



**DuocUC<sup>®</sup>** INFORMÁTICA Y  
TELECOMUNICACIONES

# ■ Fundamentos de HTTP y REST: Conceptos y Buenas Prácticas

---

DESARROLLO FULLSTACK I  
DSY1103

# En el capítulo anterior...

¿Qué consideraciones éticas debemos tener en los siguientes casos?



Al implementar algoritmos de inteligencia artificial en sistemas de recomendación es importante garantizar que las recomendaciones no promuevan estereotipos dañinos o filtren contenido de manera discriminatoria.



**Diseñar un sistema automatizado para la selección de currículums**



Los desarrolladores deben ser transparentes sobre cómo se toman las decisiones y evitan sesgos injustos en el proceso de selección





# • Contenidos

- ¿Qué es la Web y cómo funciona?.
- Conceptos de HTTP.
- Métodos y Respuestas de HTTP.
- REST.
- ¿Qué son las APIs?.
- Buenas prácticas en APIs REST.
- POSTMAN.
- Realizar actividades.
- Reflexión.



La tecnología **no es una cosa**, es una forma de **hacer cosas**.

– *Tim O'Reilly*

A black and white photograph of a man in a suit standing in a modern office, holding a tablet and smiling. The office has glass partitions and modern furniture. A blue square is in the top right corner.

01

## Cómo funciona la Web



# ¿Qué es la **Web**?



# • ¿Qué es la Web?

La **Web**, o **World Wide Web**, es una red global de información que permite el acceso a documentos y recursos a través de Internet. Consiste en:

## **PÁGINAS WEB**

Documentos de hipertexto que contienen texto, imágenes, videos y enlaces.

## **SITIOS WEB**

Colecciones de páginas web relacionadas, generalmente alojadas en un mismo dominio.



## **SERVIDORES WEB**

Computadoras que almacenan y entregan páginas web a los usuarios.

## **NAVEGADORES WEB**

Programas que permiten a los usuarios ver e interactuar con páginas web.



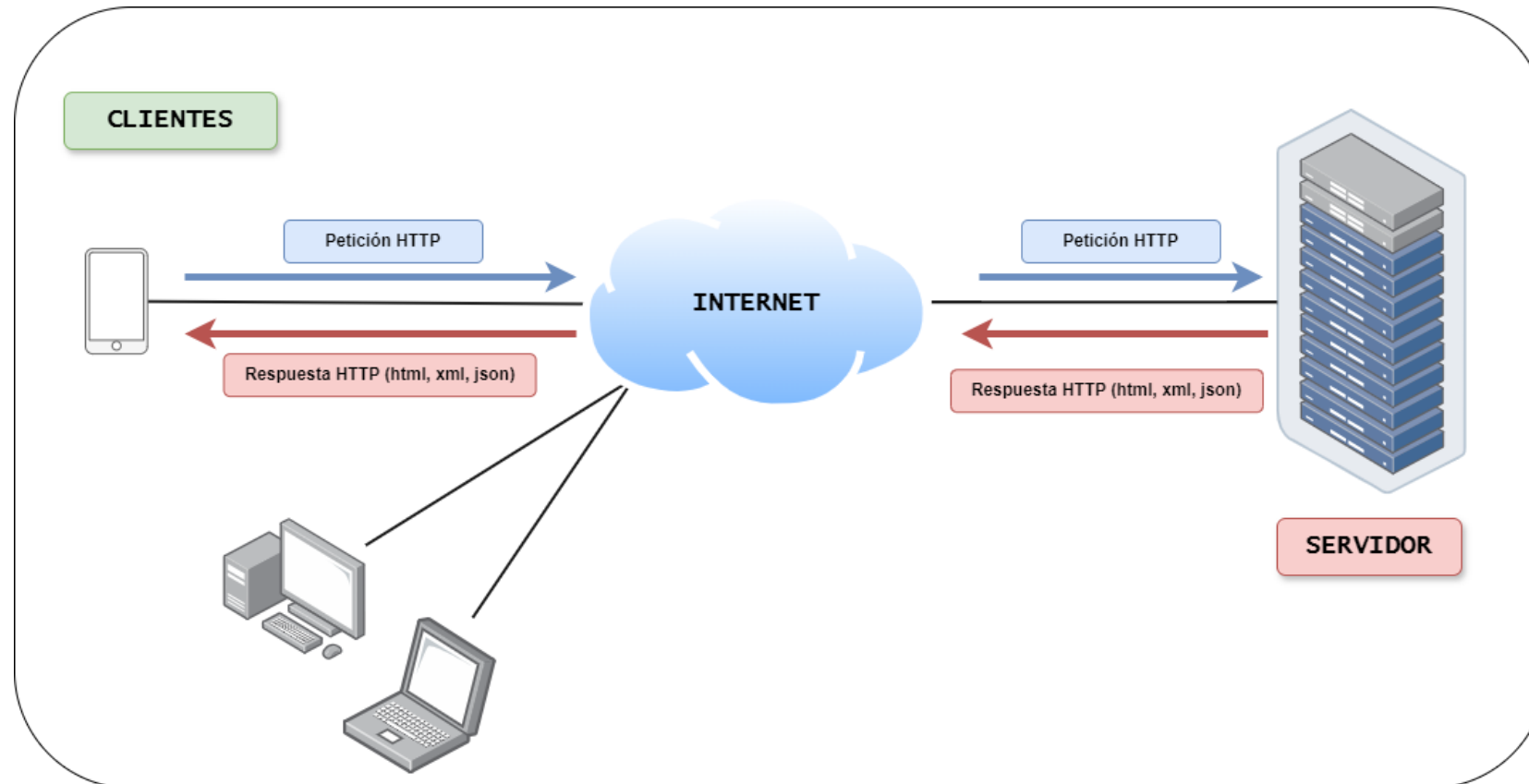
# ¿Cómo funciona la Web?





