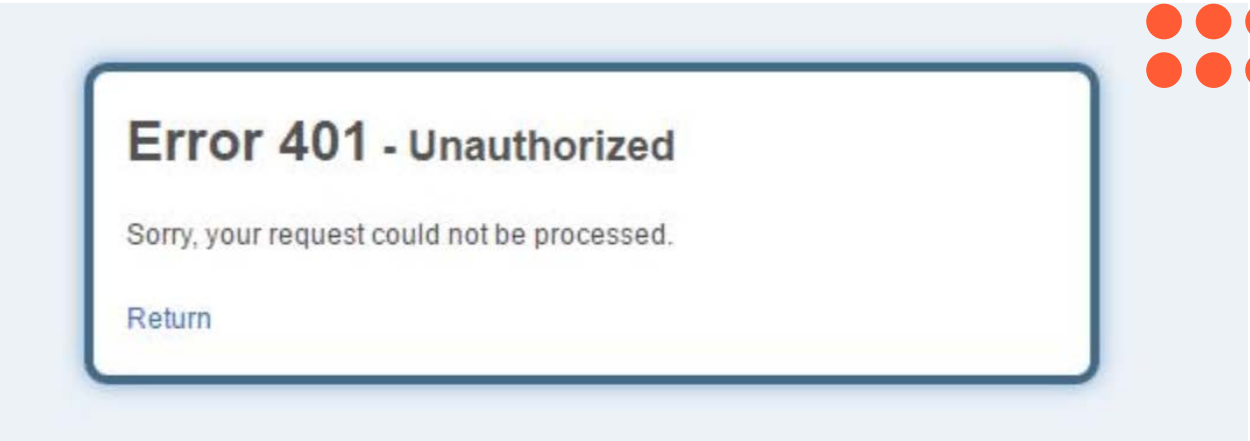


Image Source

The 401 Unauthorized error is an HTTP status code that indicates that authentication is required to access protected resources.

The most common reasons for receiving this error include missing or incorrect authentication credentials, such as an incorrect password or username, or not including the necessary credentials for accessing the server. Another common reason is that the authentication token or session ID has expired or been revoked, which results in the server revoking any requests made by the client.

To resolve the 401 unauthorized error, developers must assess the credentials they have and ensure that they have the necessary permission to access the requested resources. If the client does not have the required credentials, they may need to request additional authorization permission from server administrators.

403 Unauthorized Error:

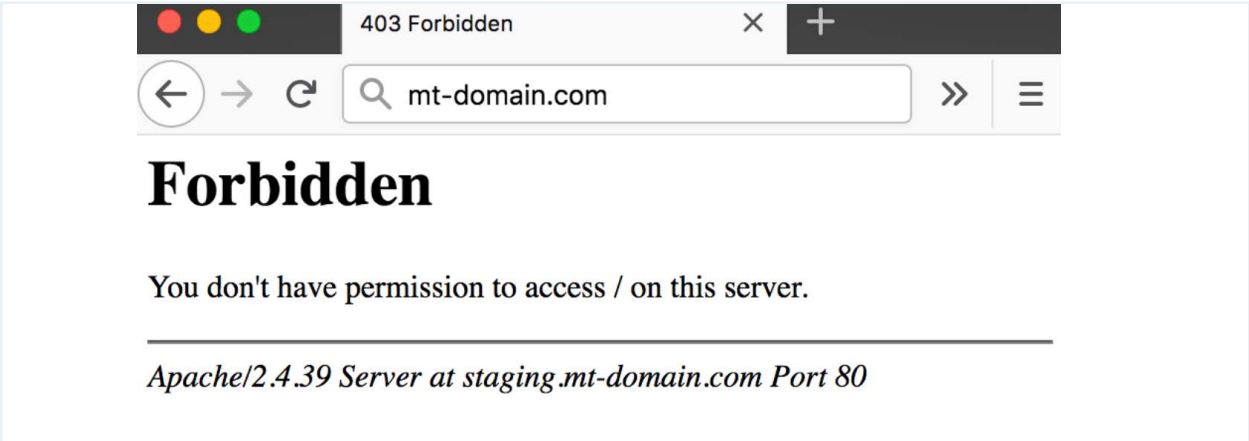


Image Source

The 403 forbidden error is displayed when a client — most likely a web browser — was able to make a connection with the server for requests, but the server is refusing to fulfill the request with the resources desired within the request.

