



CONSTRUYENDO UN TODO LIST

GO - REACT

INTEGRANTES

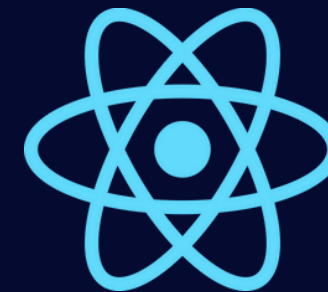
ARIEL JOSUE LOPEZ GALVEZ
TEORÍA / CONCEPTOS BASE



JOEL ALEXANDER GUZARO TZUNUN
BACKEND



SERGIO JOEL RODAS VALDEZ
FRONTEND





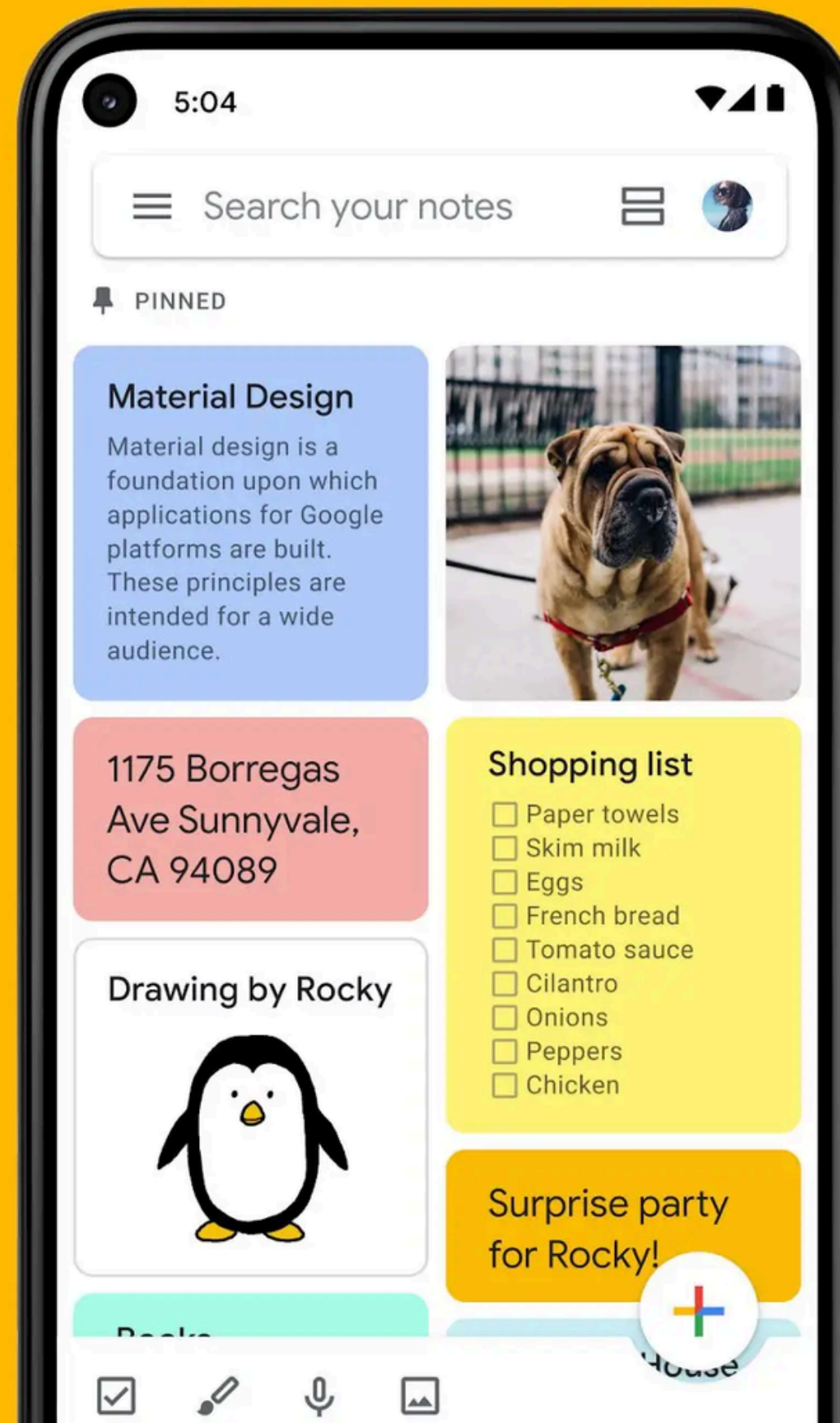
OBJETIVOS

- Comprender los conceptos básicos de cómo funciona una aplicación web.
- Identificar el rol del frontend, backend y la API en la comunicación cliente-servidor.
- Conocer los formatos y métodos principales para el intercambio de información (HTTP y JSON).
