

HTML 5: MARKUP & APIS

Marc
Genís
Carla
Ton

HTML 5

Qué es HTML5?

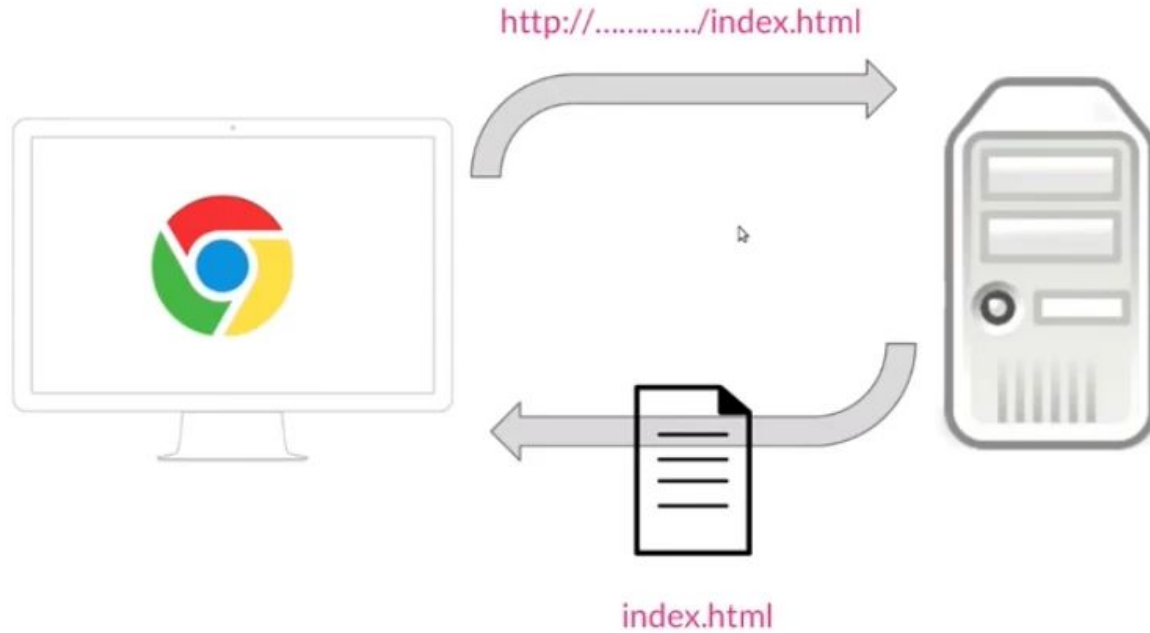
HyperText

Markup

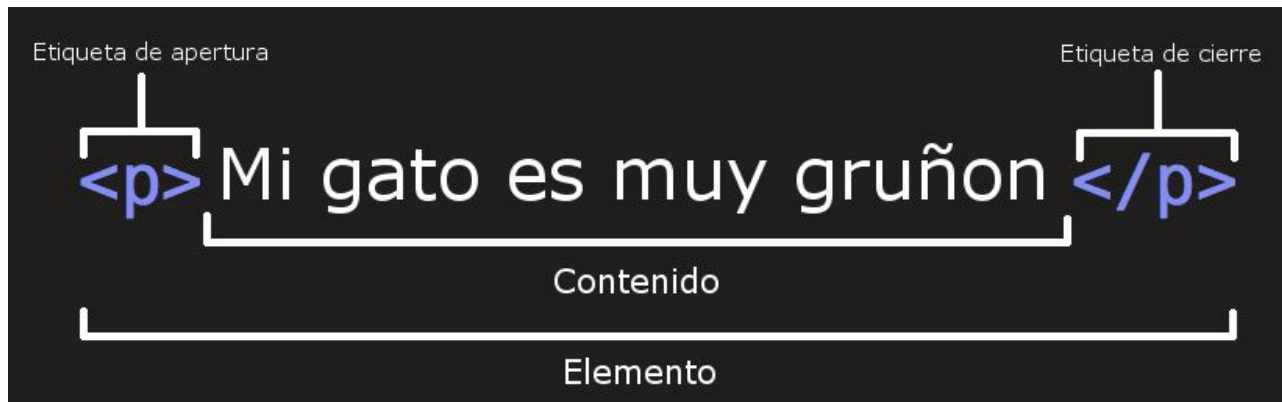
Language



Funcionamiento del HTML5



Anatomía de un elemento en HTML5



Estructura básica de una página HTML

```
<!doctype html>
<html>

<head>
  <head>
    <meta charset="utf-8"/>
    <title>Título de la web</title>
  </head>

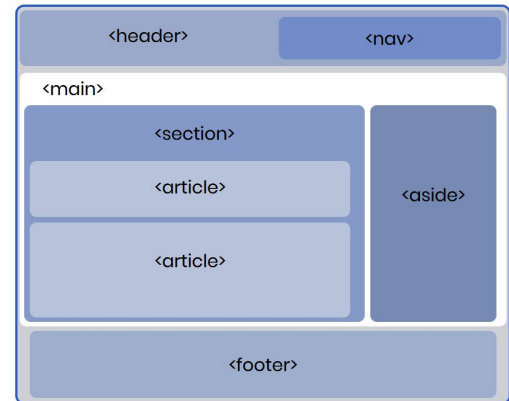
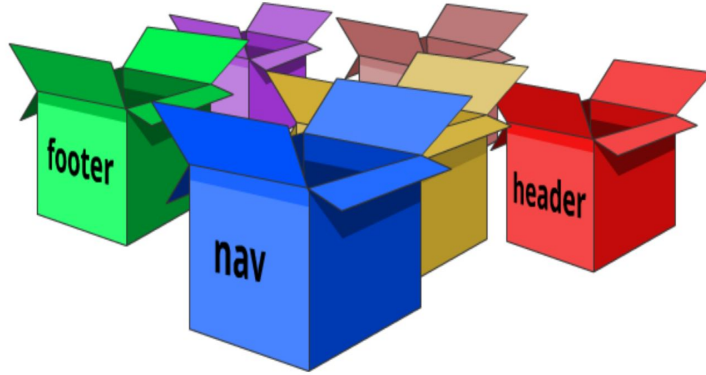
  <body>
    <body>
      Contenido de la web
    </body>

</html>
```

Qué contiene el head?

