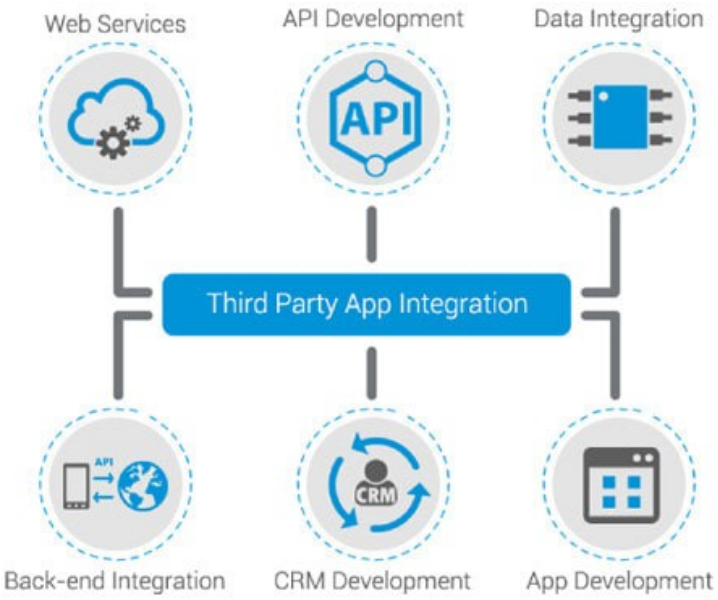
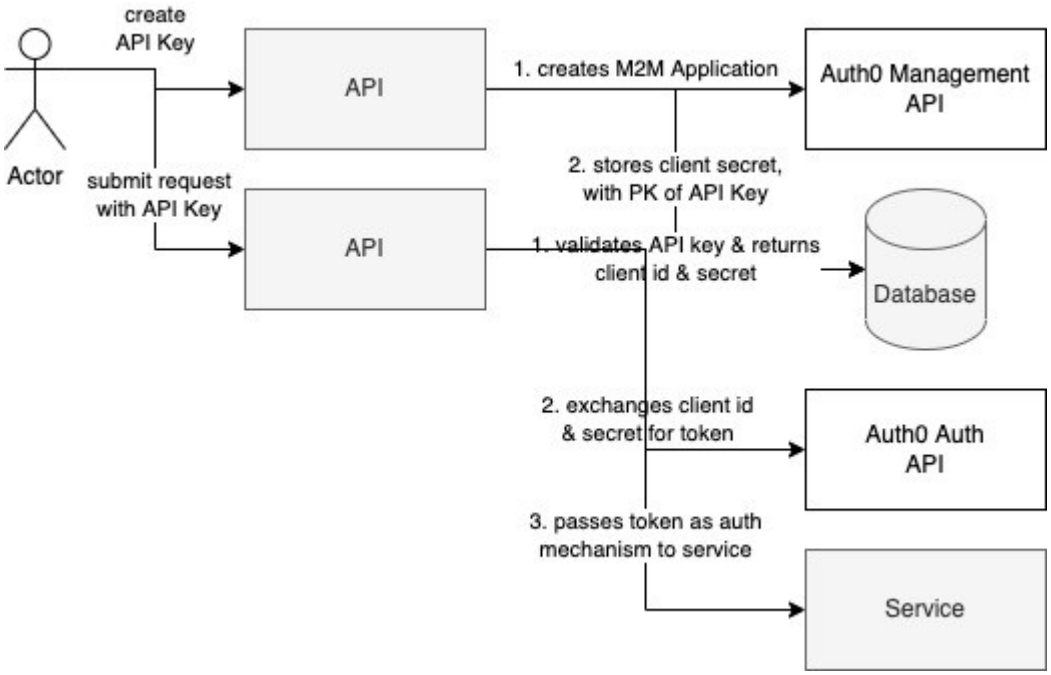
Api

Un API (Interface de Programación de Aplicaciones) es un sistema de comunicación entre lenguajes o equipos. Nos ayudará a recibir datos y generar peticiones normalmente en JSON, que es actualmente el idioma universal para comunicar con cualquier otro lenguaje.



Autentificación

La autenticación API es un proceso mediante el cual una aplicación verifica la identidad de un usuario o de otro sistema. Se utiliza comúnmente en el desarrollo de aplicaciones web y móviles para garantizar la seguridad y la integridad de los datos. Existen varios métodos de autenticación API, como el uso de *tokens* de acceso, claves de API y protocolos como *OAuth*. Estos métodos permiten a las aplicaciones comunicarse de forma segura con servidores externos y acceder a recursos protegidos. La autenticación API es esencial para proteger la información confidencial y garantizar que solo usuarios autorizados puedan acceder a los datos.