- Preparar las bases teóricas necesarias para entender la demostración práctica de backend (Go) y frontend (React).

¿QUÉ ES UN SITIO WEB?

- Una aplicación a la que ingresamos directamente desde un navegador web, sin necesidad de instalarla.
- Está formada por frontend (interfaz) + backend (lógica y datos).
- Permite interacción en tiempo real con información que vive en un servidor.





CLIENTE – SERVIDOR

CLIENTE
(FRONTEND)



PIDE LA
COMIDA

MESERO
(API)



LLEVA LA
ORDEN

COCINA
(BACKEND)



PREPARA LA
COMIDA





¿QUÉ ES EL FRONTEND?

- Es la interfaz gráfica de la sitio web.
- Lo que el usuario ve y con lo que interactúa.
- Tecnologías comunes: HTML, CSS, JavaScript, React.
- Ejemplo: botones, formularios, listas de tareas visibles.

¿QUÉ ES EL BACKEND?

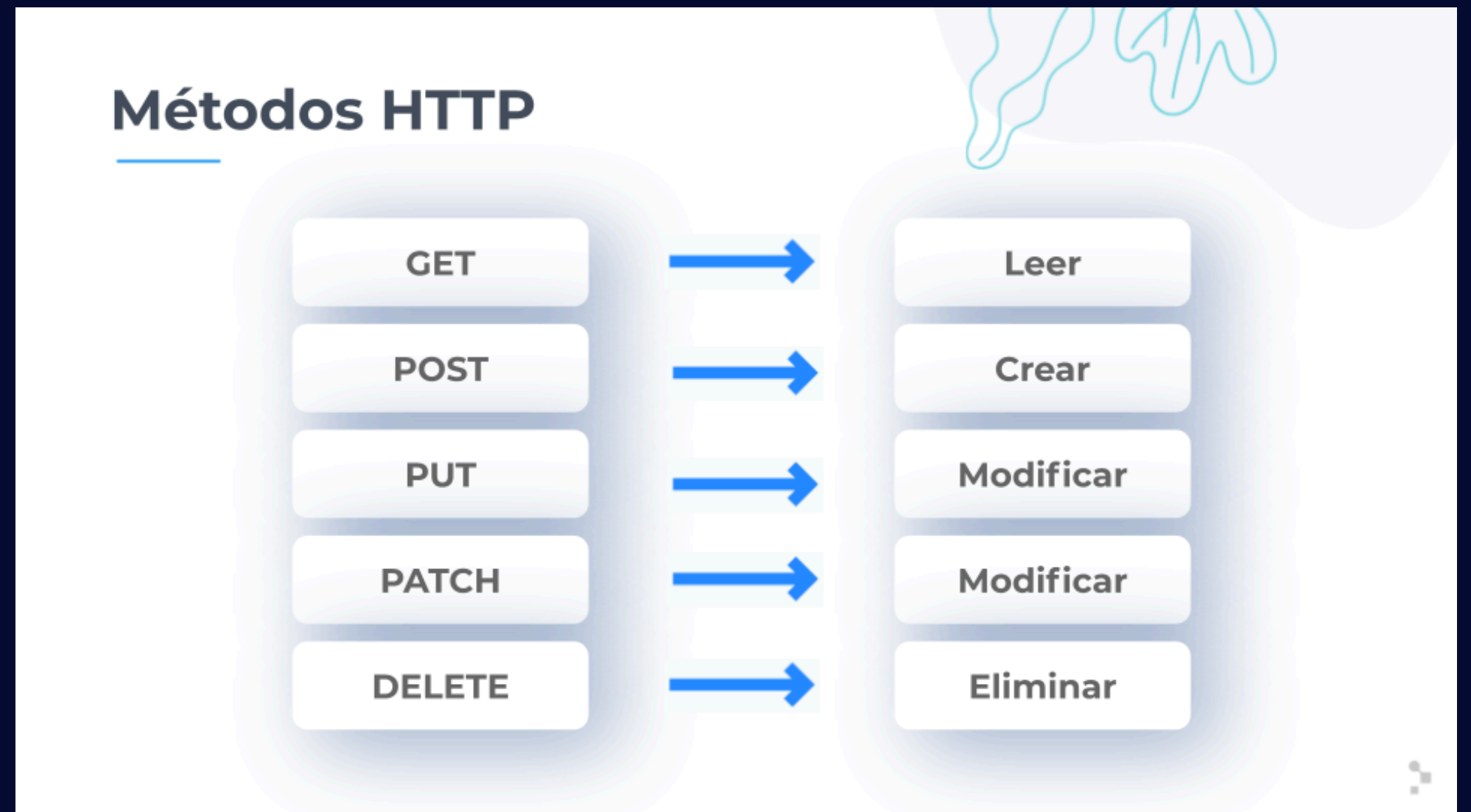
- Es la lógica y el procesamiento detrás de la app.
- Maneja datos, reglas de negocio y seguridad.
- Tecnologías comunes: Go, Node.js, bases de datos.
- Ejemplo: guardar una tarea en la base de datos, devolver datos al frontend.

