Ayudantía 3

Consumo de APIs

IIC2513 - Tecnologías y Aplicaciones Web

Andrés Venegas – Gabriel Quiroz

En base a la Ayudantía realizada el **2023-2** por **Juan Fernández** y **Pedro Ríos**

Contenidos

01 API/RESTful

02 Métodos HTTP

03 Headers y body

04 Códigos de estado de respuesta HTTP

05 Axios

06 Postman

¿Qué es una API? 🤔

Es un **intermediario** que define cómo deben interactuar las aplicaciones entre sí, permitiendo acceso controlado a sus **funciones** y **datos**.

Una API **expone** una lista de funciones, datos o servicios que una aplicación puede solicitar mediante solicitudes **HTTP**.

¿Qué es una API? 🤔

Es un **intermediario** que define cómo deben interactuar las aplicaciones entre sí, permitiendo acceso controlado a sus **funciones** y **datos.**

Una API **expone** una lista de funciones, datos o servicios que una aplicación puede solicitar mediante solicitudes **HTTP**.

¿Qué es una API? 🤔

Es un **intermediario** que define cómo deben interactuar las aplicaciones entre sí, permitiendo acceso controlado a sus **funciones** y **datos**.

Una API **expone** una lista de funciones, datos o servicios que una aplicación puede solicitar mediante solicitudes **HTTP**.

¿Qué es una API?

https://www.flaticon.com/

¿Qué es una API?

Analogía

https://www.flaticon.com/

¿Qué es una API?

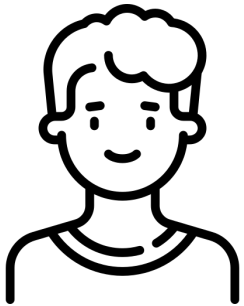
Analogía

✨RESTAURANTE✨

https://www.flaticon.com/

¿Qué es una API?

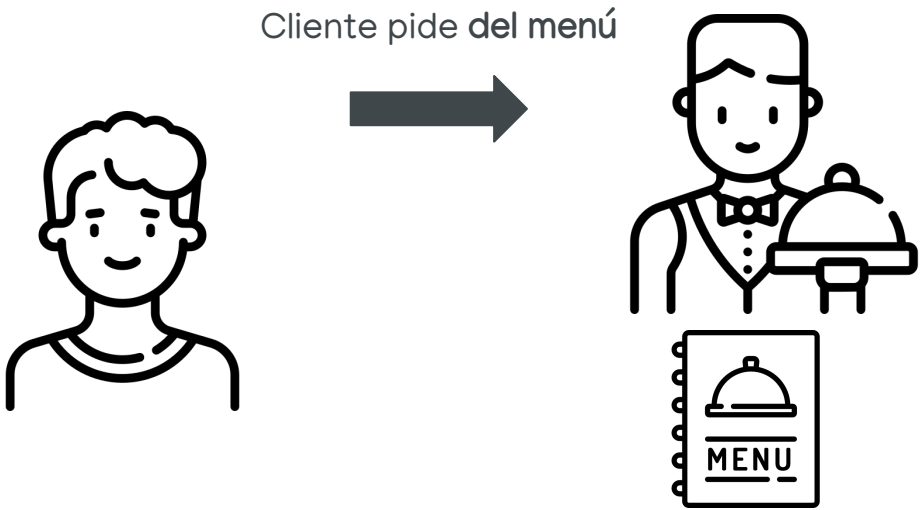
Analogía



https://www.flaticon.com/

¿Qué es una API?

Analogía



https://www.flaticon.com/

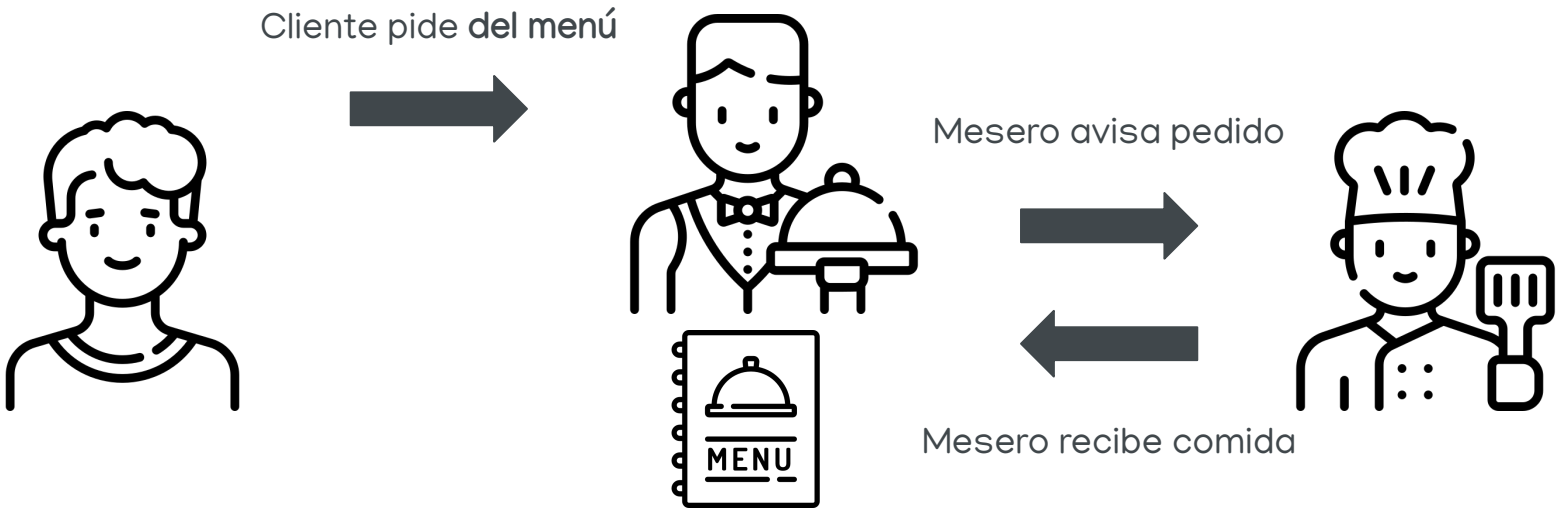
¿Qué es una API?

Analogía

https://www.flaticon.com/

¿Qué es una API?