Authorization and authentication could be the leading cause of this error — similar to the issues for 401 unauthorized error — but in this case, you are communicating with the server so the files requested may not be valid for your level of authorization. IP address blocking can also cause the 403 Forbidden error, as the server may block requests from certain IP addresses that it considers to be suspicious or malicious.

The 403 Forbidden error can also occur due to server misconfiguration or when the server is denying requests by default, resulting in the error code being returned.

When faced with a 403 error, developers should assess it as a lack of permission for accessing the requested resource. Resolving this issue typically involves dealing with the server's configurations or permissions for the desired resource.

404 Not Found Error:

The 404 Not Found error code indicates that although the client successfully connected to the server, the server was unable to locate the requested item. In other words, the server could not find the resource that the client was requesting.There are several causes of error code 404:

- Requested URL does not exist
- Requested URL has moved or deleted
- Temporary maintenance for the server
- Client may have followed broken link

Overall, a 404 error code indicates requested resource is not available. Using an alternative resource or reporting the issue to server administrators are the necessary steps to bypass this issue.

429 Too Many Requests Error

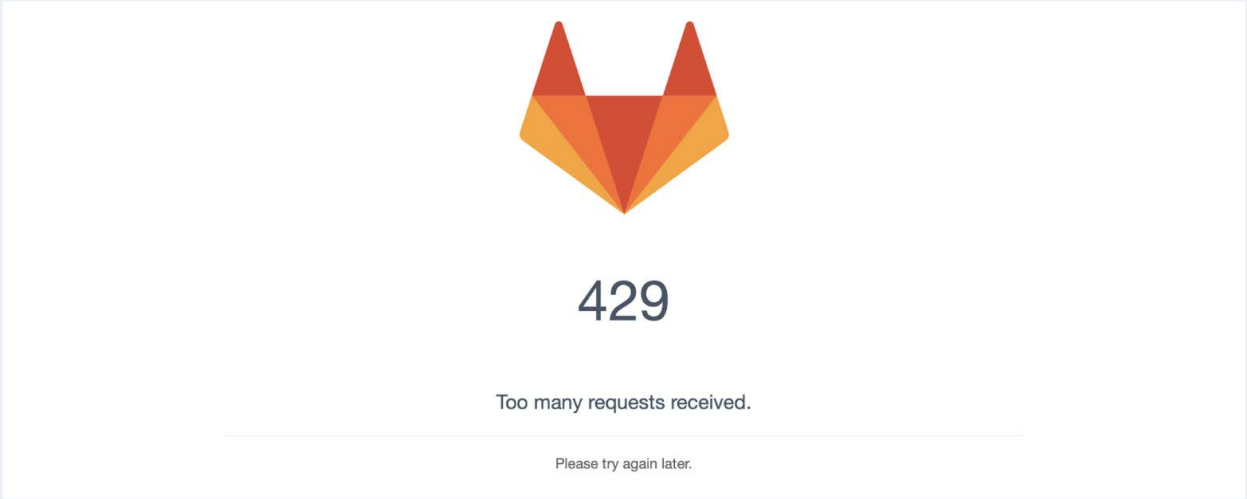
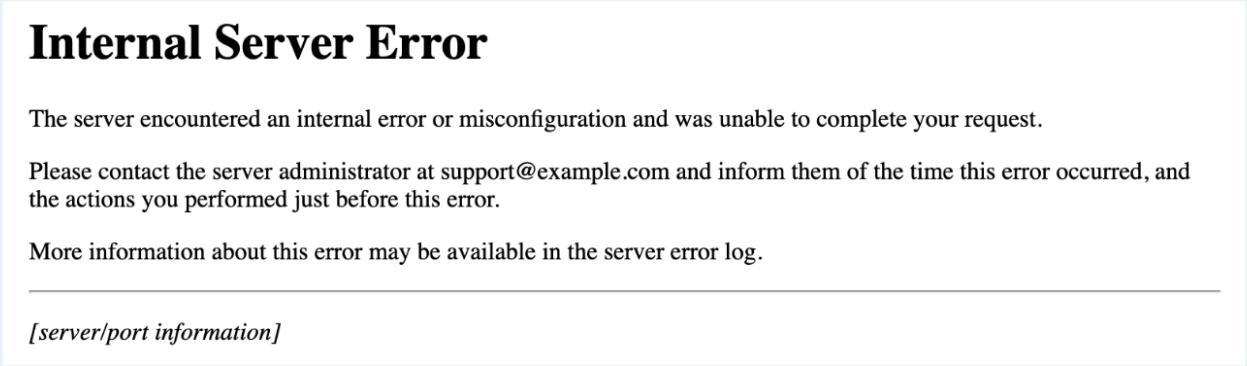