# • ¿Cómo funciona la Web?

La web funciona con un modelo **cliente-servidor**. El cliente (navegador web) envía solicitudes al servidor, y el servidor responde con el contenido solicitado.



# ¿Qué se entiende por **HTML**?

```
index.html X
A > index.html > html > body
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>👉 MATHIAS STORE 👉 - Tienda de Productos Deportivos 🇪🇸</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10  <header>
11    <div class="container">
12      <h1>👉 MATHIAS STORE 🇪🇸</h1>
13      <nav>
14        <ul>
15          <li><a href="#home">Inicio</a></li>
16          <li><a href="#products">Productos</a></li>
17        </ul>
18      </nav>
19    </div>
20  </header>
21  <main>
22    <section id="home">
23      <div class="container">
24        <h2>👉 Bienvenido a MATHIAS STORE 🇪🇸</h2>
25        <p>Tu tienda online de productos deportivos de alta calidad.</p>
26      </div>
27    </section>
28  <section id="products">...
29 </section>
30  <section id="about">...
31 </section>
32  <section id="contact">...
33 </section>
34 </main>
35 <footer>...
36 </footer>
37 </body>
38 </html>
```

# • HTML

**HTML** es un lenguaje de marcado utilizado para crear y estructurar páginas web. Está diseñado principalmente para la **presentación** y **navegación** de contenido en la web.



HTML utiliza principalmente los métodos **HTTP GET** y **POST** para interactuar con los servidores.

**GET** se usa para solicitar y mostrar contenido.

**POST** se usa para enviar datos de formularios al servidor.

Tradicionalmente, **HTML** solo soporta métodos **GET** y **POST** para formularios.

Los métodos **PUT**, **PATCH** y **DELETE** no son nativamente compatibles con formularios HTML.



A black and white photograph of a man and a woman in a server room. The man is holding a tablet and pointing at it, while the woman looks on. In the background, there are server racks and another person working at a desk. The scene is dimly lit, with light coming from the server racks and the tablet.

02

## Conceptos de HTTP

# • ¿Qué es HTTP?

HTTP (**HyperText Transfer Protocol**) es el protocolo utilizado por la web para la transferencia de documentos y datos.

El usuario ingresa una URL en el navegador.



<https://www.duoc.cl>

El navegador envía una solicitud HTTP al servidor web correspondiente.

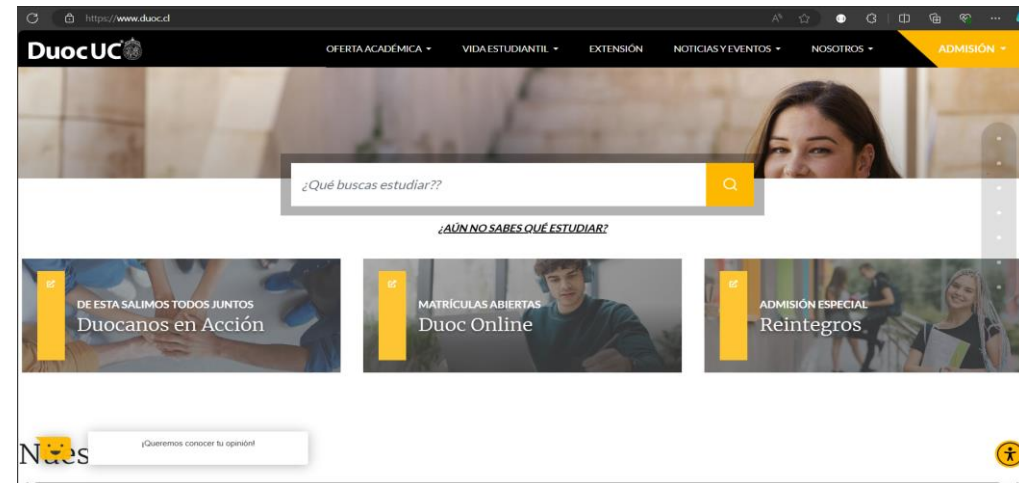
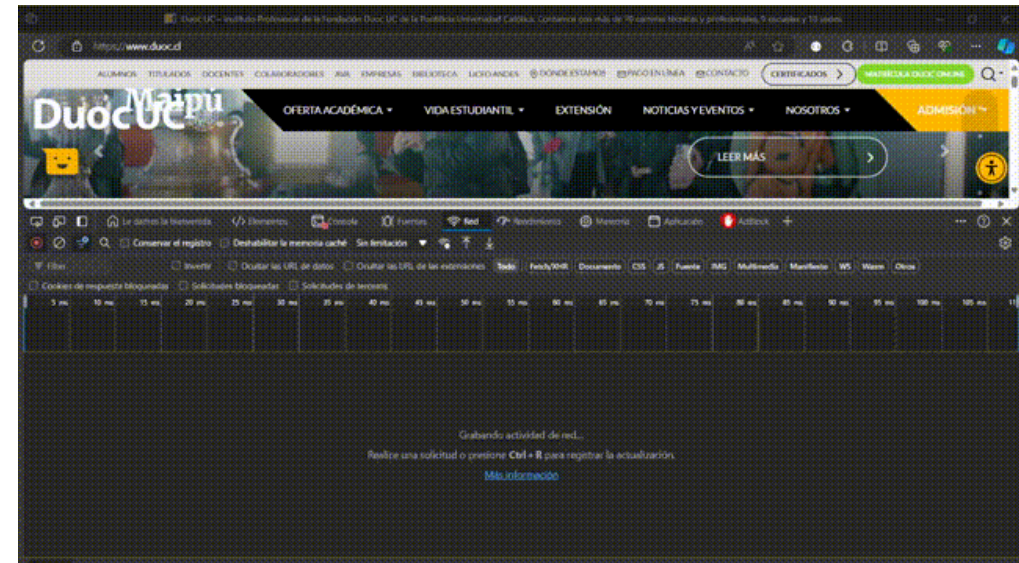


# • ¿Qué es HTTP?

El servidor web responde con el archivo HTML de la página web.

