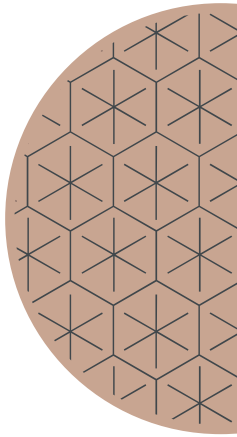
Utilización de mecanismos de comunicación asíncrona

Programación asíncrona

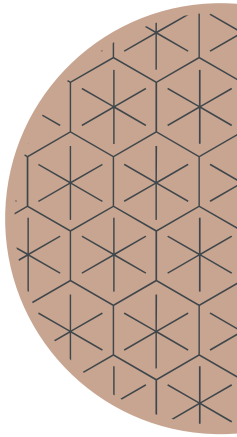
La programación asíncrona es una técnica que permite a tu programa iniciar una tarea potencialmente de larga duración y seguir siendo capaz de responder a otros eventos mientras esa tarea se ejecuta, en lugar de tener que esperar hasta que esa tarea haya terminado. Una vez que la tarea ha finalizado, el programa recibe el resultado.



Programación asíncrona

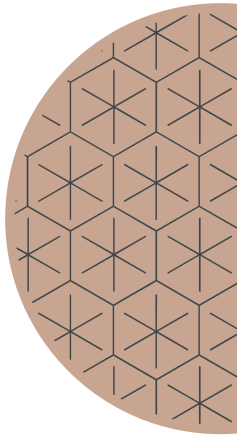
Muchas funciones proporcionadas por los navegadores, especialmente las más interesantes, pueden potencialmente tardar mucho tiempo, y por lo tanto, son asíncronas. Por ejemplo:

- Hacer peticiones HTTP usando fetch()
- Acceder a la cámara o micrófono de un usuario utilizando getUserMedia()
- Pedir a un usuario que seleccione archivos usando showOpenFilePicker()



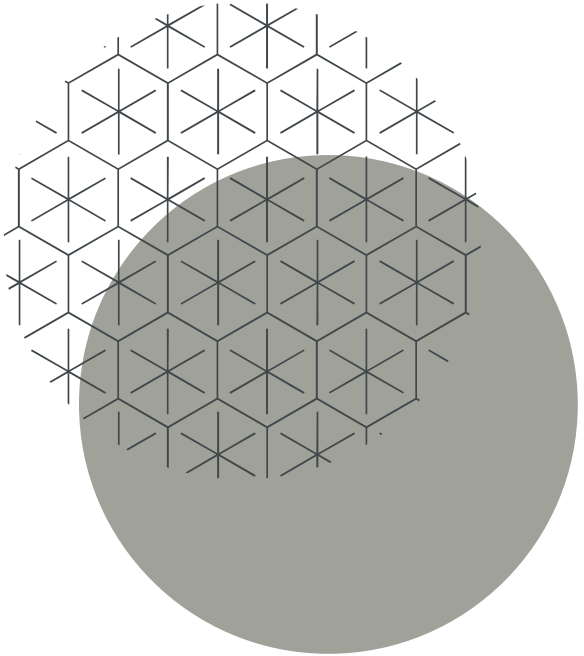
Event handlers

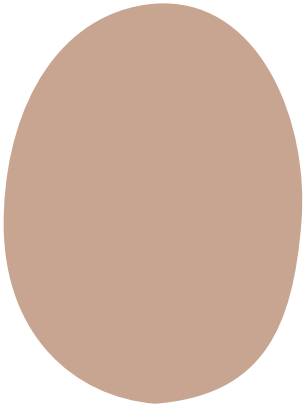
La descripción que acabamos de ver de las funciones asíncronas podría recordarte a los manejadores de eventos, y si es así, estarías en lo cierto.



Los manejadores de eventos son en realidad una forma de programación asíncrona: proporcionas una función (el manejador de eventos) que será llamada, no de inmediato, sino cuando ocurra el evento.

Ajax





Ajax

AJAX = JavaScript asíncrono y XML.

AJAX no es un lenguaje de programación.