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Título de la página</title>
<meta name="Description" content="Descripción">
<link rel="canonical" href="Dirección URL de la página">
</head>
```

Qué contiene el body?



Otros elementos

`
`

Salto de línea

`<p>`

Párrafos

`<h1>`, `<h2>`, `<h3>`, `<h4>`

Encabezados

``

Enlaces

``

Imágenes

APIs

Tipos de API - Application Programming Interface

Local APIs

La API original, creada para proporcionar servicios de sistema operativo o middleware a los programas de aplicación

Web APIs

Diseñado para representar recursos ampliamente utilizados como páginas HTML y se accede mediante un protocolo HTML simple.
A menudo llamadas APIs REST vs APIs RESTFUL

Programa APIs

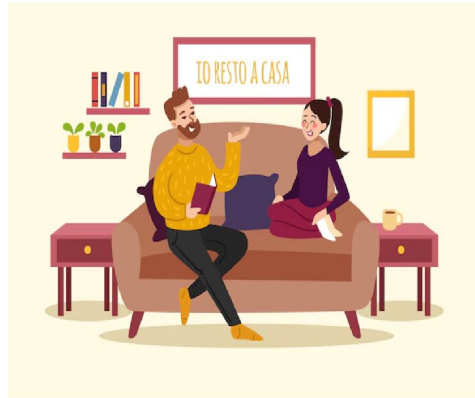
Basado en la tecnología RCP que hace que un componente de programa remoto parezca ser local al resto del software

¿Para qué sirve una API?

Servicio local



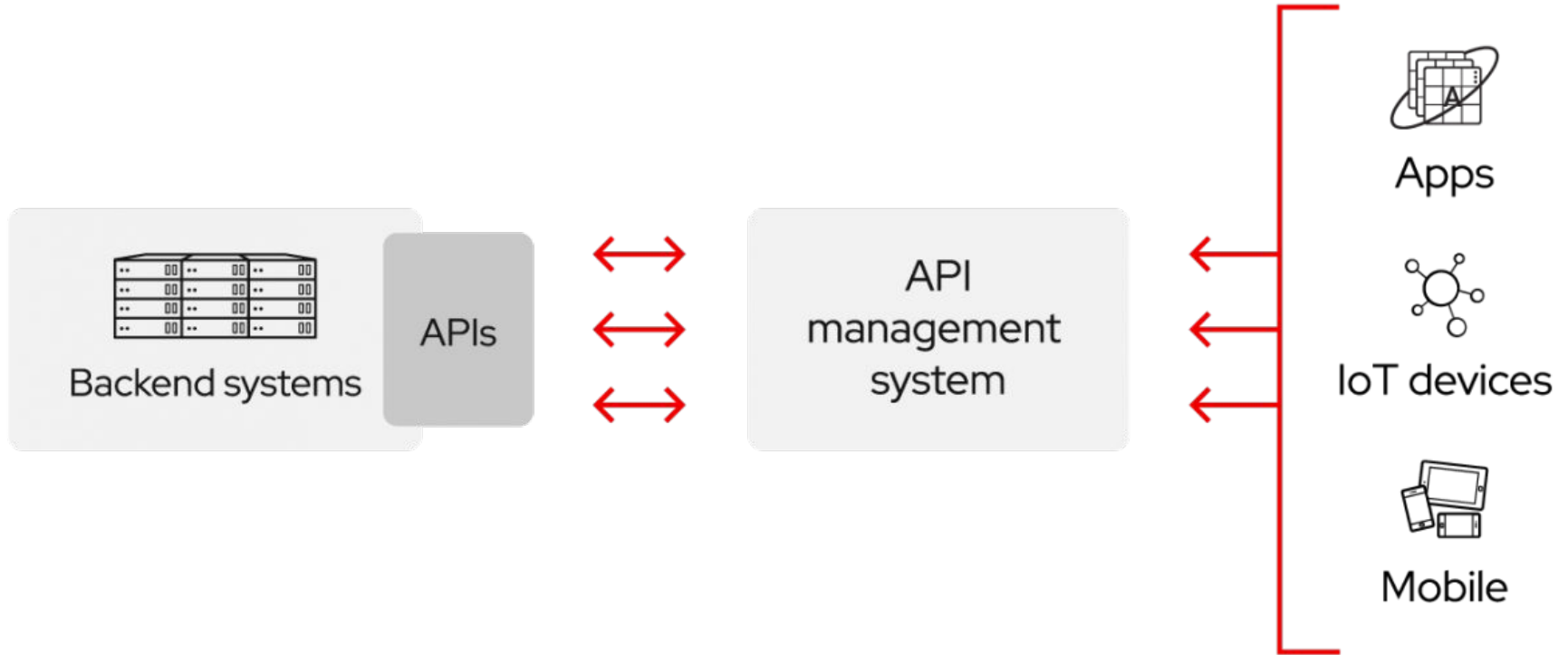
© CanStockPhoto.com - csp83586709



Servicio externo



¿Qué es una API?

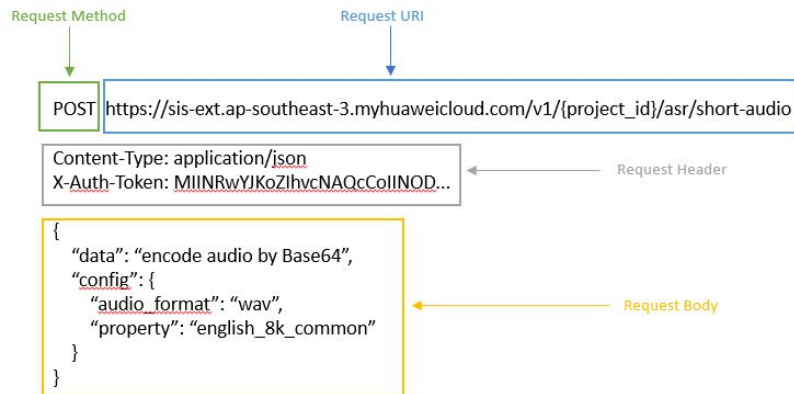


RESTful APIs - Representational State Transfer

- Arquitectura cliente-servidor través de HTTP.

Métodos:

- GET
- POST
- DELETE
- PUT



- Recurso: todo dentro de una API RESTful debe ser un recurso.
- URI: los recursos en REST siempre se manipulan a partir de la URI