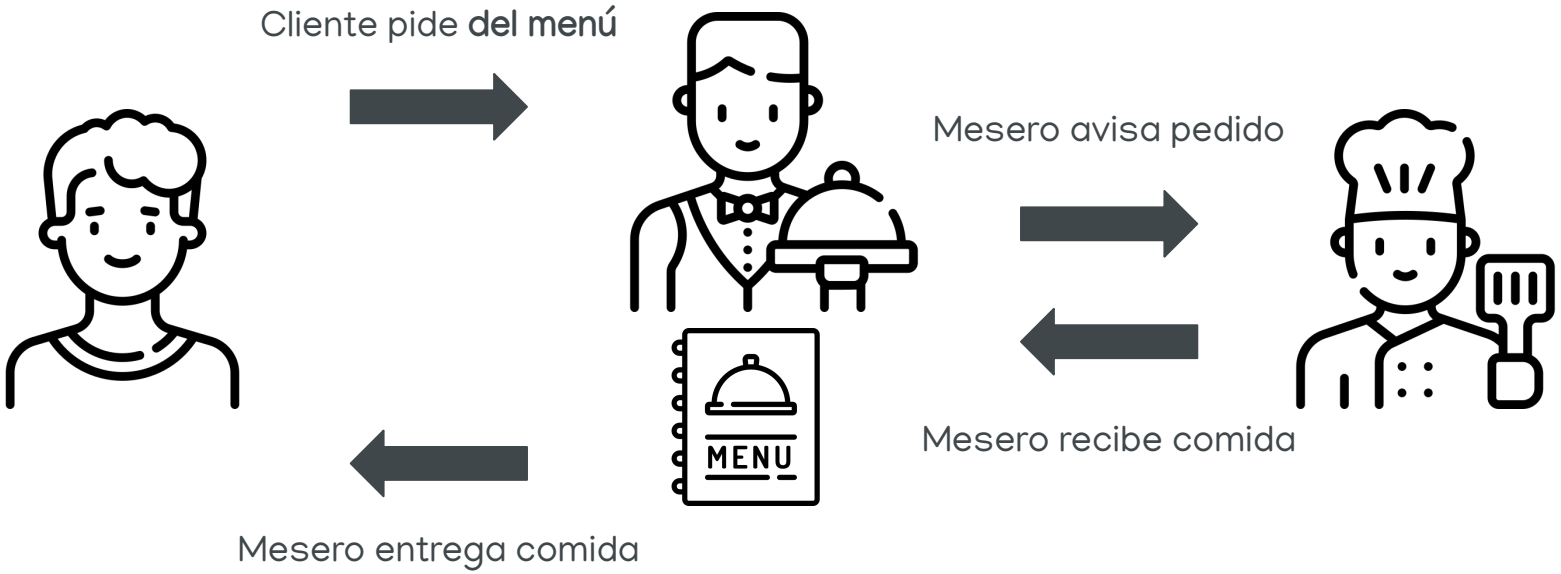
Analogía



https://www.flaticon.com/

¿Qué es una API?

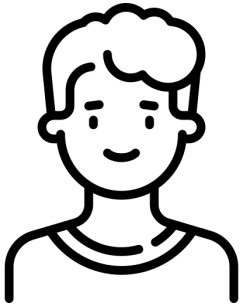
Analogía



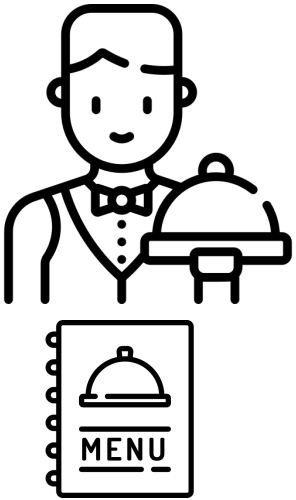
https://www.flaticon.com/

¿Qué es una API?

Realidad



CLIENTE



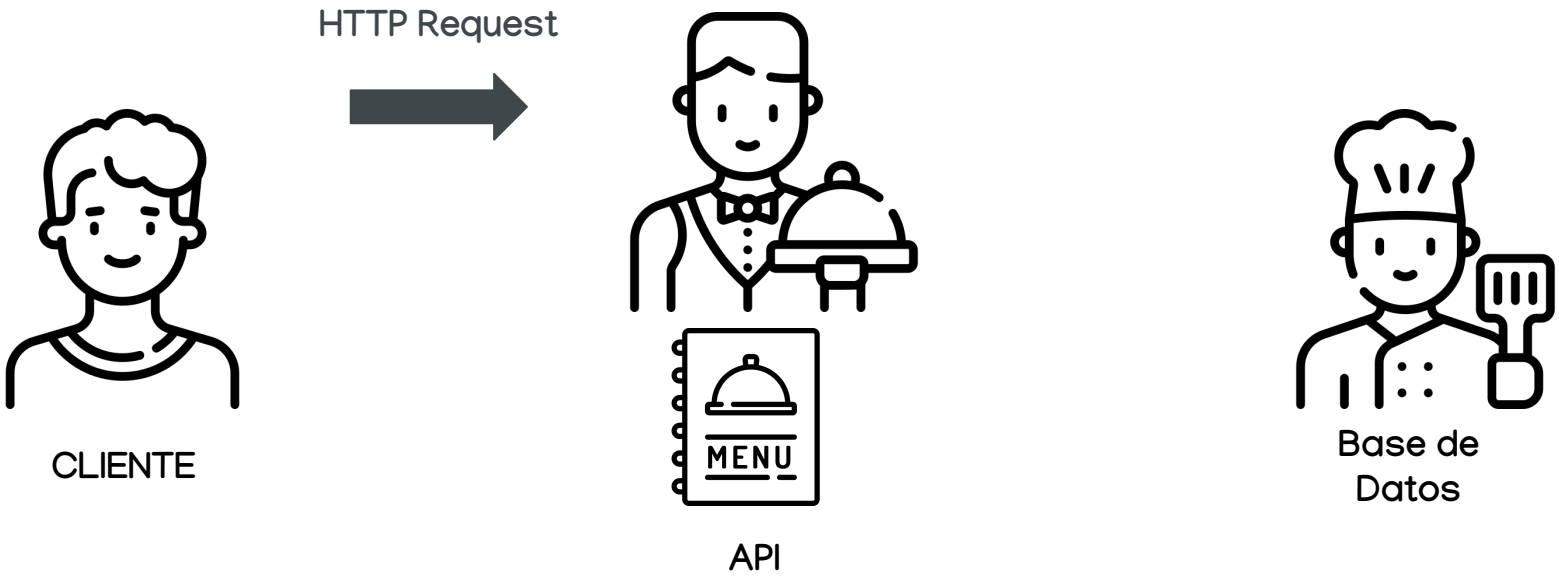
API

Base de Datos

https://www.flaticon.com/

¿Qué es una API?