AJAX sólo utiliza una combinación de:

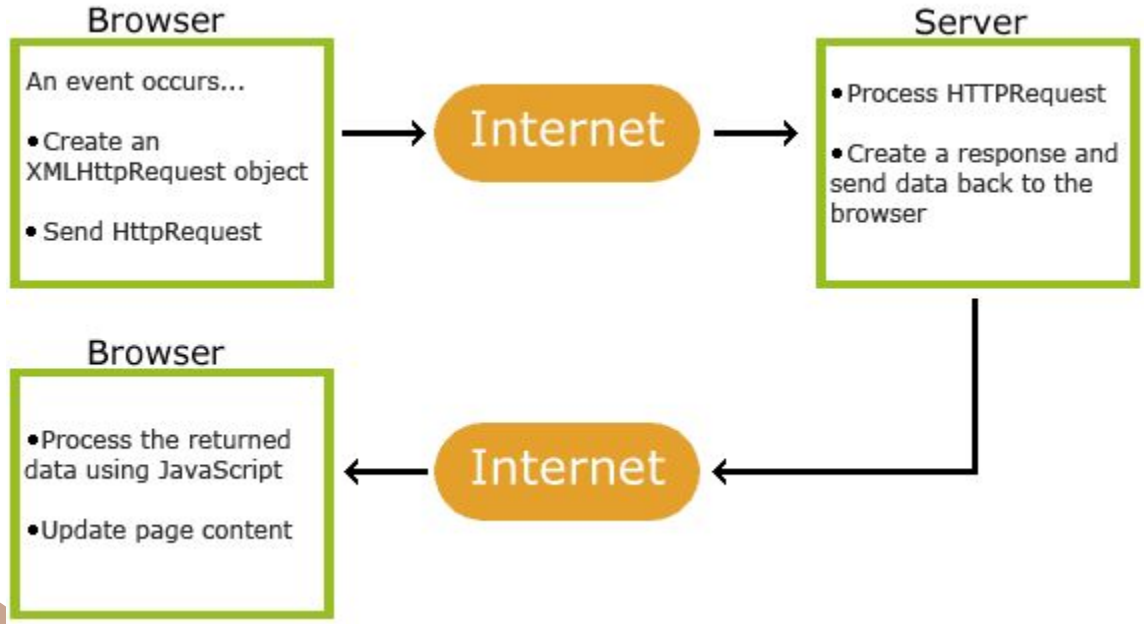
- Un objeto XMLHttpRequest integrado en el navegador (para solicitar datos a un servidor web)
- JavaScript y HTML DOM (para mostrar o utilizar los datos)

Ajax

AJAX es un nombre engañoso. Las aplicaciones AJAX pueden utilizar XML para transportar datos, pero es igualmente habitual transportar datos como texto plano o texto JSON.

AJAX permite actualizar páginas web de forma asíncrona mediante el intercambio de datos con un servidor web entre bastidores. Esto significa que es posible actualizar partes de una página web sin recargarla entera.

Ajax


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Browser
• Process the returned data using JavaScript
• Update page content
Internet

API

¿Qué es un API?

API es el acrónimo de Application Programming Interface (interfaz de programación de aplicaciones), que es un intermediario de software que permite que dos aplicaciones se comuniquen entre sí.

Cada vez que usas una aplicación como Instagram, envías un mensaje instantáneo o consultas el tiempo en tu teléfono, estás usando una API.

Ejemplo de API

Cuando usas una aplicación en tu teléfono móvil, la aplicación se conecta a Internet y **envía datos** a un servidor.

El **servidor** recupera esos datos, los **interpreta**, **realiza** las **acciones** necesarias y los **devuelve** a su teléfono.

A continuación, la **aplicación interpreta** esos datos y te **presenta** la **información** que querías de forma legible. Esto es lo que es una API.

Capa de seguridad del uso de APIs

Los datos de tu teléfono nunca están totalmente expuestos al servidor, y del mismo modo el servidor nunca está totalmente expuesto a tu teléfono. En su lugar, cada uno se comunica con pequeños paquetes de datos, compartiendo sólo lo necesario, como cuando se pide comida para llevar. Le dices al restaurante lo que quieres comer, ellos te dicen lo que necesitan a cambio y, al final, recibes tu comida.