Image Source

Error code 429 too many requests indicates too many request attempts were made. It's also referred to as the rate limit exceeded. Consider the following to address the issue. Rate limiting occurs when a server receives an excessive number of requests from a client within a given period of time. This can result in the server rejecting or slowing down requests. Server overload, on the other hand, occurs when a server is unable to handle a high volume of requests. This could be due to the server settings not being customized to handle such a volume of requests.

To prevent intentional overloading of the server, the security services of the server may limit requests from certain IP addresses. This can help prevent an attack aimed at overwhelming the server. To overcome error code 429 too many requests, users should wait a specified period of time to continue making additional requests.

500 Internal Server



Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator at support@example.com and inform them of the time this error occurred, and the actions you performed just before this error.

More information about this error may be available in the server error log.

[server/port information]

Image Source

500 internal server error code is standardized for internal error. These status codes represent that the server encountered an unexpected issue that prevented it from fulfilling the users request. Causes might include:

- **Server Misconfiguration.**
  The server could have a syntax error within the server's configuration.
- **Database Errors.**
  If there's connection issues with the databasel the server could return a 500 error code.
- **Software or Application Issues:**
  Code errors within the software or application running within the server will trigger 500 error codes.
- **Resource Exhaustion:**
  This will undoubtedly return a 500 error code if the server is reaching its memory capacity.
- **Security-Related Issues::**
  These could cause an intentional response of a 500 error code, preventing a hack attempt or denial of service attack.

Resolving a 500 error code could require advanced knowledge of error status codes, maintaining regular tests on the API errors is a great way to provide feedback for consumers receiving these status of error codes.

What are REST APIs?

A REST (Representational State Transfer) API is a form of web service that enables HTTP protocols to push and pull data between multiple software systems. Introduced in the year 2000 by Roy Fielding, these were a significant show of what APIs really could do. Building on the structure of APIs, RESTful APIs have some significant differences between traditional APIs. RESTful APIs are a type of web service that cohere to these principles and are widely used because they are simple, flexible, and easy to integrate with other systems.

- The REST API is stateless. All requests include all the necessary data to complete the request. Traditional APIs require session information to complete the request, but REST APIs allow for servers to be maintained easier and reduce server load.
- REST API architecture is designed for clients to receive data from the server, allowing for distinct separation between client and server responsibilities and increasing the chances for scalability of the system.
- REST APIs introduced caching, allowing for the reuse of already requested data. A basic API does not have caching abilities.