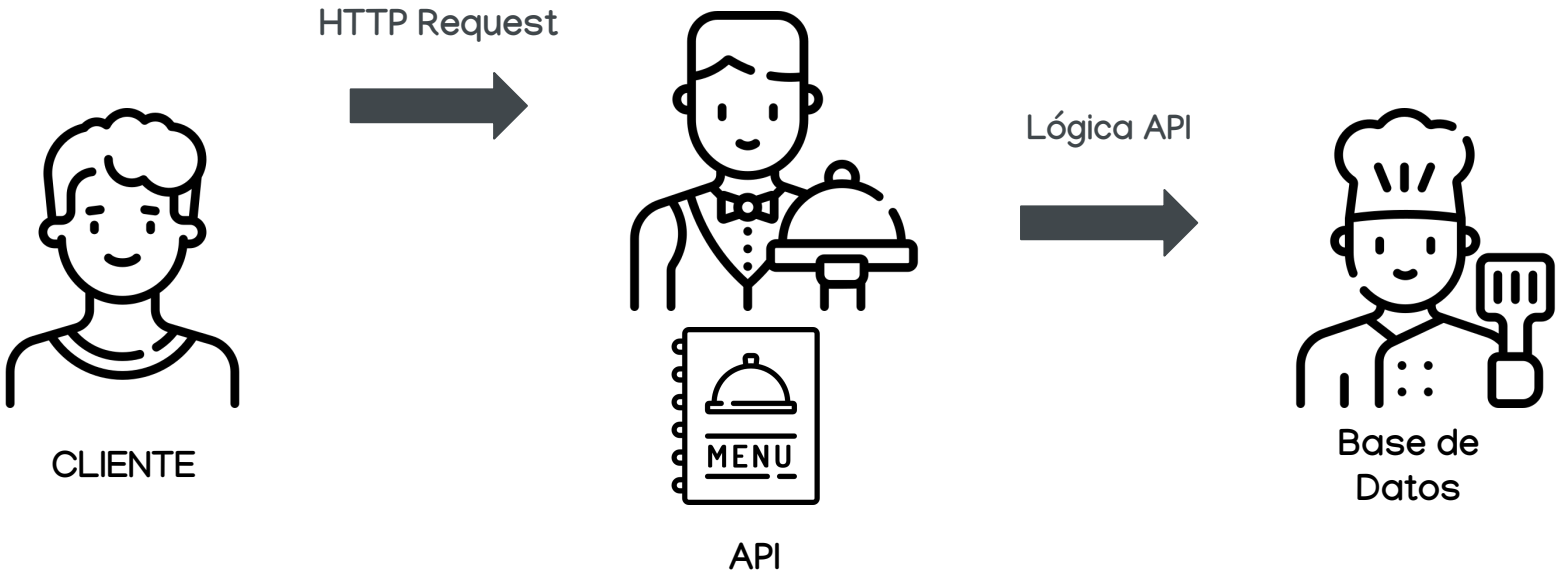
Realidad



https://www.flaticon.com/

¿Qué es una API?

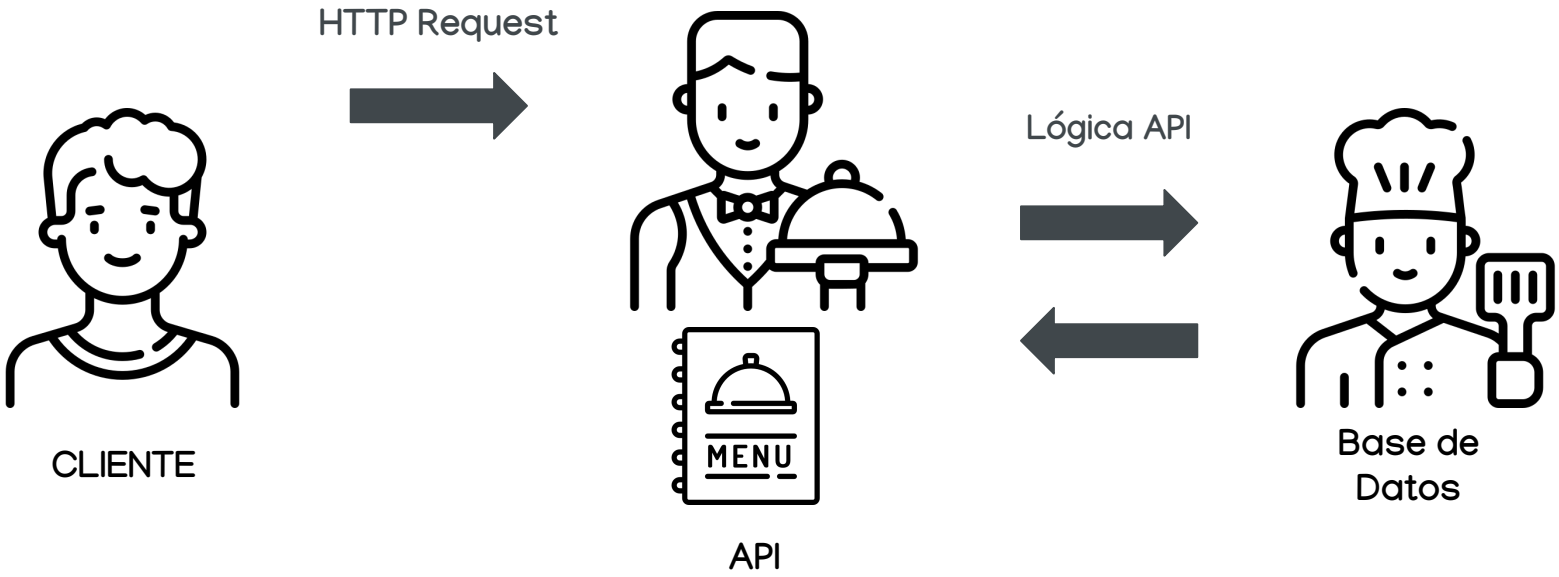
Realidad



https://www.flaticon.com/

¿Qué es una API?