¿QUÉ ES UNA API REST?

- API (Application Programming Interface): Es el puente de comunicación entre frontend y backend.
- REST: Estilo que usa endpoints y métodos HTTP para intercambiar información.
- Permite que distintas aplicaciones se entiendan entre sí sin importar el lenguaje de programación.

¿QUÉ ES HTTP?

- HTTP (HyperText Transfer Protocol): Protocolo que permite que frontend y backend se comuniquen.
- Cada acción que hace el usuario (ver, agregar, editar, borrar datos) se traduce en un método HTTP.



¿QUÉ ES UN ENDPOINT?

- URL específica en el backend donde se ofrecen datos o servicios.
- Cada endpoint hace algo concreto según el método HTTP.

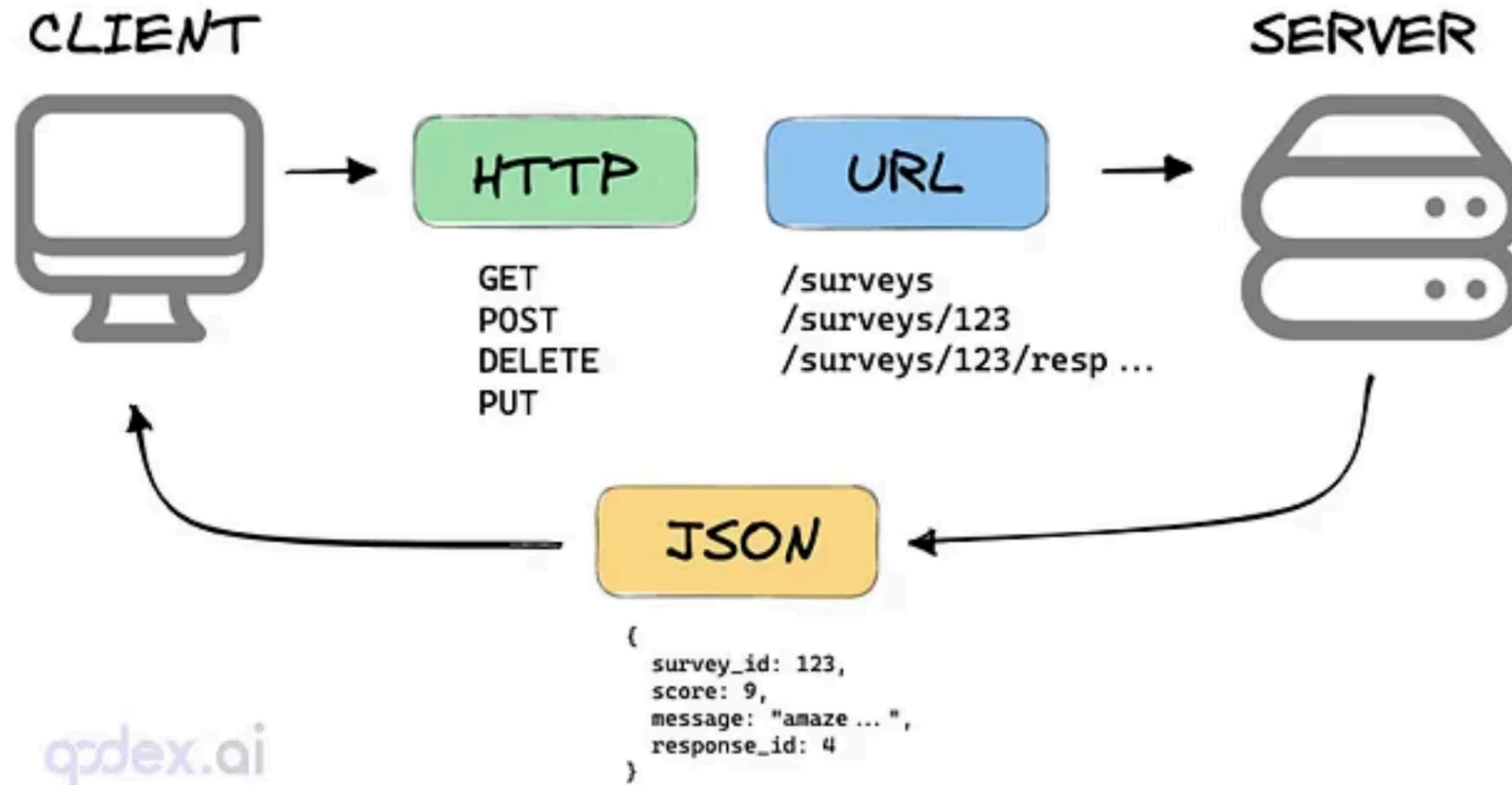
POST	/register
POST	/login
POST	/refresh
GET	/confirmEmail
POST	/resendConfirmationEmail
POST	/forgotPassword
POST	/resetPassword
POST	/manage/2fa
GET	/manage/info
POST	/manage/info

¿QUÉ ES JSON?

- Formato de intercambio de datos entre frontend y backend.
- Ligero y fácil de leer por humanos y máquinas.
- Parecido a los objetos de JavaScript.