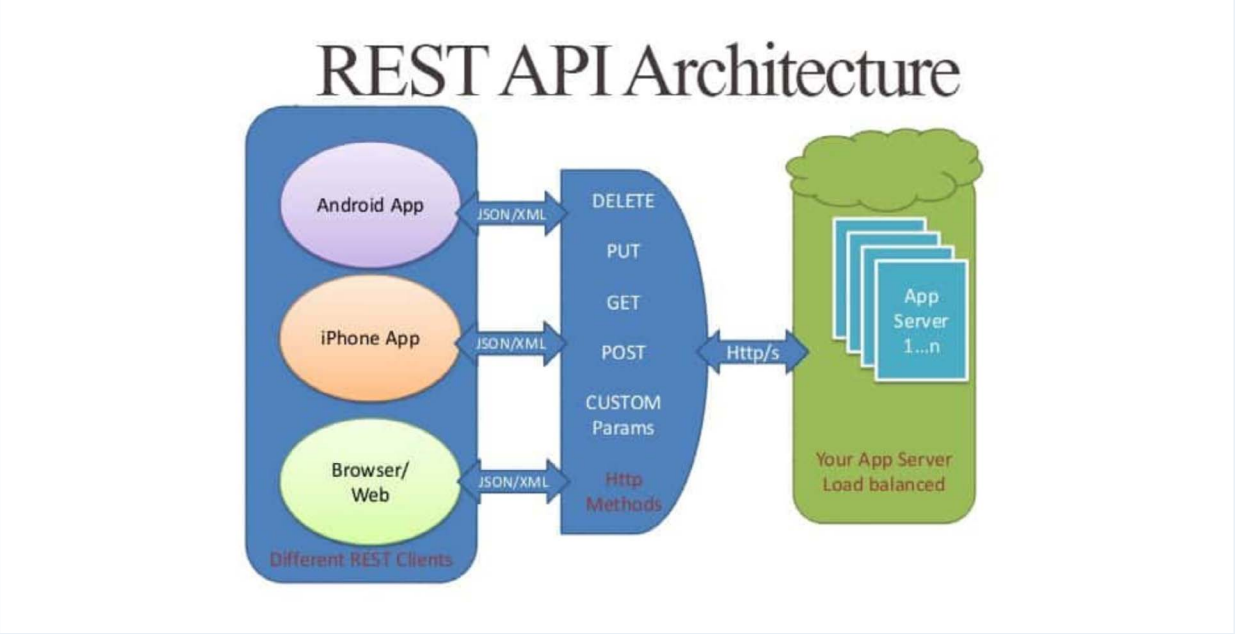
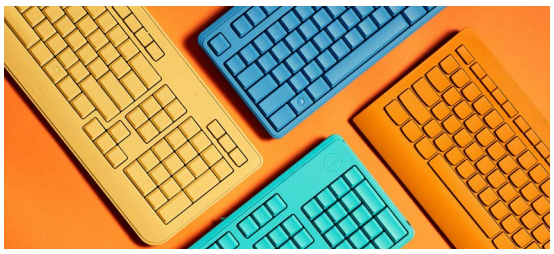


Image Source

REST APIs follow certain principles that basic APIs do not have, which allows for gateways to flexible and scalable solutions for web services. Rest APIs increase the ease of use and efficiency of performance.