El navegador analiza el HTML y descarga los recursos adicionales (CSS, JavaScript, imágenes, etc.).

El navegador renderiza la página web completa para el usuario.





# • Métodos de HTTP

Son las **acciones** que se pueden realizar sobre los recursos en una arquitectura web, como solicitar, crear, actualizar o eliminar información. Cada uno tiene un propósito y semántica específica para interactuar con los recursos en el servidor.

GET

POST

PUT

DELETE

HEAD

OPTIONS

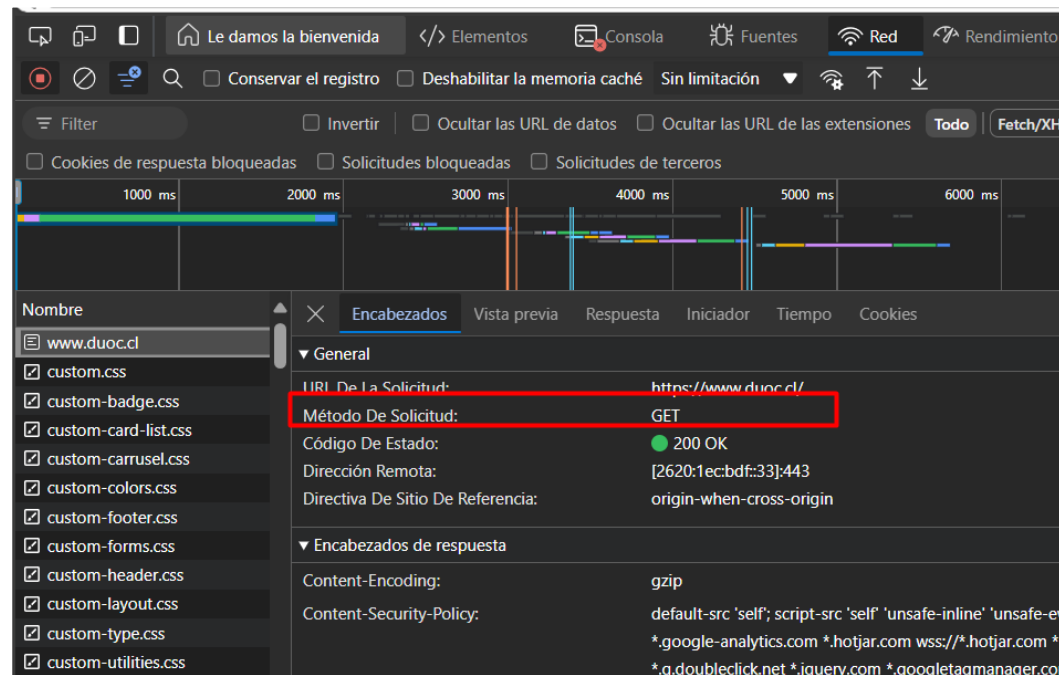
PATCH

# • Método GET

GET

El método **GET** se utiliza para **solicitar datos de un servidor**. Es una operación de solo lectura y no modifica los datos en el servidor.

Siempre que navegamos por internet o en una aplicación utilizamos este método para obtener recursos.



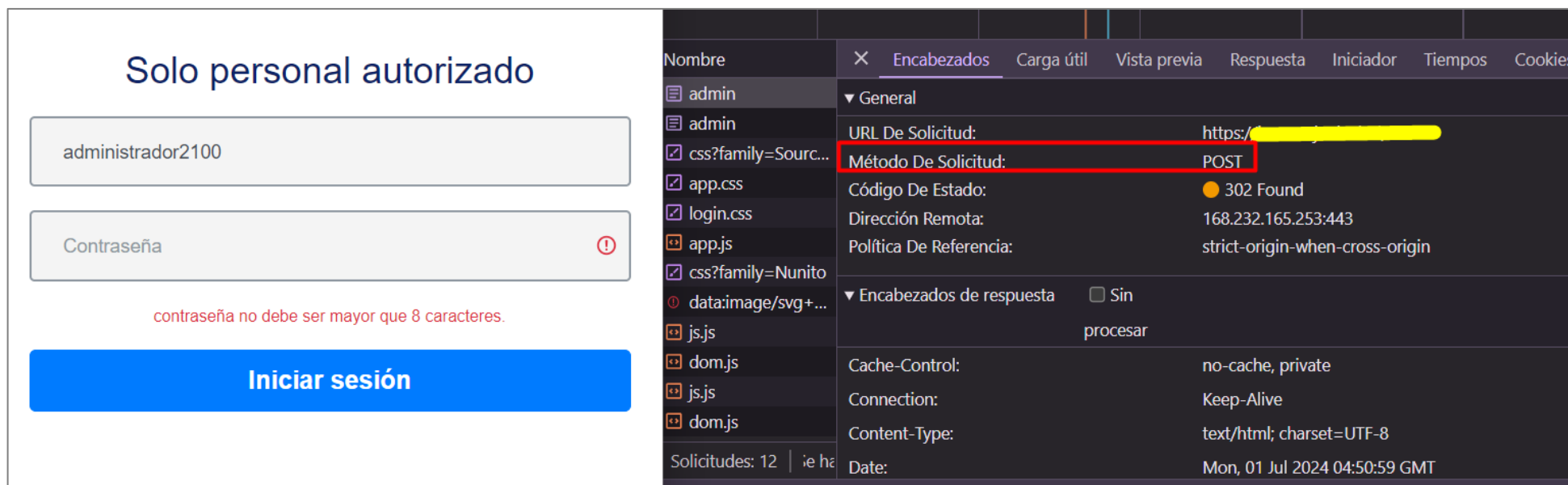
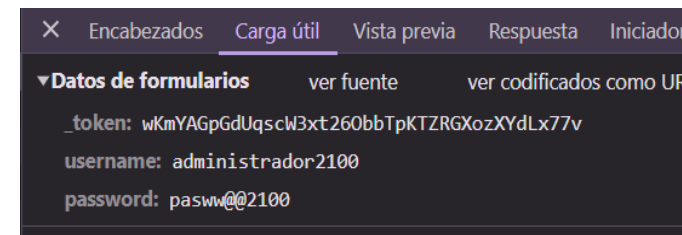
# • Método POST