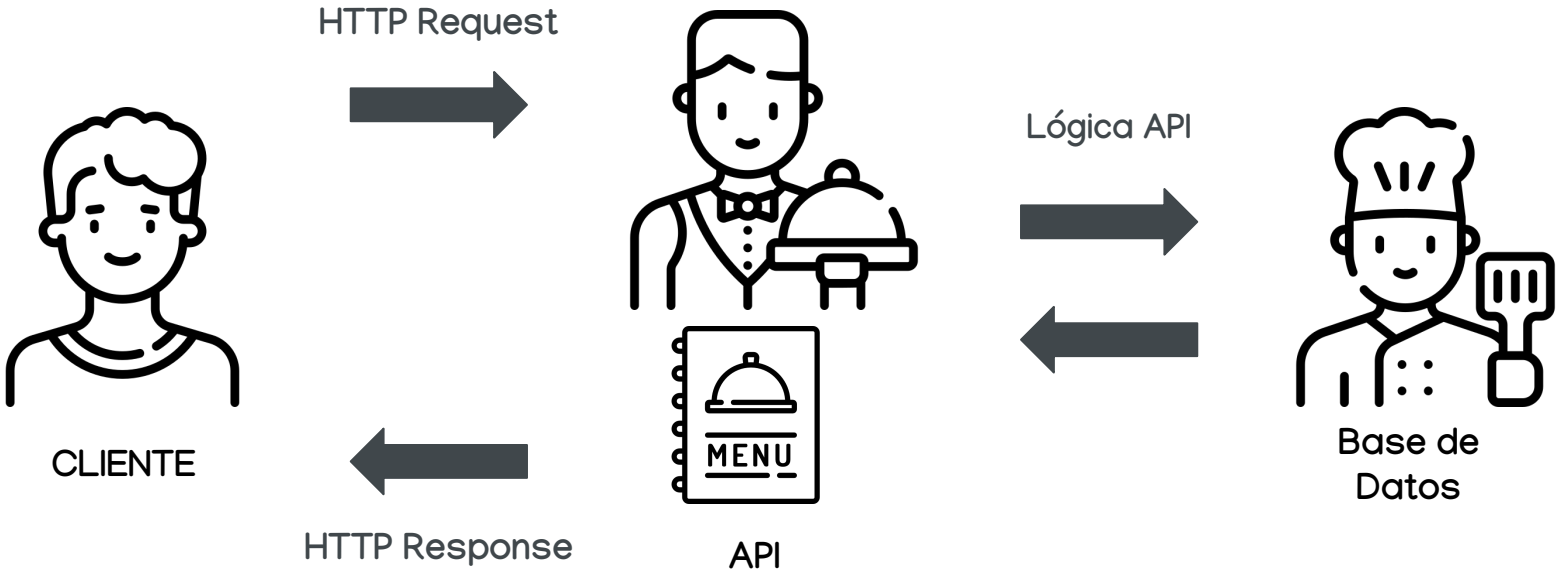
Realidad



https://www.flaticon.com/

¿Qué es una API?

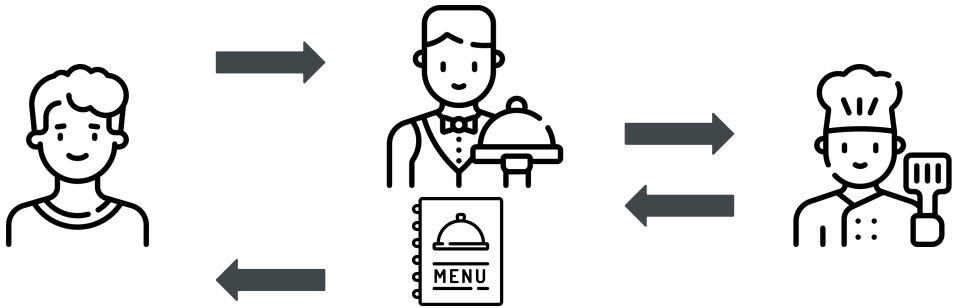
Realidad



https://www.flaticon.com/

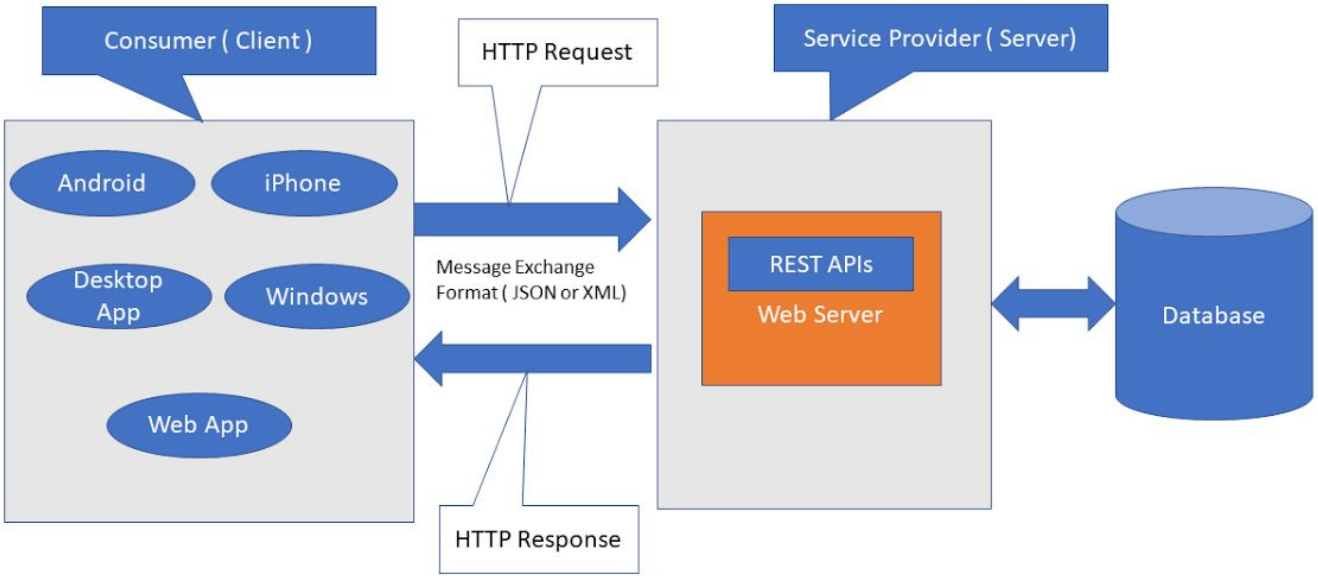
¿Qué es una API?

- Existen **opciones específicas** de lo que le podemos solicitar a la **API** (existe un ***menú***).
- El cliente **no tiene conocimiento** de lo que pasa en la **BDD** (***cocina***), solo sabe que recibe lo que solicitó.



https://www.flaticon.com/

Arquitectura API RESTful



¿Qué es REST?

REST (REpresentational State Transfer) es un conjunto de principios y convenciones que debe cumplir un servicio web.

Los servicios REST utilizan los métodos HTTP, como **GET/POST/PUT/DELETE**, para realizar operaciones sobre los recursos (objetos, documentos, o datos).

De la clase, les deberían sonar los **Principios REST**

Principios REST

- **Protocolo Cliente-Servidor:** el cliente y servidor deben estar separados y poder evolucionar independientemente.
- **Sin estado:** cada solicitud del cliente al servidor debe contener toda la información necesaria para comprender y procesar la solicitud.
- **Cacheable:** el servicio debe poder indicar si una respuesta es cacheable. El caché del cliente puede reciclar los datos de respuesta para solicitudes similares en el futuro.
- **Interfaz uniforme:** identificación única de recursos mediante URL, manipulación a través de representaciones (como JSON o XML), respuestas autodescriptivas y uso de hipermedia para guiar al cliente.
- **Sistema de capas:** los componentes están separados en capas, evitando la interacción entre capas y restringiendo su visibilidad más allá de la capa inmediata.