- Acción: todas las peticiones a tu API RESTful deben estar asociadas a uno de los verbos de HTTP: GET para obtener un recurso, POST para escribir un recurso, PUT para modificar un recurso y DELETE para borrarlo.

Parámetros en una llamada a API

- **Path / endpoint**

GET https://prod-tenant1.bmc.com/ims/api/v1/roles/{id}

Method Scheme BMC Helix Portal URL Service Path Endpoint Path

The diagram shows the URL 'GET https://prod-tenant1.bmc.com/ims/api/v1/roles/{id}' with five red vertical lines pointing to its parts: 'GET' (Method), 'https' (Scheme), 'prod-tenant1.bmc.com' (BMC Helix Portal URL), '/ims/api/v1/roles' (Service Path), and '{id}' (Endpoint Path).

- **Body params**

POST http://scws.mydomain.com:10010/ScApp.svc/GetShopsDistance

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "nLatitude": 45.127787,
3   "nLongitude": 7.823318
4 }
```

The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'POST' and the URL is 'http://scws.mydomain.com:10010/ScApp.svc/GetShopsDistance'. Below this are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is selected and highlighted with an orange underline. Under the 'Body' tab, there are radio buttons for different content types: 'none', 'form-data', 'x-www-form-urlencoded', 'raw' (which is selected), 'binary', 'GraphQL', and 'JSON'. Below the radio buttons is a text editor showing a JSON body: { "nLatitude": 45.127787, "nLongitude": 7.823318 }. The text editor has line numbers 1 through 4 on the left.