POST

El método **POST** se utiliza para **enviar datos al servidor** con el fin de crear un nuevo recurso.

Se utiliza para enviar formularios, archivos, etc.

Los datos enviados se incluyen en el cuerpo de la solicitud.





## • Método PUT

PUT

El método **PUT** se utiliza para **reemplazar por completo un recurso existente** en el servidor con un nuevo conjunto de datos. Cuando se envía una solicitud **PUT**, se deben incluir todos los campos del recurso en el cuerpo de la solicitud, incluso si algunos de ellos no han cambiado.

## • Método PATCH

PATCH

El método **PATCH** se utiliza para realizar **una actualización parcial de un recurso existente** en el servidor. Cuando se envía una solicitud PATCH, solo se incluyen los campos del recurso que se van a actualizar en el cuerpo de la solicitud. Los campos que no se incluyen en la solicitud PATCH no se modifican.

PUT reemplaza por completo un recurso, mientras que PATCH actualiza solo los campos especificados de un recurso existente. PATCH es más eficiente cuando solo se necesita actualizar una parte de un recurso, ya que reduce el tamaño de la solicitud y la carga de procesamiento en el servidor.

## • Método DELETE

DELETE

El método **DELETE** se utiliza para **eliminar un recurso del servidor**, no se incluyen datos en el cuerpo de la solicitud.

## • Método HEAD

HEAD

El método **HEAD** es similar al método **GET**, pero en lugar de **devolver el cuerpo de la respuesta**, **solo devuelve los encabezados de la respuesta**.

## • Método OPTIONS

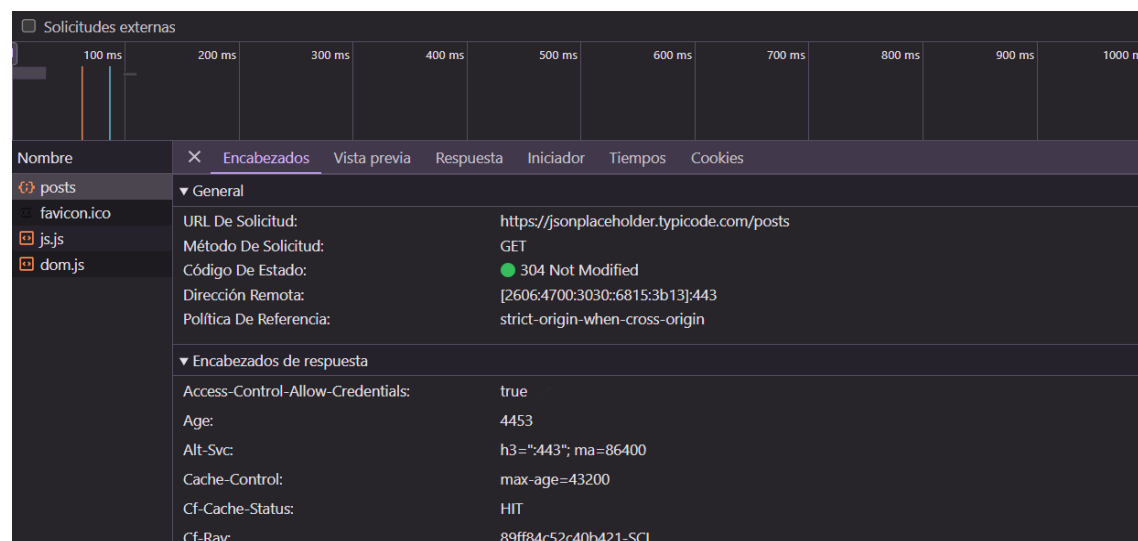
OPTIONS

El método **OPTIONS** se utiliza para **solicitar información sobre las opciones de comunicación disponibles** para un recurso determinado o un servidor en general.

# • Respuestas en HTTP

Son **mensajes enviados por el servidor al cliente** (normalmente un navegador web o una aplicación) en respuesta a una solicitud **HTTP**. Estas respuestas indican si la solicitud fue **exitosa** y proporcionan los datos solicitados o una explicación del error si la solicitud no se pudo completar.