Métodos HTTP en REST

Los métodos **HTTP** son acciones que permiten a las aplicaciones interactuar con recursos en una **API REST.**

- **GET: recuperar** información de un recurso
- POST: **crear** un nuevo recurso
- PUT: **actualizar** un recurso existente por **completo**
- PATCH: **actualizar parcialmente** un recurso existente
- DELETE: **eliminar** un recurso

Métodos HTTP en REST

Los métodos **HTTP** son acciones que permiten a las aplicaciones interactuar con recursos en una **API REST.**

- GET: **recuperar** información de un recurso
- **POST: crear** un nuevo recurso
- PUT: **actualizar** un recurso existente por **completo**
- PATCH: **actualizar parcialmente** un recurso existente
- DELETE: **eliminar** un recurso

Métodos HTTP en REST

Los métodos **HTTP** son acciones que permiten a las aplicaciones interactuar con recursos en una **API REST.**

- GET: **recuperar** información de un recurso
- POST: **crear** un nuevo recurso
- **PUT: actualizar** un recurso existente por **completo**
- PATCH: **actualizar parcialmente** un recurso existente
- DELETE: **eliminar** un recurso

Métodos HTTP en REST

Los métodos **HTTP** son acciones que permiten a las aplicaciones interactuar con recursos en una **API REST.**

- GET: **recuperar** información de un recurso
- POST: **crear** un nuevo recurso
- PUT: **actualizar** un recurso existente por **completo**
- **PATCH: actualizar parcialmente** un recurso existente
- DELETE: **eliminar** un recurso

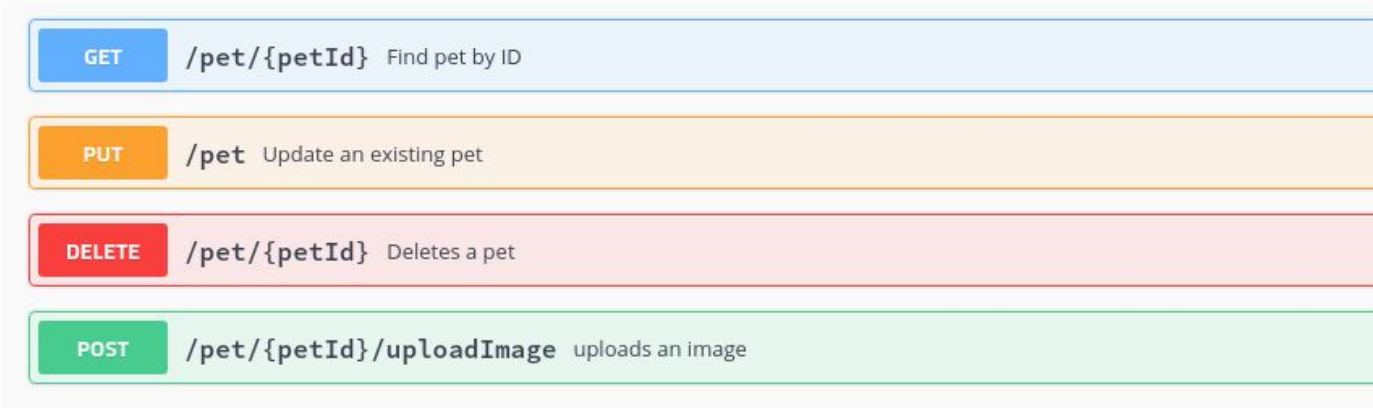
Métodos HTTP en REST

Los métodos **HTTP** son acciones que permiten a las aplicaciones interactuar con recursos en una **API REST.**

- GET: **recuperar** información de un recurso
- POST: **crear** un nuevo recurso
- PUT: **actualizar** un recurso existente por **completo**
- PATCH: **actualizar parcialmente** un recurso existente
- **DELETE: eliminar** un recurso

Endpoints

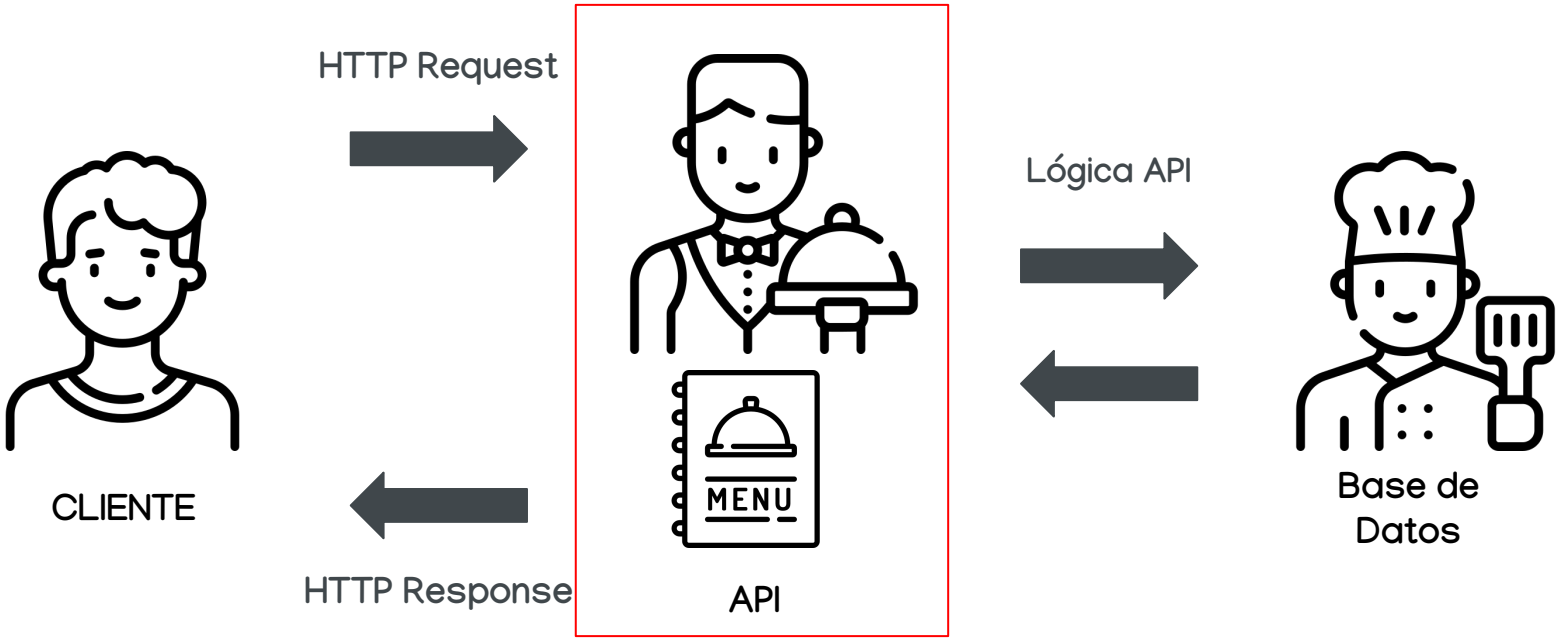
Son un **punto de acceso** específico que se utiliza para realizar operaciones sobre recursos.