Security and Authentication

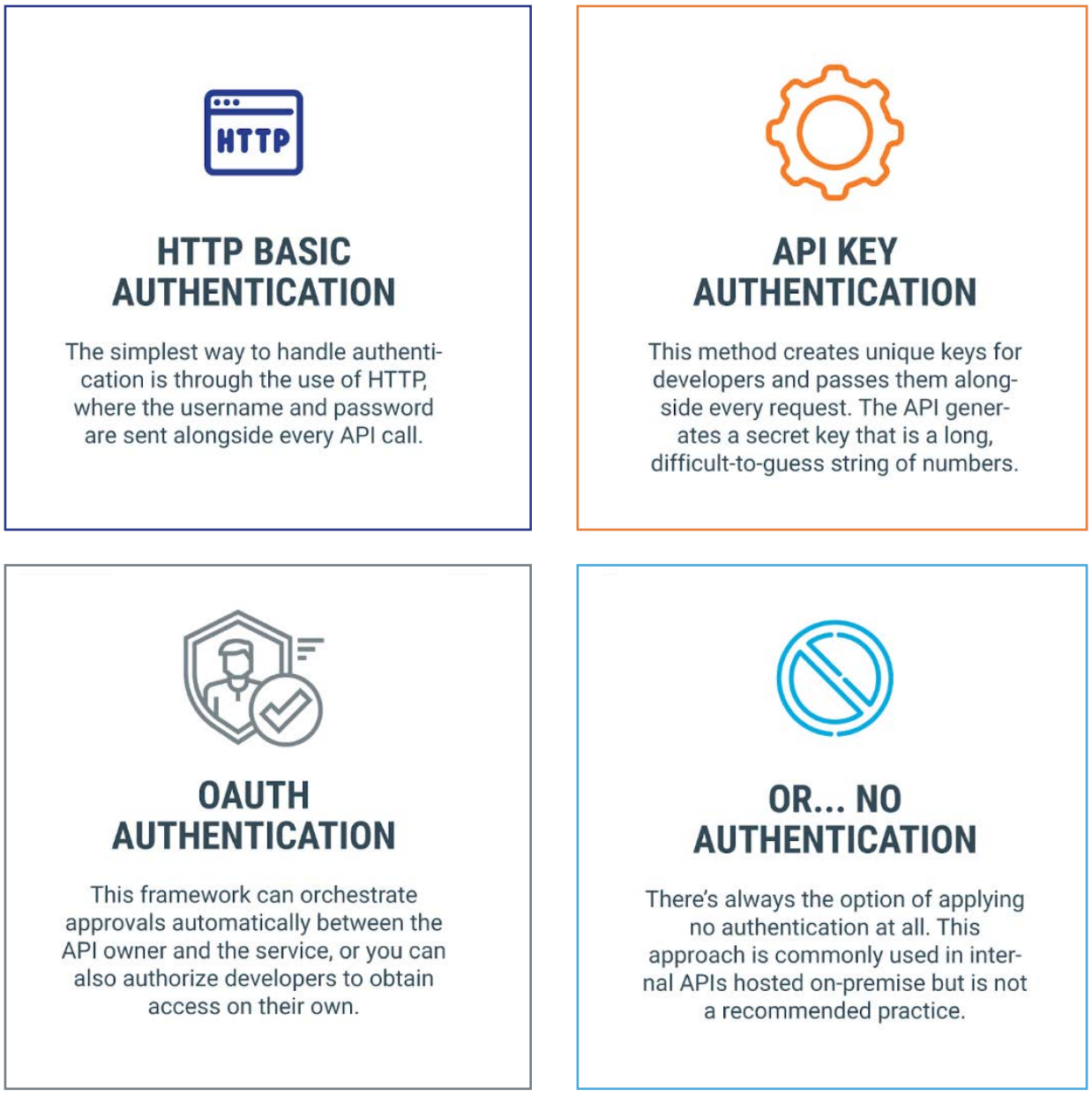


Image Source

While APIs are extremely useful, they can also pose security risks if they are not properly secured and maintained. APIs are used to retrieve sensitive client and server data, certain steps should be followed when securing and authenticating APIs.

- Authentication of an API has several different methods to verify authenticated clients, such as API keys, OAuth, and JSON web tokens. API keys are the easiest method when it comes to authentication, but does not include the extra security features of OAuth and JSON web tokens.

- Authorization is the process in which many permissions are allowed to the authenticated client and which endpoints can be accessed with these levels of permissions.

- Encryption of sensitive data should be used when transporting sensitive data. HTTPS is the most common way of encrypting API requests.

- Rate limiting is the process of allowing a set amount of transactions and requests to occur within a set limit of time to prevent misuse of an API and ensure the API remains responsive.

- Monitoring and logging of all potential security risks, such as suspicious behavior and unauthorized access attempts, is essential to the security of the API.

Following these security and authentication methods and responding in a timely manner can ensure developers that they can deliver a reliable safe service for their API.

Closing

Hopefully this was a comprehensive read on how to understand and begin to implement APIs into your applications. Now you know how critical APIs are to the future of software development and how utilizing APIs effectively can help developers from small phone applications to intricate banking financial systems that the world depends on. APIs remain effective and continue to bring better solutions to future software development.

Subscribe to HubSpot's Website Blog