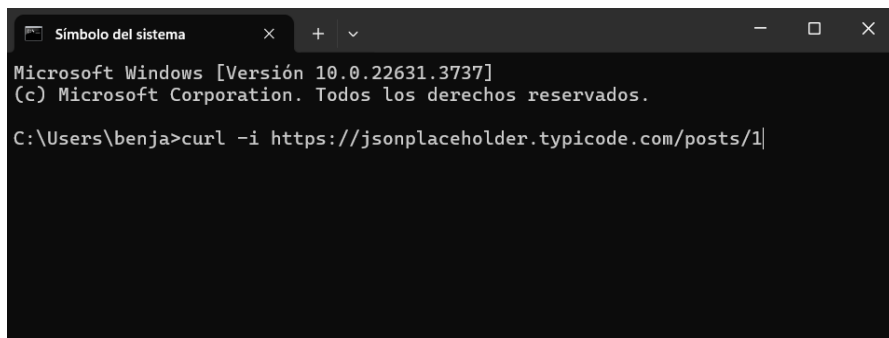
Cada respuesta HTTP incluye un código de estado HTTP y, opcionalmente, datos adicionales en el cuerpo de la respuesta.



# • Estructura de una Respuesta HTTP

Una respuesta HTTP típica tiene la siguiente estructura:

- **Línea de estado:** Indica el resultado de la solicitud, incluido el código de estado HTTP y una breve descripción.
- **Encabezados HTTP:** Proporcionan información adicional sobre la respuesta, como el tipo de contenido, la longitud del contenido y las políticas de caché.
- **Cuerpo de la respuesta:** Contiene los datos solicitados, como una página HTML, un archivo JSON, una imagen, etc.

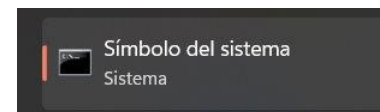


```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22631.3737]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\benja>curl -i https://jsonplaceholder.typicode.com/posts/1
```

- 1.- Abrir **Símbolo del sistema**
- 2.- escribir el siguiente comando curl

**curl -i https://jsonplaceholder.typicode.com/posts/1**

- 3.- Observa
- 4.- Identifica el **estado, encabezados y el cuerpo.**





# • Códigos de Estado HTTP

1XX

## CÓDIGOS INFORMATIVOS

El servidor ha recibido la petición y procederá con ella.

2XX

## CÓDIGOS DE ÉXITO

El servidor ha recibido, entendido y procesado la solicitud correctamente.

3XX

## CÓDIGOS DE REDIRECCIÓN

Estos códigos indican que se deben tomar más acciones para completar la solicitud, generalmente redirigiendo a una URL diferente..

4XX

## CÓDIGOS DE ERROR DE CLIENTE

Estos códigos indican que hubo un problema con la solicitud del cliente.

5XX

## CÓDIGOS DE ERROR DE SERVIDOR

Estos códigos indican que el servidor falló al cumplir una solicitud válida.

- ❑ **100 Continue:** El cliente debe continuar con la solicitud.
- ❑ **101 Switching Protocols:** El servidor acepta cambiar a un protocolo diferente.
- ❑ **200 OK:** La solicitud ha tenido éxito.
- ❑ **201 Created:** La solicitud ha sido cumplida y se ha creado un nuevo recurso.
- ❑ **202 Accepted:** La solicitud ha sido aceptada para procesamiento, pero el procesamiento no se ha completado.
- ❑ **204 No Content:** La solicitud ha tenido éxito, pero no hay contenido que enviar en la respuesta.

# • Códigos de Estado HTTP

1XX

## CÓDIGOS INFORMATIVOS

El servidor ha recibido la petición y procederá con ella.

2XX

## CÓDIGOS DE ÉXITO

El servidor ha recibido, entendido y procesado la solicitud correctamente.

3XX

## CÓDIGOS DE REDIRECCIÓN

Estos códigos indican que se deben tomar más acciones para completar la solicitud, generalmente redirigiendo a una URL diferente..

4XX

## CÓDIGOS DE ERROR DE CLIENTE

Estos códigos indican que hubo un problema con la solicitud del cliente.

5XX

## CÓDIGOS DE ERROR DE SERVIDOR

Estos códigos indican que el servidor falló al cumplir una solicitud válida.

- ❑ **301 Moved Permanently:** El recurso solicitado ha sido movido permanentemente a una nueva URL.
- ❑ **302 Found:** El recurso solicitado se encuentra temporalmente en una URL diferente.
- ❑ **304 Not Modified:** El recurso no ha sido modificado desde la última solicitud.
  
- ❑ **400 Bad Request:** La solicitud no pudo ser entendida o estaba mal formada.
- ❑ **401 Unauthorized:** La autenticación es necesaria y ha fallado o no se ha proporcionado.
- ❑ **403 Forbidden:** El servidor entiende la solicitud, pero se niega a autorizarla.
- ❑ **404 Not Found:** El recurso solicitado no pudo ser encontrado en el servidor.

# • Códigos de Estado HTTP

1XX

## CÓDIGOS INFORMATIVOS

El servidor ha recibido la petición y procederá con ella.

2XX

## CÓDIGOS DE ÉXITO

El servidor ha recibido, entendido y procesado la solicitud correctamente.

3XX

## CÓDIGOS DE REDIRECCIÓN

Estos códigos indican que se deben tomar más acciones para completar la solicitud, generalmente redirigiendo a una URL diferente..

4XX

## CÓDIGOS DE ERROR DE CLIENTE

Estos códigos indican que hubo un problema con la solicitud del cliente.

5XX

## CÓDIGOS DE ERROR DE SERVIDOR

Estos códigos indican que el servidor falló al cumplir una solicitud válida.