¿Qué es una API?

Comparemos una API con lo que sucede en un **restaurante**

Realidad



https://www.flaticon.com/

Contenidos

01 API/RESTful

02 Métodos HTTP

03 **Headers y body**

04 Códigos de estado de respuesta HTTP

05 Axios

06 Postman

Headers y Body

Los **Headers** y el **Body** son componentes fundamentales de las solicitudes y respuestas **HTTP** que permiten la comunicación efectiva entre el cliente y el servidor en la web.

https://www.sitepoint.com/rest-api/

Headers

Se encuentran en la parte superior de la **solicitud** o **respuesta** y contienen **metadatos** clave que proporcionan información crucial sobre la comunicación entre el cliente

https://www.sitepoint.com/rest-api/

Principales Headers

1. **Content-Type:** Este encabezado especifica el tipo de contenido que se encuentra en el cuerpo de la solicitud o respuesta.

2. **Authorization:** El encabezado de autorización se utiliza para enviar credenciales de autenticación al servidor. Esto es esencial cuando se necesita autenticar el acceso a recursos protegidos, como contraseñas o tokens de acceso

3. **Accept:** Indica los tipos de contenido que el cliente está dispuesto a aceptar como respuesta. Ayuda al servidor a seleccionar la mejor representación del recurso solicitado si hay múltiples opciones disponibles.

Body

O cuerpo de la **solicitud** o **respuesta** HTTP, es donde se encuentran los datos reales que se envían entre el cliente y el servidor. La utilidad del cuerpo radica en la capacidad de transmitir información más allá de metadatos

https://www.sitepoint.com/rest-api/

Aspectos relevantes

1. **Envío de datos:** El cuerpo de una solicitud se utiliza para enviar datos desde el cliente al servidor. Por ejemplo, en una solicitud POST para crear una nueva entrada en un blog, el cuerpo puede contener el texto del artículo que se está creando.

2. **Recepción de datos:** El cuerpo de una respuesta se utiliza para transmitir datos desde el servidor al cliente. Esto puede incluir el contenido de una página web, una imagen, un archivo JSON con resultados de una consulta, etc.

Aspectos relevantes

3. **Formato de datos:** El cuerpo puede contener datos en diversos formatos, como JSON, XML, HTML, texto sin formato, imágenes, videos y más. La elección del formato depende del tipo de datos que se esté transmitiendo.

4. **Estado y respuestas:** En las respuestas del servidor, el cuerpo a menudo contiene información sobre el estado de la solicitud, como códigos de estado (por ejemplo, 200 OK o 404 Not Found) y mensajes descriptivos que ayudan a los clientes a comprender la respuesta

Contenidos

01 API/RESTful

02 Métodos HTTP

03 Headers y body

04 Códigos de estado de respuesta HTTP

05 Axios

06 Postman

Códigos de respuesta

Los códigos de respuesta HTTP son una parte fundamental del protocolo HTTP (Hypertext Transfer Protocol), que es el protocolo utilizado para la comunicación entre clientes (como navegadores web o aplicaciones) y servidores web. Estos códigos se utilizan para **proporcionar información sobre el resultado de una solicitud** realizada por un cliente a un servidor web.

Principales códigos

1xx: Respuestas informativas.

2xx: Respuestas exitosas.

3xx: Redirecciones.

4xx: Errores del cliente.

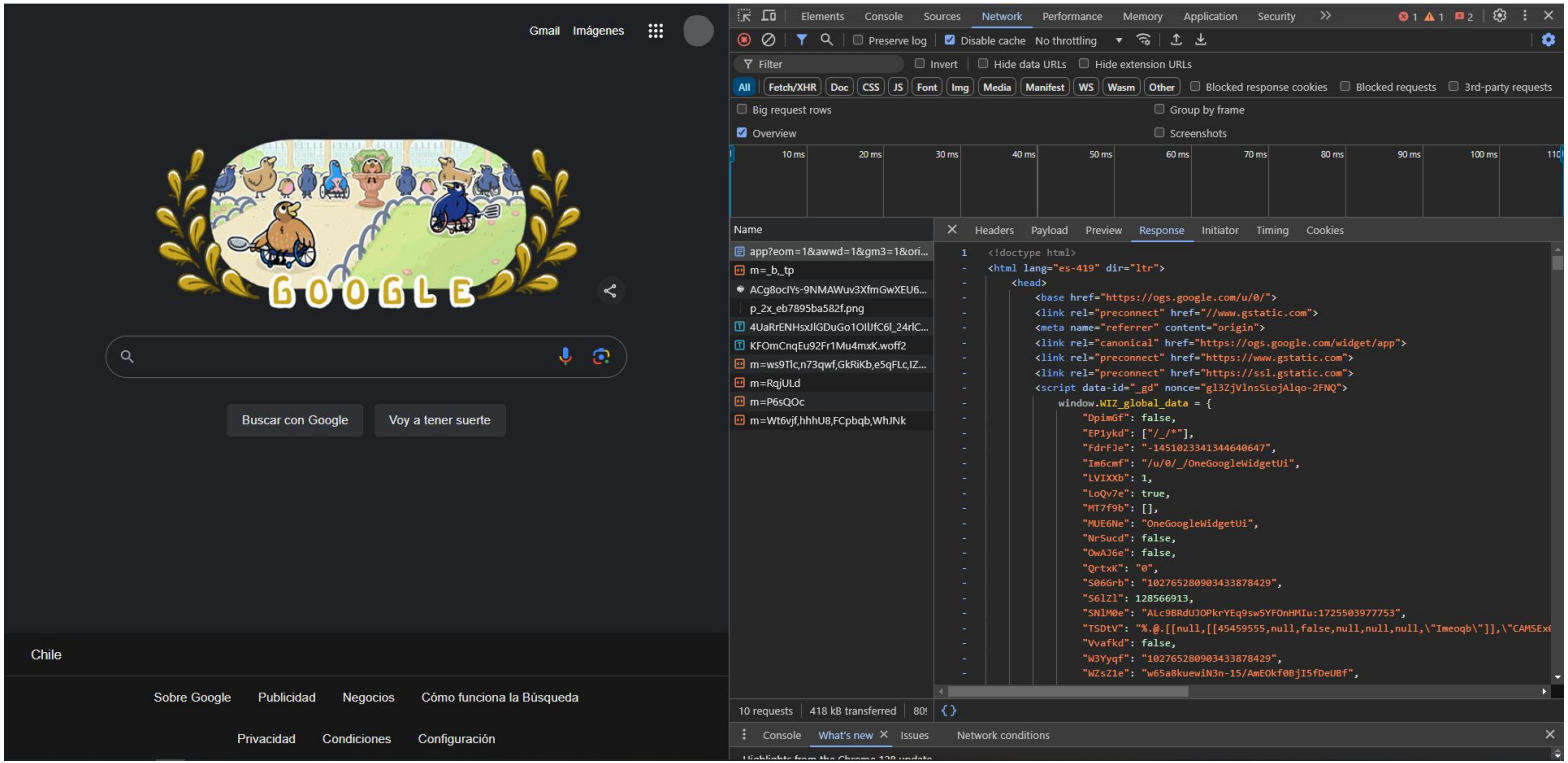
5xx: Errores del servidor.