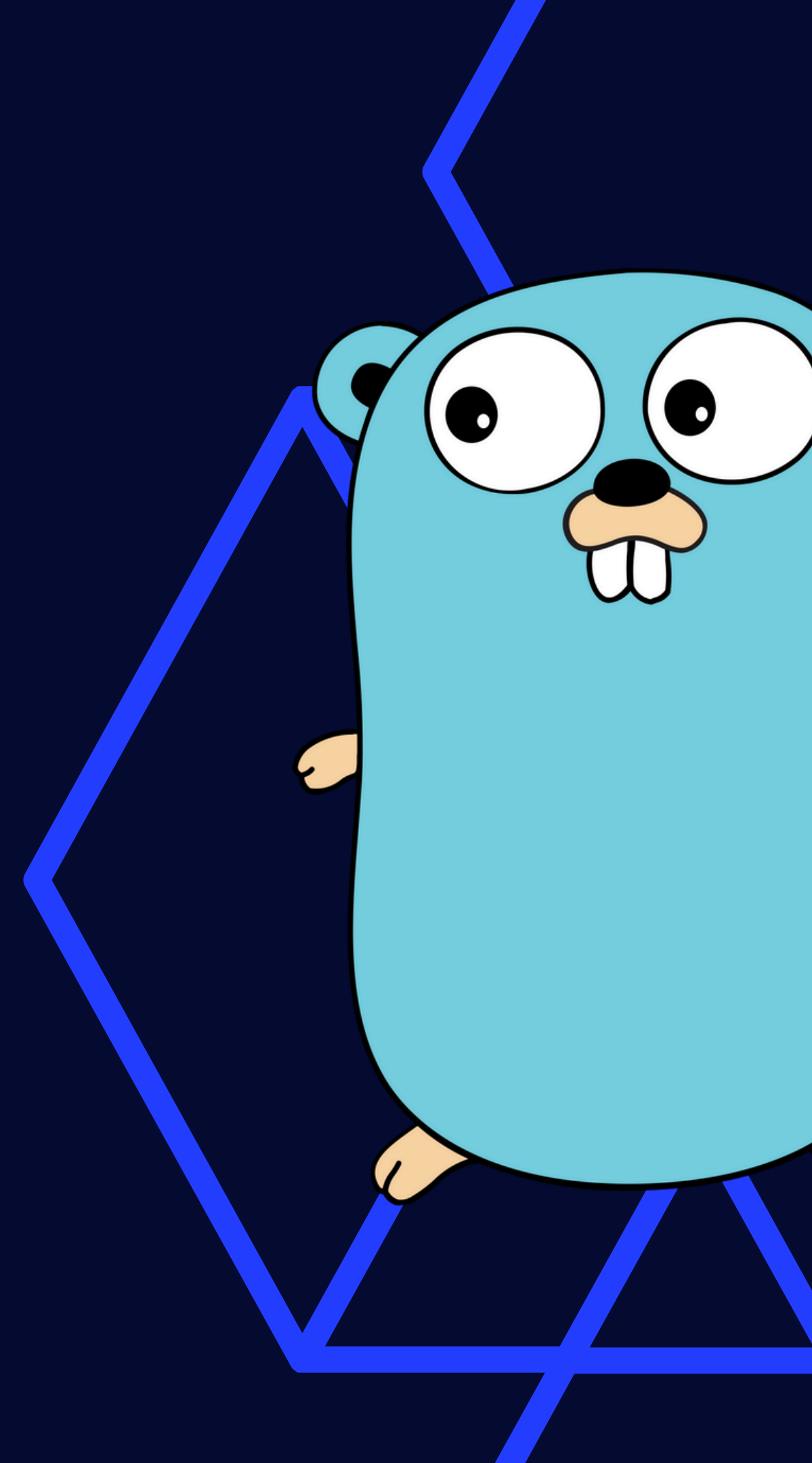
```
{  
  "libro": [  
    {  
      "id": "01",  
      "lenguaje": "Java",  
      "edición": "tercera",  
      "autor": "Herbert Schildt"  
    },  
    {  
      "id": "07",  
      "lenguaje": "C++",  
      "edición": "seguna",  
      "autor": "E.Balagurusamy"  
    }  
  ]  
}
```