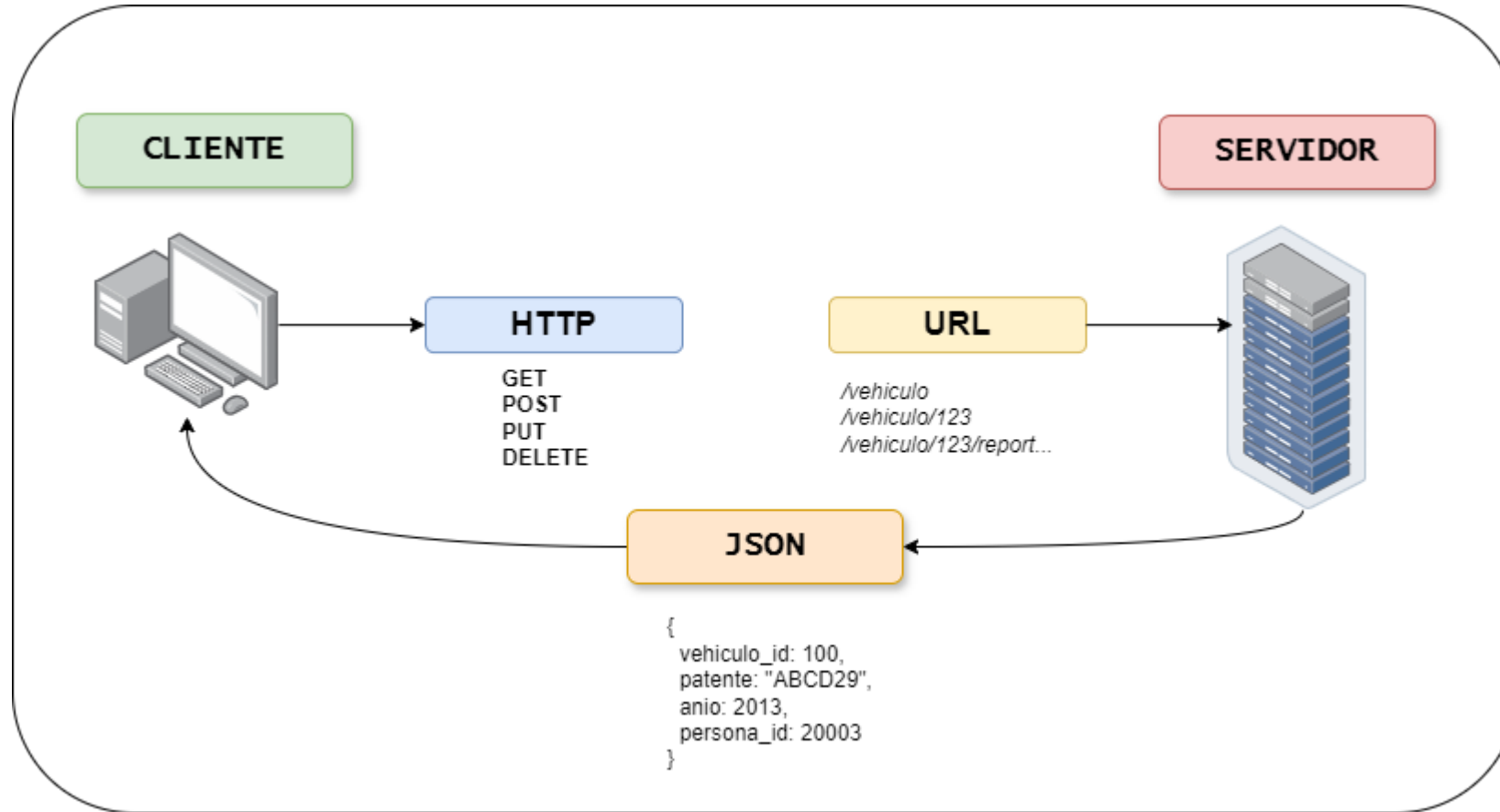
- ❑ **500 Internal Server Error:** Se ha producido un error en el servidor y no se puede completar la solicitud.
- ❑ **501 Not Implemented:** El servidor no reconoce el método de solicitud y no puede soportarlo.
- ❑ **502 Bad Gateway:** El servidor recibió una respuesta inválida de un servidor upstream.
- ❑ **503 Service Unavailable:** El servidor no está disponible actualmente (sobrecargado o en mantenimiento).



03

REST

# ¿Qué se entiende por **REST**?





# • REST

**REST (Representational State Transfer)** es una arquitectura de software que se utiliza para diseñar servicios web que permiten la comunicación entre diferentes sistemas a través de la red, especialmente en la web.

Es fundamental en el desarrollo de **APIs (Interfaces de Programación de Aplicaciones)** que permiten que diferentes aplicaciones se **comuniquen entre sí**.

REST utiliza los métodos HTTP (**GET, POST, PUT, PATCH, DELETE**) de manera específica para realizar operaciones CRUD (**Crear, Leer, Actualizar, Eliminar**) en los recursos.

Se centra en la manipulación de recursos a través de una **API web** utilizando métodos **HTTP** estándar.

Cada solicitud del cliente al servidor debe contener toda la información necesaria para entender y procesar la solicitud, ya que el **servidor no guarda ninguna información sobre el estado del cliente entre solicitudes**. Esto se conoce como **Stateless** (Sin estado).



# • Ventajas de usar REST en el backend

## SIMPLICIDAD Y ESTÁNDAR

Al utilizar métodos HTTP estándar, REST es fácil de entender y usar.

## ESCALABILIDAD

La naturaleza stateless de REST facilita la escalabilidad, ya que cada solicitud es independiente.

## FLEXIBILIDAD

REST permite el uso de diferentes formatos de datos (como JSON, XML, HTML), lo que lo hace muy flexible.

## INTEROPERABILIDAD

Al seguir estándares abiertos, las APIs RESTful pueden ser utilizadas por diferentes aplicaciones y sistemas, independientemente de la plataforma en la que estén desarrolladas.

# ¿Qué son las **APIs**?



# • APIs

**API (Application Programming Interface)**, es una interfaz que permite que **dos aplicaciones se comuniquen entre sí**. Las APIs definen el conjunto de reglas y protocolos que las aplicaciones deben seguir para solicitar y compartir datos o funcionalidades.