La API moderna

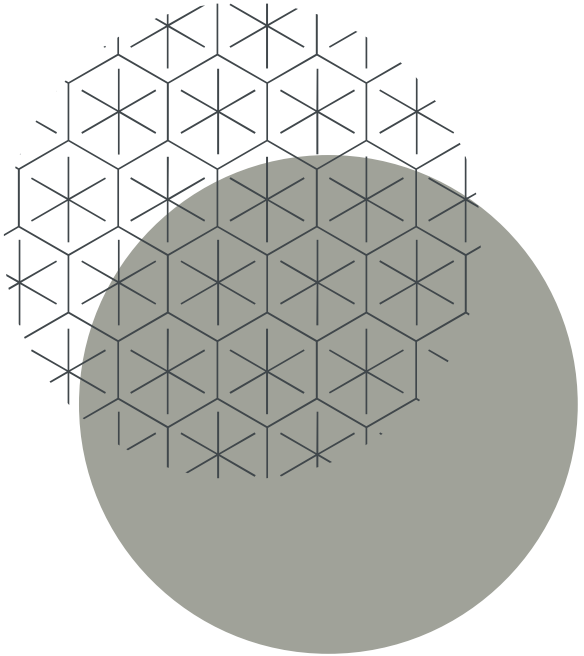
A lo largo de los años, lo que es una "API" ha descrito a menudo cualquier tipo de interfaz de conectividad genérica con una aplicación. Sin embargo, más recientemente, las API modernas han adquirido algunas características que las hacen extraordinariamente valiosas y útiles:

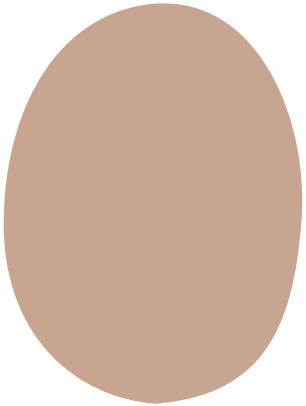
1. Las API modernas se adhieren a los estándares (normalmente HTTP y REST), que son fáciles de desarrollar, de acceder y de entender en general.

La API moderna

1. Se tratan más como productos que como código. Están diseñadas para ser consumidas por audiencias específicas.
2. Al estar mucho más estandarizados, tienen una disciplina mucho más fuerte para la seguridad y la gobernanza, así como supervisados y gestionados para el rendimiento y la escala
3. La API moderna tiene su propio ciclo de vida de desarrollo de software de diseño, pruebas, construcción, gestión y versionado. Además, las API modernas están bien documentadas para su consumo y versionado.

API REST





¿Qué es un API REST?

Una API REST (también conocida como API RESTful) es una interfaz de programación de aplicaciones que se ajusta a las restricciones del estilo arquitectónico REST y permite la interacción con servicios web RESTful. REST significa transferencia de estado representacional y fue creado por el informático Roy Fielding.

¿Qué es un API REST?

REST es un conjunto de restricciones arquitectónicas, no un protocolo ni un estándar. Los desarrolladores de APIs pueden implementar REST de diversas maneras.

Cuando se realiza una solicitud de cliente a través de una API RESTful, ésta transfiere una representación del estado del recurso al solicitante o al punto final. Esta información, o representación, se entrega en uno de varios formatos a través de HTTP: JSON (Javascript Object Notation), XML, Python, PHP o texto plano.

¿Qué es el formato JSON?

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos. Es fácil de leer y escribir para los humanos. Es fácil de analizar y generar para las máquinas.

JSON es un formato de texto completamente agnóstico del lenguaje, pero utiliza convenciones que resultan familiares a los programadores de la familia de lenguajes C, como C, C++, C#, Java, JavaScript, Perl, Python y muchos otros. Estas propiedades hacen de JSON un lenguaje de intercambio de datos ideal.