FLUJO



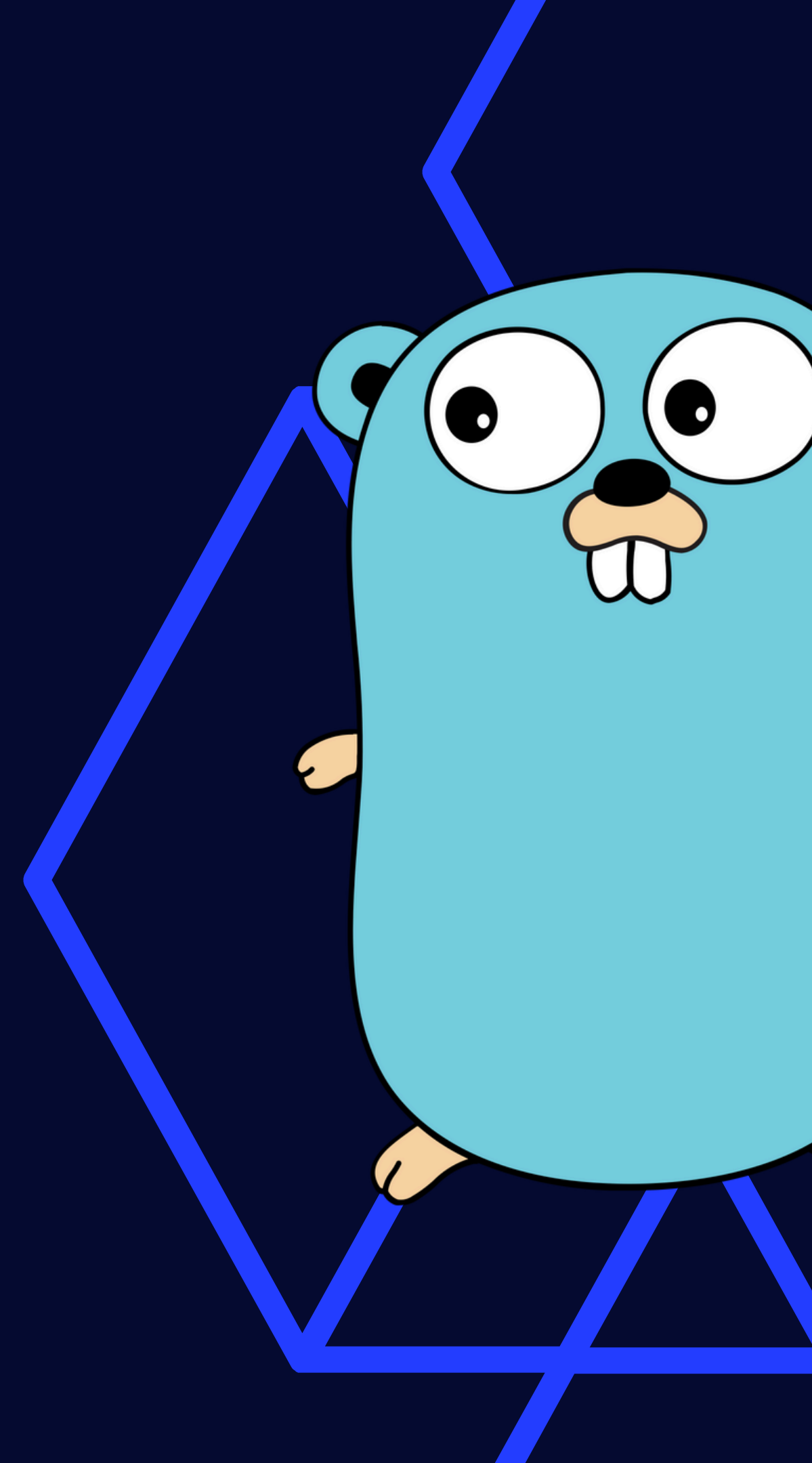
¿QUÉ ES GO?

- Lenguaje de programación creado por Google (2009).
- Rápido, eficiente y concurrente.
- Ideal para construir APIs y servidores web.
- Sintaxis sencilla y curva de aprendizaje amigable.



¿POR QUÉ USAR GO EN EL BACKEND?

- Alto rendimiento.
- Simplicidad en el código.
- Excelente manejo de concurrencia (goroutines).
- Comunidad activa y soporte de Google.



SINTAXIS BÁSICA EN GO

```
1 package main
2
3 import "fmt"
4
5 var mensaje string = "Hola Mundo"
6
7 func sumar(a int, b int) int {
8     return a + b
9 }
10
11 func main() {
12     fmt.Println(mensaje)
13     fmt.Println("2 + 3 =", sumar(2, 3))
14 }
```