## SOLICITUD

El cliente (la aplicación que solicita información) envía una solicitud a la API.

## RESPUESTA

El servidor (la API) procesa la solicitud y devuelve una respuesta con los datos solicitados.

## FORMATO DE DATOS

JSON, XML, CSV, YAML, etc.

# • Formato de datos en APIs

## JSON (JavaScript Object Notation)

Formato ligero y fácil de leer/escribir.

```
informacion.json X
A > {} informacion.json > ...
1  {
2    "id": 123,
3    "run": "12345678-K",
4    "nombre": "Diego Palmeiro",
5    "correo": "d.palmeiro@duoc.cl",
6    "direccion": {
7      "calle": "Calle O'Higgins n° 123456",
8      "ciudad": "Concepción",
9      "pais": "Chile"
10   },
11   "telefonos": [
12     {
13       "tipo": "casa",
14       "numero": "123-456-789"
15     },
16     {
17       "tipo": "movil",
18       "numero": "987-654-321"
19     }
20   ]
21 }
```

## XML (eXtensible Markup Language)

Formato más pesado y estructurado.

```
informacion.xml X
A > {} informacion.xml
1  <usuario>
2    <id>123</id>
3    <run>12345678-K</run>
4    <nombre>Diego Palmeiro</nombre>
5    <correo>d.palmeiro@duoc.cl</correo>
6    <direccion>
7      <calle>Calle O'Higgins n° 123456</calle>
8      <ciudad>Concepción</ciudad>
9      <pais>Chile</pais>
10   </direccion>
11   <telefonos>
12     <telefono>
13       <tipo>casa</tipo>
14       <numero>123-456-789</numero>
15     </telefono>
16     <telefono>
17       <tipo>movil</tipo>
18       <numero>987-654-321</numero>
19     </telefono>
20   </telefonos>
21 </usuario>
```



# • Buenas prácticas para el diseño de APIs REST

Una **API REST (Interfaz de Programación de Aplicaciones Representational State Transfer)** es un conjunto de reglas y convenciones para crear e interactuar con servicios web. **REST** es una **arquitectura** de software que impone ciertas restricciones para el diseño de los servicios web, lo que facilita la creación de sistemas escalables, mantenibles y eficientes.

- **SIMPLICIDAD:** Las APIs REST son fáciles de entender y usar debido al uso de HTTP y sus métodos estándar.
- **ESCALABILIDAD:** La arquitectura REST permite la construcción de sistemas escalables.
- **FLEXIBILIDAD:** Los recursos pueden ser representados en múltiples formatos.
- **INTEROPERABILIDAD:** Las APIs REST pueden ser consumidas por diferentes plataformas y lenguajes de programación.

# • Buenas prácticas para el diseño de APIs REST

RECOMENDACIONES

## Versionado de la API

Facilita la evolución de la API sin romper la compatibilidad con versiones anteriores.

`/api/v1/libros`

`/api/v2.1/personas`

`/api/v2.2beta/boletas`

`/api/v2024/ventas`

# • Buenas prácticas para el diseño de APIs REST

RECOMENDACIONES

## Nombres Descriptivos y Consistentes

Utiliza sustantivos en **plural** para los nombres de los recursos.

`/api/v1/productos` en lugar de `/api/v1/producto`

`/api/v1/estudiantes/11/notas` en lugar de `/obtenerNotasEstudiante/11`

`/api/v1/productos/11`

`/api/v1/boleta/2000`

`/api/v1/compra/2000`

# • Buenas prácticas para el diseño de APIs REST

RECOMENDACIONES

## Nomenclatura de API para Recursos con Nombres Múltiples Palabras

Cuando tienes nombres de recursos que consisten en más de una palabra, es importante mantener una convención de nombres clara y consistente. La convención más común es utilizar la notación **kebab-case** o **camelCase** para separar las palabras.

`/api/v1/obtenerDatos`

`/api/v1/obtener-datos`

`/api/v1.2beta/informesDeVentas`

`/api/v1.2beta/informes-de-ventas`

`/api/v2/registrosAcademicos`

`/api/v2/registros-academicos`

`/api/v2025/informesDeVentas`

`/api/v2025/informes-de-ventas`

# • Buenas prácticas para el diseño de APIs REST

RECOMENDACIONES

## Uso Correcto de Verbos HTTP

- GET para obtener recursos.
- POST para crear nuevos recursos.
- PUT para actualizar recursos existentes.
- DELETE para eliminar recursos.

## Usa los Códigos Correctos

- 200 OK para solicitudes exitosas.
- 201 Created para creación exitosa de recursos.
- 204 No Content para eliminaciones exitosas.
- 400 Bad Request para solicitudes inválidas.
- 401 Unauthorized para acceso no autorizado.
- 404 Not Found para recursos no encontrados.
- 500 Internal Server Error para errores del servidor.

GET

POST

PUT

DELETE



# • Buenas prácticas para el diseño de APIs REST

RECOMENDACIONES

## Parámetros de Ruta y Consulta

- Utiliza parámetros de ruta para identificar recursos específicos.
- Utiliza parámetros de consulta para filtros y opciones.
- Ejemplo:
  - Parámetro de ruta: `/api/v1/products/{id}`
  - Parámetro de consulta: `/api/v1/products?category=electronics`

## Jerarquía y Anidamiento Lógico

- Organiza los endpoints para reflejar relaciones entre recursos.
- `/api/v1/products/{productId}/reviews` para las reseñas de un producto específico.