Ejemplo de JSON

```json
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

¿Qué es un API REST?

Las cabeceras y los parámetros también son importantes en los métodos HTTP de una solicitud HTTP de la API RESTful, ya que contienen información importante de identificación en cuanto a los metadatos de la solicitud, la autorización, el identificador uniforme de recursos (URI), el almacenamiento en caché, las cookies y más.

Hay cabeceras de solicitud y cabeceras de respuesta, cada una con su propia información de conexión HTTP y códigos de estado.

¿Qué es un API REST?

Para que una API se considere RESTful, debe ajustarse a estos criterios:

1. Una arquitectura cliente-servidor formada por clientes, servidores y recursos, con peticiones gestionadas a través de HTTP.
2. Comunicación cliente-servidor sin estado, lo que significa que no se almacena información del cliente entre las solicitudes de obtención y cada solicitud es independiente e inconexa.
3. Datos almacenables en caché que agilizan las interacciones cliente-servidor.

¿Qué es un API REST?

1. Una interfaz uniforme entre los componentes para que la información se transfiera de forma estándar.

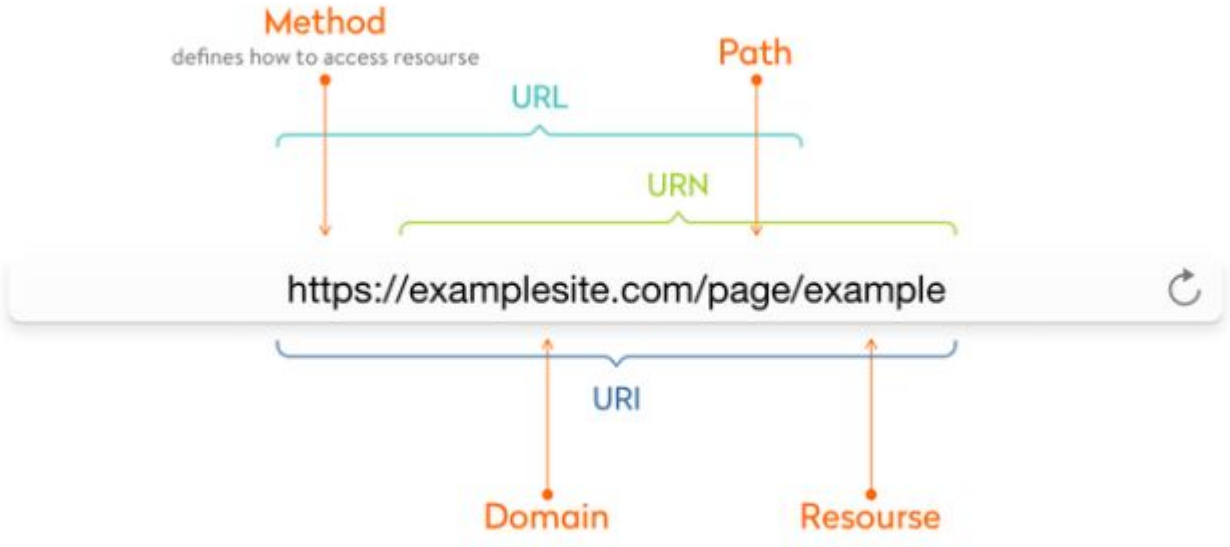
2. Un sistema de capas que organiza en jerarquías, invisibles para el cliente, cada tipo de servidor (los responsables de la seguridad, el equilibrio de carga, etc.) que interviene en la recuperación de la información solicitada.

Cómo hacer un API REST

¿Cómo hacer API un API REST?

El principio clave de REST es dividir su API en recursos lógicos. Un identificador de recursos uniforme, o URI, es una secuencia de símbolos que identifica un recurso, la estructura y sintaxis actual de los URI está definida por el estándar RFC 3986.

¿Cómo hacer API un API REST?



[Components of URI]

¿Cómo hacer API un API REST?