Output

Hola Mundo

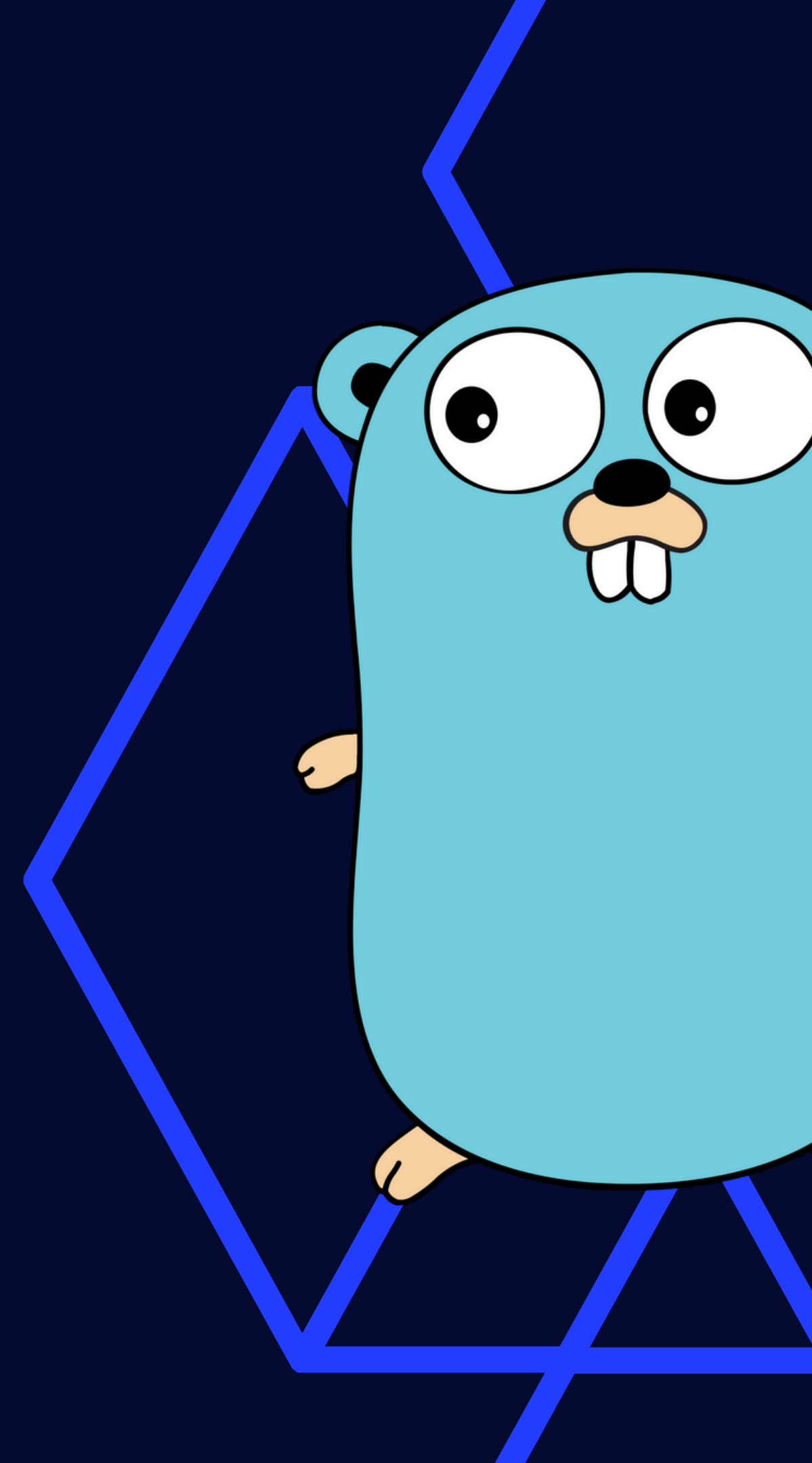
2 + 3 = 5

UN SERVIDOR SENCILLO EN GO

```
1  package main
2
3  import (
4      "fmt"
5      "net/http"
6  )
7
8  func main() {
9      http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request)
10         {
11             fmt.Fprintln(w, "Hola desde Go!")
12         })
13     http.ListenAndServe(":8080", nil)
14 }
```


HERRAMIENTAS QUE USAREMOS CON GO

- Mux: Librería para manejar rutas y endpoints de forma sencilla.
- Postman: Herramienta para probar peticiones HTTP (GET, POST, PUT, DELETE).



POST



http://example.com/fhir/Patient

Params

Authorization ●

Headers (10)

Body ●

Pre-request Script

☐ none

☐ form-data

☐ x-www-form-urlencoded

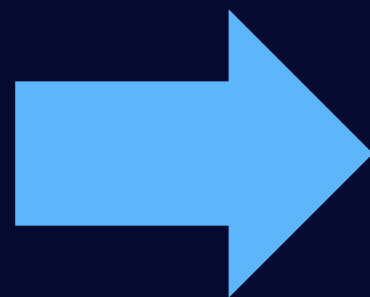
☒ raw

☐ binary

```
1  {
2    "resourceType": "Patient",
3    "identifier": [
4      {
5        "system": "http://example.com/patient-identifier",
6        "value": "123456789"
7      },
8      {
9        "system": "http://example.com/national-identifier",
10       "value": "NAT-123"
11     }
12   ]
13 }
```

¿QUÉ ES HTML?