# • Buenas prácticas para el diseño de APIs REST

API	MÉTODO HTTP	PATH	CÓDIGO DE ESTADO	DESCRIPCIÓN
Listar libros	GET	/api/v1/libros	200 (OK)	Se recupera todos los recursos
Guardar libro	POST	/api/v1/libros	201 (Created)	Se crea un nuevo recurso
Obtener libro	GET	/api/v1/libros/{id}	200 (OK)	Se recupera un recurso
Actualizar libro	PUT	/api/v1/libros/{id}	200 (OK)	Se actualiza un recurso
Eliminar libro	DELETE	/api/v1/libros/{id}	204 (No Content)	Se elimina un recurso

# • Buenas prácticas para el diseño de APIs REST

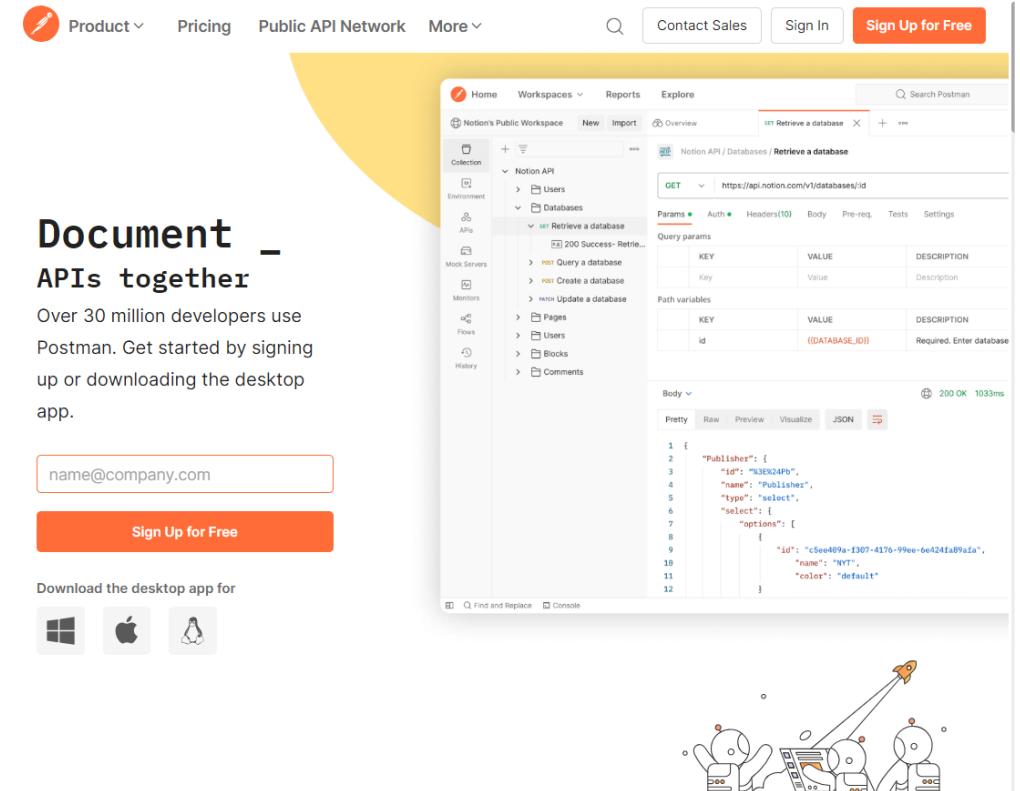
API	MÉTODO HTTP	PATH	CÓDIGO DE ESTADO	DESCRIPCIÓN
Listar prestamos	GET	/api/v1/prestamos	200 (OK)	Se recupera todos los recursos
Guardar prestamo	POST	/api/v1/prestamos	201 (Created)	Se crea un nuevo recurso
Obtener prestamo	GET	/api/v1/prestamos/{id}	200 (OK)	Se recupera un recurso
Actualizar prestamo	PUT	/api/v1/prestamos/{id}	200 (OK)	Se actualiza un recurso
Eliminar prestamo	DELETE	/api/v1/prestamos/{id}	204 (No Content)	Se elimina un recurso

# • POSTMAN

**Postman** es una **herramienta** de desarrollo de software utilizada para **probar APIs**.



Ver presentación  
n rPruebas



IR A POSTMAN

# • ACTIVIDAD 1

Utilizar **Postman** para realizar consultas a APIs públicas, comprendiendo los conceptos básicos de HTTP y los diferentes métodos de solicitud.

Ingresa a AVA de esta actividad y sigue las instrucciones de la guía *2.1.3 Actividad – Descubriendo y probando APIs públicas*.

Comparte tus resultados con tus compañeros.



## • ACTIVIDAD 2

Ver el video de [Youtube](#), anotar cada termino o tecnología mencionada por la empresa entrevistada, busca su definición y rellenar el tablero.

Ingresa a AVA de esta actividad y sigue las instrucciones de la guía *2.1.4 Actividad - Análisis técnico a una empresa tecnológica de delivery*

Comparte tus resultados con tus compañeros.





# Reflexionemos



- ¿Qué papel juegan los métodos HTTP en la interacción entre cliente y servidor?
- ¿Por qué es importante que las APIs sigan estándares abiertos como REST?
- ¿Qué limitaciones presenta HTML en cuanto a la compatibilidad con métodos HTTP como PUT y DELETE?
- ¿Cómo puede Postman ayudar en el desarrollo y prueba de APIs?

04

¡Muchas gracias!

# DuocUC<sup>®</sup>

CERCANÍA. LIDERAZGO. FUTURO.

**duoc.cl**

**7 AÑOS**  
ACREDITADO



DESDE AGOSTO 2017 HASTA AGOSTO 2024.  
DOCENCIA DE PREGRADO. GESTIÓN  
INSTITUCIONAL. VINCULACIÓN CON EL MEDIO.