Algunos ejemplos comunes de códigos de respuestas incluyen el código 200 (OK, solicitud exitosa), 404 (No encontrado, recurso no disponible), y 500 (Error interno del servidor).

Lista completa de códigos [VER] +
https://http.cat/

Ejemplo de respuesta

Ejemplo de inspeccionar



Contenidos

01 API/RESTful

02 Métodos HTTP

03 Headers y body

04 Códigos de estado de respuesta HTTP

05 **Axios**

06 Postman

Axios

Axios es un cliente HTTP simple basado en promesas para el navegador y node.js. Axios provee una librería fácil de usar en un paquete pequeño con una interfaz muy extensible”

Axios

Axios es un cliente HTTP simple basado en promesas para el navegador y node.js. Axios provee una librería fácil de usar en un paquete pequeño con una interfaz muy extensible

Axios

Axios es un cliente HTTP simple basado en promesas para el navegador y node.js. Axios provee una librería fácil de usar en un paquete pequeño con una interfaz muy extensible

Librería de Node.js que permite comunicarse con una API mediante solicitudes y respuestas

Axios

Axios es un cliente HTTP simple basado en promesas para el navegador y node.js. Axios provee una librería fácil de usar en un paquete pequeño con una interfaz muy extensible

Librería de Node.js que permite comunicarse con una API mediante solicitudes y respuestas

https://axios-http.com/docs/intro

¿Cómo se ocupa?

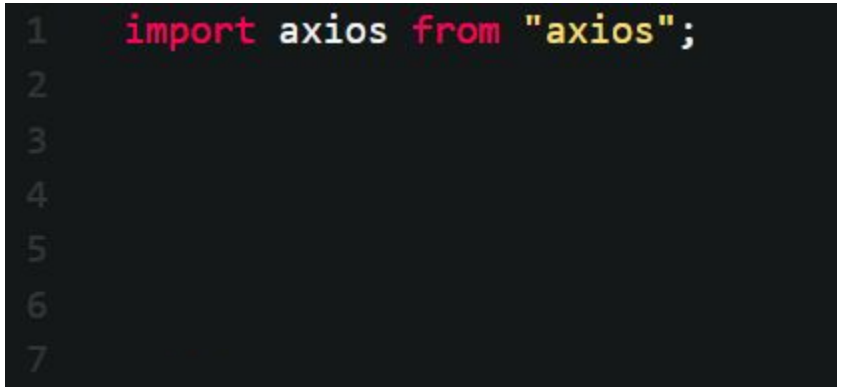
- Agregar axios al proyecto (en consola)
  - Yarn add axios
  - Npm install axios
- Importar axios
  - import axios from “axios”;
  - const axios = require(“axios”);
- Especificar endpoint a utilizar (url)
- Aclarar qué método se utilizará (get, post, put, delete).
- Agregar parámetros en caso de que sea necesario
- Realizar manejo de la respuesta.

Ejemplos con Axios

Ejemplo 1: Obtener usuarios

Ejemplos con Axios

Ejemplo 1: Obtener usuarios

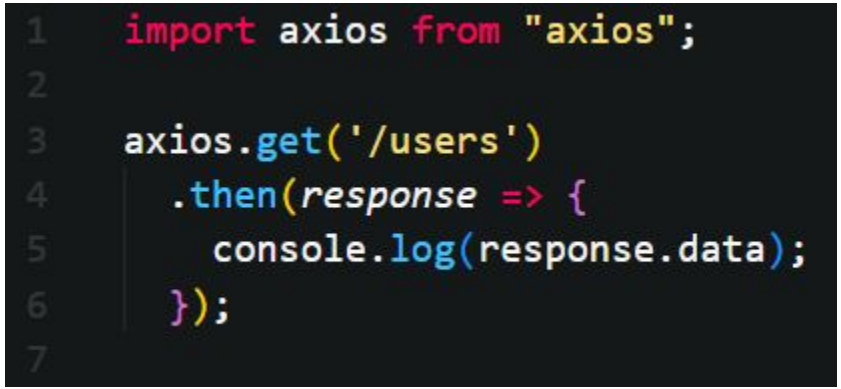


Ejemplos con Axios

Ejemplo 1: Obtener usuarios

Ejemplos con Axios

Ejemplo 1: Obtener usuarios



```
import axios from "axios";

axios.get('/users')
  .then(response => {
    console.log(response.data);
  });