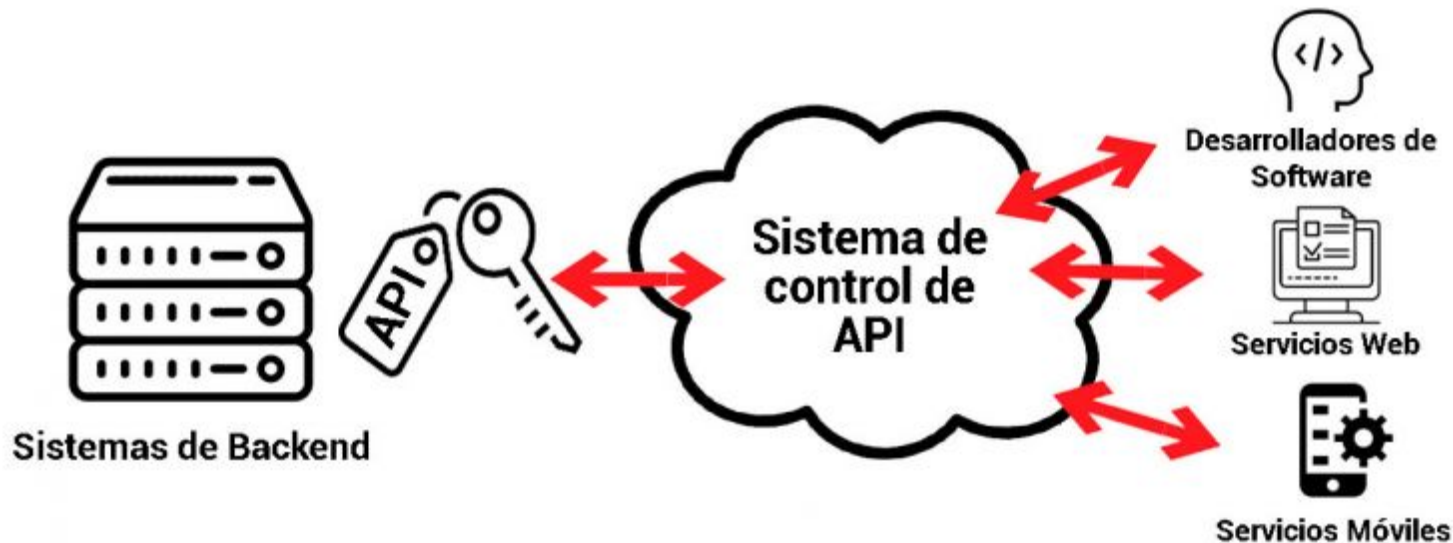
Headers - API-Keys y variables de entorno

BASH

```
USER_ID=239482 USER_KEY=foobar node app.js
```

JS

```
process.env.USER_ID // "239482"  
process.env.USER_KEY // "foobar"
```



Respuesta de una API

JSON: JavaScript Object Notation

95% de las respuestas API


```
const person = {  
  name: "John",  
  age: 30,  
  car: null  
};
```



```
'{"name":"John", "age":30, "car":null}'
```