La URI debe comunicar claramente el modelo de recursos de la API. Reglas para diseñar URIs

1. Utilice guiones pero no guiones bajos para que su URI sea legible.
2. Utilice las minúsculas siempre que sea posible.
3. En general, cada recurso de su API tendrá al menos un URI. Por eso, cada URI debe describir adecuadamente el recurso y seguir una estructura predecible estructura jerárquica para mejorar la usabilidad.

Endpoints

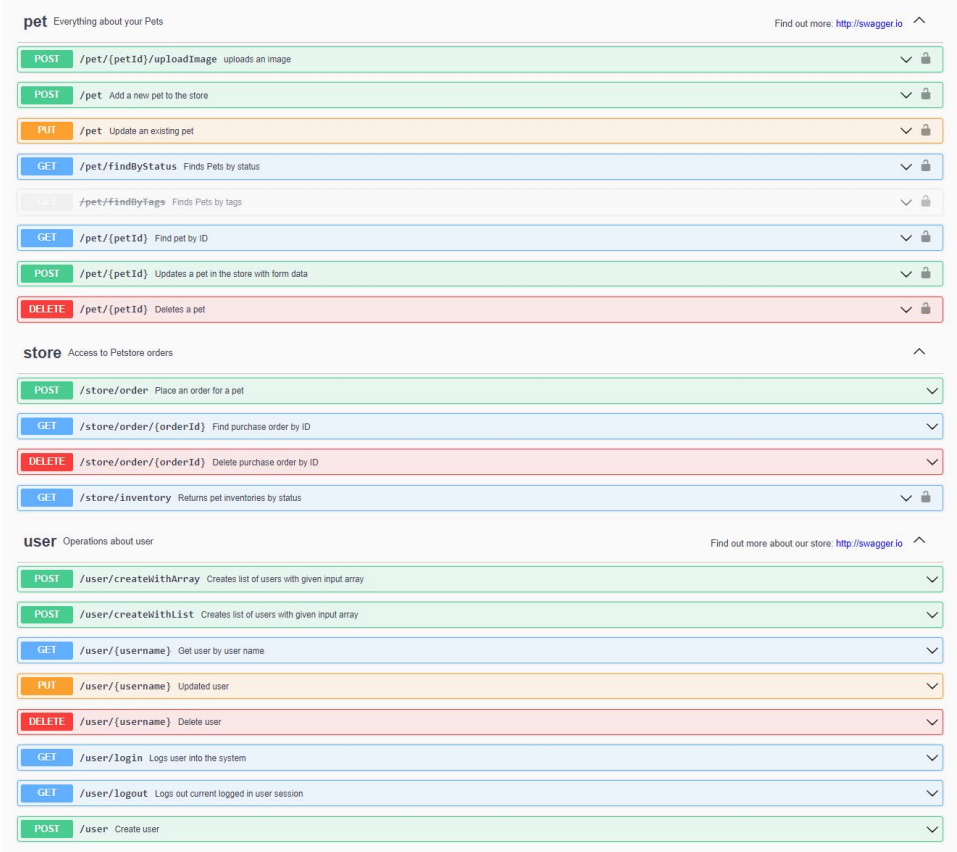
Los endpoints especifican dónde se encuentran los recursos y cómo se puede acceder a ellos mediante software de terceros. Normalmente, se accede a ellos a través de una URI a la que se envían peticiones HTTP y de la que se espera una respuesta.

1. **GET**: recupera recursos del servidor utilizando una URI determinada
2. **POST**: crea recursos en el servidor.
3. **PUT**: actualizar un recurso existente con el contenido nuevo.
4. **DELETE**: eliminar el recurso o recursos.

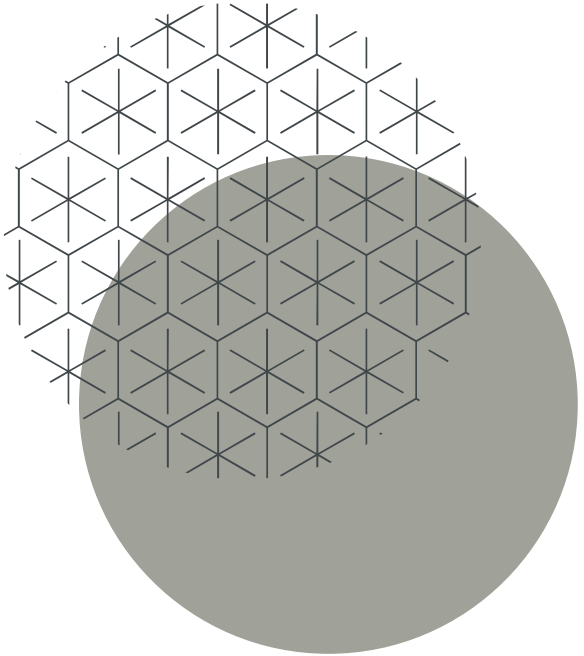
Endpoints

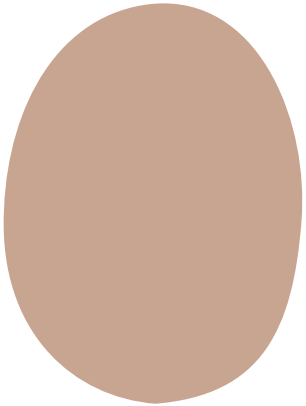
Ejemplo de API REST:

https://petstore.swagger.io/



Controller





¿Qué es un controller?

En las aplicaciones ASP.NET, el controlador es responsable de aceptar entradas, tomar u orquestar operaciones y devolver una respuesta.

Se trata de un marco de trabajo con todas las funciones, que ofrece una canalización extensible a través de filtros, validación y vinculación de modelos integrada, comportamientos basados en convenciones y declaraciones y mucho más. Para muchos, es una solución todo en uno para construir aplicaciones HTTP modernas.

REST API response codes

1. 200 – OK
2. 201 – Created
3. 204 – No content
4. 400 – Bad request
5. 401 – Unauthorized
6. 403 – Forbidden
7. 404 – Not found
8. 405 – Method not allowed
9. 409 – Conflict
10. 415 – Unsupported Media Type
11. 500 – Internal Server Error

FIN