```

Ejemplos con Axios

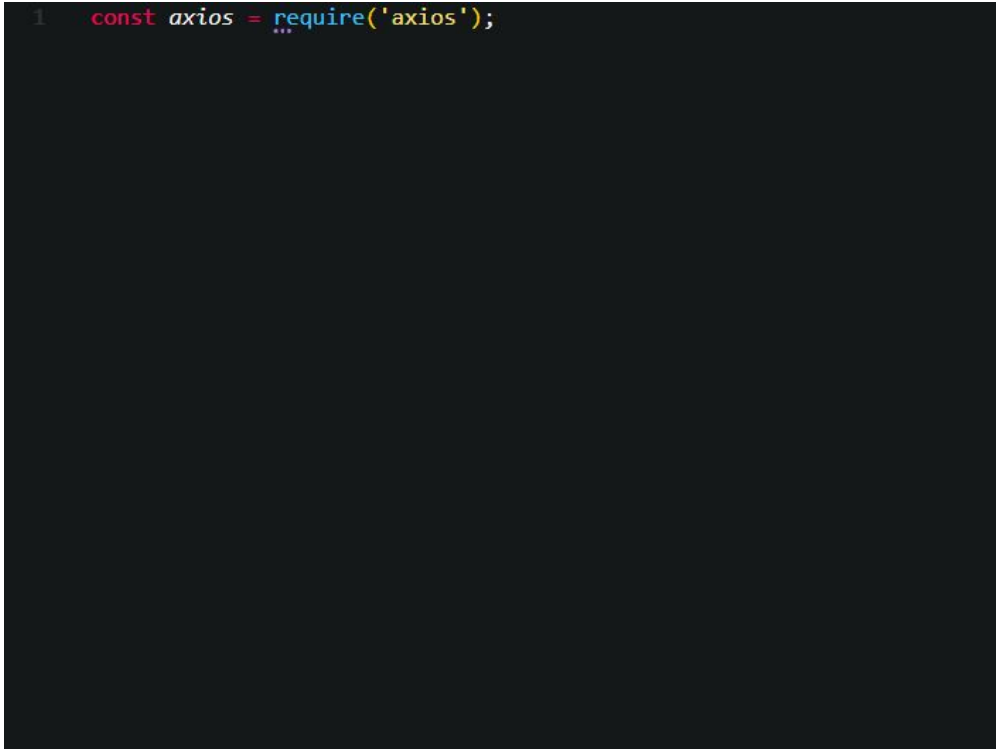
Ejemplo 1: Obtener usuarios

Output

```
[
  { id: 1, name: 'User 1' },
  { id: 2, name: 'User 2' },
  { id: 3, name: 'User 3' }
]
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario



Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
const axios = require('axios');

// Crear datos del usuario
const userData = {
  'name': 'Andrés',
  'lastName': 'Venegas',
  'age': 22
};
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
const axios = require('axios');

// Crear datos del usuario
const userData = {
  'name': 'Andrés',
  'lastName': 'Venegas',
  'age': 22
};

// Token de autenticación
const token = 'eyJhbG'
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
const axios = require('axios');

// Crear datos del usuario
const userData = {
  'name': 'Andrés',
  'lastName': 'Venegas',
  'age': 22
};

// Token de autenticación
const token = 'eyJhbG'

// Configuración de la solicitud POST
const axiosConfig = {
  'method': 'post',
  'url': 'https://ejemplo.com/api/users',
  'headers': {
    'Authorization': `Bearer ${token}`
  },
  'data': userData
};
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
const axios = require('axios');

// Crear datos del usuario
const userData = {
  'name': 'Andrés',
  'lastName': 'Venegas',
  'age': 22
};

// Token de autenticación
const token = 'eyJhbG'

// Configuración de la solicitud POST
const axiosConfig = {
  'method': 'post',
  'url': 'https://ejemplo.com/api/users',
  'headers': {
    'Authorization': `Bearer ${token}`
  },
  'data': userData
};

// Continuación ...
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
const axios = require('axios');

// Crear datos del usuario
const userData = {
  'name': 'Andrés',
  'lastName': 'Venegas',
  'age': 22
};

// Token de autenticación
const token = 'eyJhbG'

// Configuración de la solicitud POST
const axiosConfig = {
  'method': 'post',
  'url': 'https://ejemplo.com/api/users',
  'headers': {
    'Authorization': `Bearer ${token}`
  },
  'data': userData
};

// Continuación ...
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario

Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
// Continuación ...

// Hacer la solicitud POST usando Axios
axios(axiosConfig)
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
// Continuación ...

// Hacer la solicitud POST usando Axios
axios(axiosConfig)
  .then(response => {
    console.log('Respuesta del servidor:', response.data);
  })
```

Ejemplos con Axios

Ejemplo 2: Crear un usuario

```
// Continuación ...

// Hacer la solicitud POST usando Axios
axios(axiosConfig)
  .then(response => {
    console.log('Respuesta del servidor:', response.data);
  })
  .catch(error => {
    console.error('Error al hacer la solicitud:', error);
  });

```

Ejemplos con Axios

Ejemplo 2: “Crear un usuario”

Output

```
Respuesta del servidor: {
  id: 123,
  name: 'Andrés',
  lastName: 'Venegas',
  age: 22
}
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const weatherData = response.data;
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const weatherData = response.data;

    console.log(`El clima en ${city}:`);
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const weatherData = response.data;

    console.log(`El clima en ${city}:`);
    console.log(`Temperatura: ${weatherData.main.temp}°C`);
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const weatherData = response.data;

    console.log(`El clima en ${city}:`);
    console.log(`Temperatura: ${weatherData.main.temp}°C`);
    console.log(`Descripción: ${weatherData.weather[0].description}`);
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const weatherData = response.data;

    console.log(`El clima en ${city}:`);
    console.log(`Temperatura: ${weatherData.main.temp}°C`);
    console.log(`Descripción: ${weatherData.weather[0].description}`);
    console.log(`Humedad: ${weatherData.main.humidity}%`);
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

```
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const weatherData = response.data;

    console.log(`El clima en ${city}:`);
    console.log(`Temperatura: ${weatherData.main.temp}°C`);
    console.log(`Descripción: ${weatherData.weather[0].description}`);
    console.log(`Humedad: ${weatherData.main.humidity}%`);
  } catch (error) {
    console.error('Error fetching weather data:', error);
  }
};
```

Ejemplos con Axios

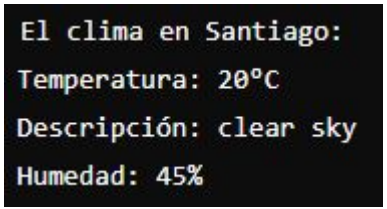
Ejemplo 3: Clima en Santiago (API OpenWeather)

```javascript
const axios = require('axios'); // O 'import axios from "axios";' en React

const fetchWeather = async (city) => {
  const apiKey = 'TU_API_KEY_AQUI';  // Coloca aquí tu API key
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric`;

  try {
    const response = await axios.get(url);
    const weatherData = response.data;

    console.log(`El clima en ${city}:`);
    console.log(`Temperatura: ${weatherData.main.temp}°C`);
    console.log(`Descripción: ${weatherData.weather[0].description}`);
    console.log(`Humedad: ${weatherData.main.humidity}%`);
  } catch (error) {
    console.error('Error fetching weather data:', error);
  }
};

// Llamar a la función con la ciudad que quieras consultar
fetchWeather('Santiago');
```

Ejemplos con Axios

Ejemplo 3: Clima en Santiago (API OpenWeather)

Output



¿Cómo se ocupa?

- Agregar axios al proyecto (en consola)
  - Yarn add axios
  - Npm install axios
- Importar axios
  - import axios from “axios”;
  - const axios = require(“axios”);
- Especificar endpoint a utilizar (url)
- Aclarar qué método se utilizará (get, post, put, delete).
- Agregar parámetros en caso de que sea necesario
- Realizar manejo de la respuesta.

Contenidos

01 API/RESTful

02 Métodos HTTP

03 Headers y body

04 Códigos de estado de respuesta HTTP

05 Axios

06 **Postman**

Postman

Postman es una herramienta popular utilizada por desarrolladores y equipos de desarrollo de API para probar, documentar, y automatizar solicitudes a APIs. Ofrece una interfaz de usuario intuitiva que permite a los usuarios crear y enviar solicitudes HTTP a servidores, inspeccionar las respuestas, y colaborar en el desarrollo de API.

POSTMAN

Postman

Nos conectaremos a una **API pública de Pokémon** y consumiremos distintos endpoints usando el método **GET**, tanto con **axios** como con **Postman.**

URL: https://pokeapi.co/, https://t2-24-2-backend.onrender.com/

REPO: Consumo de Api

AXIOS

POSTMAN