Servir un JSON como un API

Veamos un ejemplo sencillo donde devolvemos un JSON a partir de un *array*.

```php
<?php
// Información
$relojes = [
        [
          'marca' => 'Marea',
          'origen' => 'Spain'
        ],
        [
          'marca' => 'Rolex',
          'origen' => 'Suiza'
        ],
        [
          'marca' => 'Omega',
          'origen' => 'Suiza'
        ],
        [
          'marca' => 'Casio',
          'origen' => 'Japón'
        ]
];

header('Content-Type: application/json');       // cabecera con el tipo de contenido
echo json_encode($relojes);                     // convierte a json y lo muestra por pantalla
```

...generando un contenido ideal para alimentar a cualquier otros lenguaje de programación.

```
[
 {
  "marca": "Marea",
  "origen": "Spain"
 },
 {
  "marca": "Rolex",
  "origen": "Suiza"
 },
 {
  "marca": "Omega",
  "origen": "Suiza"
 },
 {
  "marca": "Casio",
  "origen": "Japón"
 }
]
```

Recoger un JSON de un API

Para obtener el JSON debemos utilizando un cliente HTTP que gestione las cabeceras, verbos y demás elementos técnicos. En PHP está altamente estandarizado con la invocación a **curl**.

Cada API puede proporcionar sus propias especificaciones a la hora de conectar con un servicio API o webservice, ejemplos de API son e-commerce, información metereológica, servicios de envío de mercancía, información sobre eventos deportivos, es decir, cualquier información que se desee compartir con otras aplicaciones.

A continuación puede observar una función de ejemplo que puede ser de ayuda:

```
<?php
/**
 * Obtiene un JSON de una url
 * @param {string} $url
 * @param {string} $method - Por defecto GET.
 * @param {array} $body - Array con elementos a enviar. Por defecto [].
 * @return {array}
 */

function getJSON($url, $method, $body) {
   $curl = curl_init();
   $json = json_encode($body);

   curl_setopt($curl, CURLOPT_URL, $url);
   curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
   curl_setopt($curl, CURLOPT_CUSTOMREQUEST, $method);
   curl_setopt($curl, CURLOPT_POSTFIELDS, $json);
   curl_setopt($curl, CURLOPT_HTTPHEADER, array(
      'Content-Type: application/json',
      'Content-Length: ' . strlen($json))
   );

   $result = curl_exec($curl);
   curl_close($curl);

   return json_decode($result, $assoc = true) ;
}
$data =  getJSON('https://iestacio/api/v3/profesores', 'GET', $elementos);
```

Aquí puede ver otro ejemplo de llamada:

```
$data = getJSON('https://iestacio/api/v3/alumnos', 'POST', array('curso' => 'DAM'));
```

Estructura de rutas

Antes de realizar tu API debes dar una estructura lógica para que los desarrolladores asimilen rápidamente como hacer llamadas. En la siguiente tabla puedes ver un ejemplo de una supuesta biblioteca con rutas usando un formato ampliamente extendido llamado *API REST*.

Veamos por ejemplo como podría ser una estructura de rutas para una biblioteca:

Ruta	Método	Funcionalidad
/signup	POST	Registro
/auth/login	POST	Inicio de sesión
/auth/logout	GET	Cerrar sesión
/biblioteca	GET	Lista todas las bibliotecas
/biblioteca	POST	Crea una nueva biblioteca
/biblioteca/45	GET	Obtiene la biblioteca
/biblioteca/45	PUT	Actualiza la biblioteca
/biblioteca/45	DELETE	Borra la biblioteca
/biblioteca/45/libros	GET	Lista todos los libros de la biblioteca
/biblioteca/45/libros/21	GET	Obtiene el libro de la biblioteca
/biblioteca/45/libros/21	PUT	Actualiza el libro de la biblioteca
/biblioteca/45/libros/21	DELETE	Borra el libro de la biblioteca

Cabeceras interesantes

Enviar contenido JSON:

```
<?php
header('Content-Type: application/json');
```

Habilitar CORS, para indicar que dominios o recursos externos pueden acceder a sus datos:

```
<?php
header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
header('Access-Control-Allow-Credentials: true');
```

Indicar caché

```
<?php
header('Access-Control-Max-Age: 86400');      Ejemplo de un día en segundos.
```

Métodos disponibles

```
<?php
header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS');
```

Control de método

En algún momento habrá que vigilar que método estamos recibiendo (GET, POST, PUT, DELETE, etc). Y en caso contrario devolver un código de error apropiado.

```
<?php
// Filtramos por el método POST
if ($_SERVER['REQUEST_METHOD'] == 'POST') {

   // Cabecera que indica el tipo de contenido a servir
   header('Content-Type: application/json');
   // Indicamos en la cabecera que tipo de métodos están disponibles
   header("Access-Control-Allow-Methods: POST, OPTIONS");
   // Indicamos el código que devolveremos. 200 en caso que todo sea correcto.
   http_response_code(200);

   // Imprimimos un JSON de respuesta con el código 200
   echo json_encode([
      'status' => 'ok'
   ]);

} else {

   // Indicamos el código que devolveremos. 405: método no permitido
   http_response_code(405);

   // Imprimimos un JSON de respuesta.
   echo json_encode([
      'status' => 'ok'
   ]);

}
```

En el ejemplo solo devolvemos el estatus de la conexión, pero podríamos añadir también los relojes del primer ejemplo:

```
   // Imprimimos un JSON de respuesta con el código 200 y los datos de relojes
   echo json_encode([
      'status' => 'ok',
      ‘data’ => $relojes,
   ]);
```