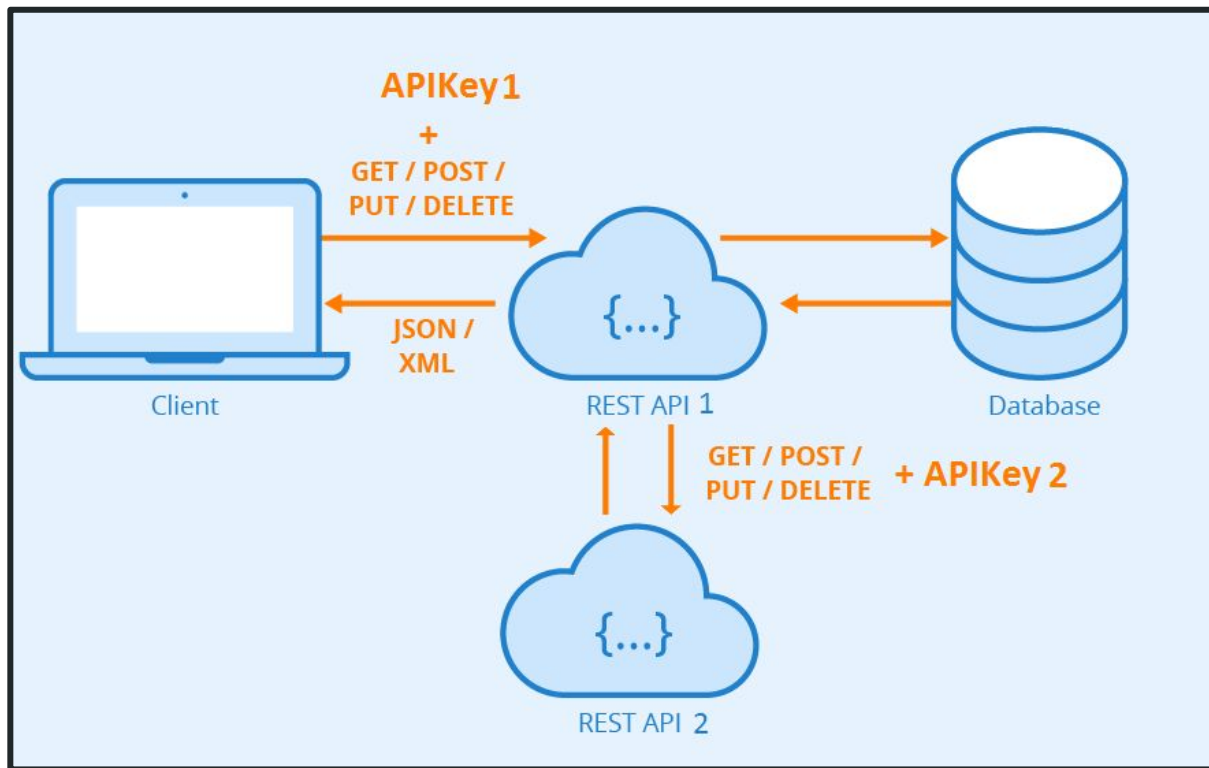
XML: Extensible Markup Language

```
<note>  
  <from>Jani</from>  
  <to>Tove</to>  
  <message>Remember me this  
  weekend</message>  
</note>
```

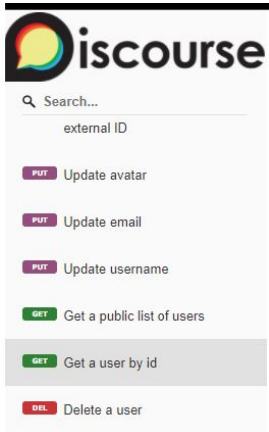


```
note.getElementsByTagName("from");
```

RESTful APIs



Cómo usar una API



Get a user by id

PATH PARAMETERS

id integer
required

Responses

> 200 response

GET /admin/users/{id}.json

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  "id": 0,
  "username": "string",
  "name": "string",
  "avatar_template": "string",
  "active": true,
```

```
const responseJson = await fetch('https://YYYYYYY/admin/users/Marc.json', {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json',
    'Api-Key': 'XXXX',
    'Api-Username': 'XXXX'
  },
});

const response = await JSON.parse(responseJson);
```

Cómo usar una API

Add group members

PATH PARAMETERS

→ id integer
required

REQUEST BODY SCHEMA: application/json

→ usernames string
comma separated list

Responses

> 200 success response

```
const data = {
  "usernames": "Putin, Trump"
};
const dataJson = await JSON.stringify(data);

const responseJson = await fetch('https://YYYYYYY/groups/WW3/members.json', {
  method: 'PUT',
  body: dataJson,
  headers: {
    'Content-Type': 'application/json',
    'Api-Key': 'XXXX',
    'Api-Username': 'XXXX'
  },
});

const response = await JSON.parse(responseJson);
```

PUT /groups/{id}/members.json

Request samples

Payload

Content type
application/json

Copy Expand all Collapse all

```
{
  "usernames": "username1,username2"
}
```

Response samples

200

Content type
application/json

Copy Expand all Collapse all

```
{
  "success": "string",
  - "usernames": [
    null
  ],
  - "emails": [
    null
  ]
}
```

Cómo construir una API



```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!')
});

app.post('/user', (req, res) => {
  const headers = req.headers;
  const body = req.body;
  // Verificaciones
  // Conexiones DB
  // ETC...
  res.status(200).end();
});

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
});
```

Muchas gracias por su atención