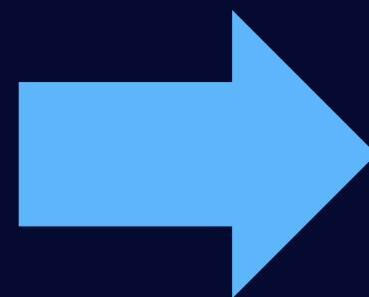
- Lenguaje que define la estructura de una página web.
- Usa etiquetas para organizar el contenido.
- Ejemplos de etiquetas: `<h1>`, `<p>`, `<button>`.



```
<h1>Mi primera página</h1>  
<p>Hola, mundo</p>
```

¿QUÉ ES CSS?

- Lenguaje que define el estilo y diseño de la página.
- Controla colores, tamaños, posiciones, fuentes.
- Se aplica sobre el HTML.



```
h1 {  
  color: blue;  
}  
  
p {  
  font-size: 20px;  
}
```

HTML + CSS JUNTOS

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo | HTML | CSS</title>
    <style>
      h1 { color: blue; }
      p { font-size: 20px; }
    </style>
  </head>
  <body>
    <h1>Mi primera página</h1>
    <p>Hola, mundo con HTML y CSS</p>
  </body>
</html>
```

Mi primera página

Hola, mundo con HTML y CSS

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo | HTML | CSS</title>
    <style>
      h1 { color: orange; }
      p {
        font-size: 50px;
        color: blue;
      }
    </style>
  </head>
  <body>
    <h1>Mi primera página</h1>
    <p>Hola, mundo con HTML y CSS</p>
  </body>
</html>
```

Mi primera página

Hola, mundo con HTML y CSS

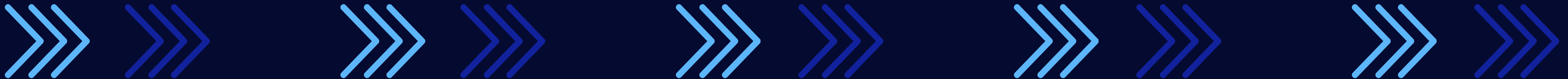
¿QUÉ ES UN FRAMEWORK?

- Conjunto de herramientas y reglas que facilitan el desarrollo.
- Te da una estructura predefinida para organizar tu código.
- Ejemplos: Angular, Django, Laravel.



¿QUÉ ES UNA LIBRERÍA?

- Conjunto de funciones que usas cuando las necesitas.
- Tú decides cómo organizar tu aplicación.
- Ejemplo: React (no es framework, es librería).

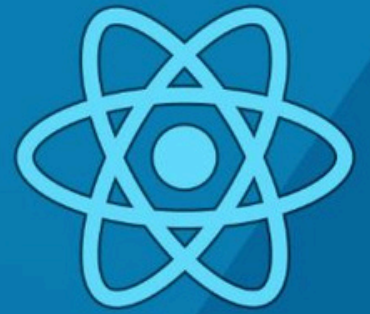


¿QUÉ ES REACT?

- Librería de JavaScript creada por Facebook.
- Sirve para construir interfaces de usuario dinámicas.
- Basada en componentes reutilizables.
- Gran comunidad y soporte.

SINTAXIS BÁSICA DE REACT.JS

React es la librería JavaScript más usada en el frontend.



```
1  import React from 'react'
2
3  const Saludo = ({ nombre }) => (
4    <span>
5      { `Hola ${nombre}` }
6    </span>
7  )
8  export default Saludo
```

Importa React en este archivo

Declaración del **componente Saludo**

El **objeto props** trae los atributos del componente.

Dentro de las llaves **se escribe JavaScript**

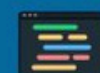
Se debe retornar **un único elemento con sintaxis JSX** (no es HTML)

Se exporta el componente.

```
1  <Saludo nombre="EDteam" />
```

Se invoca el componente pasándole las props.

Aprende React.js y domina el desarrollo frontend en:



ed.team/cursos/react

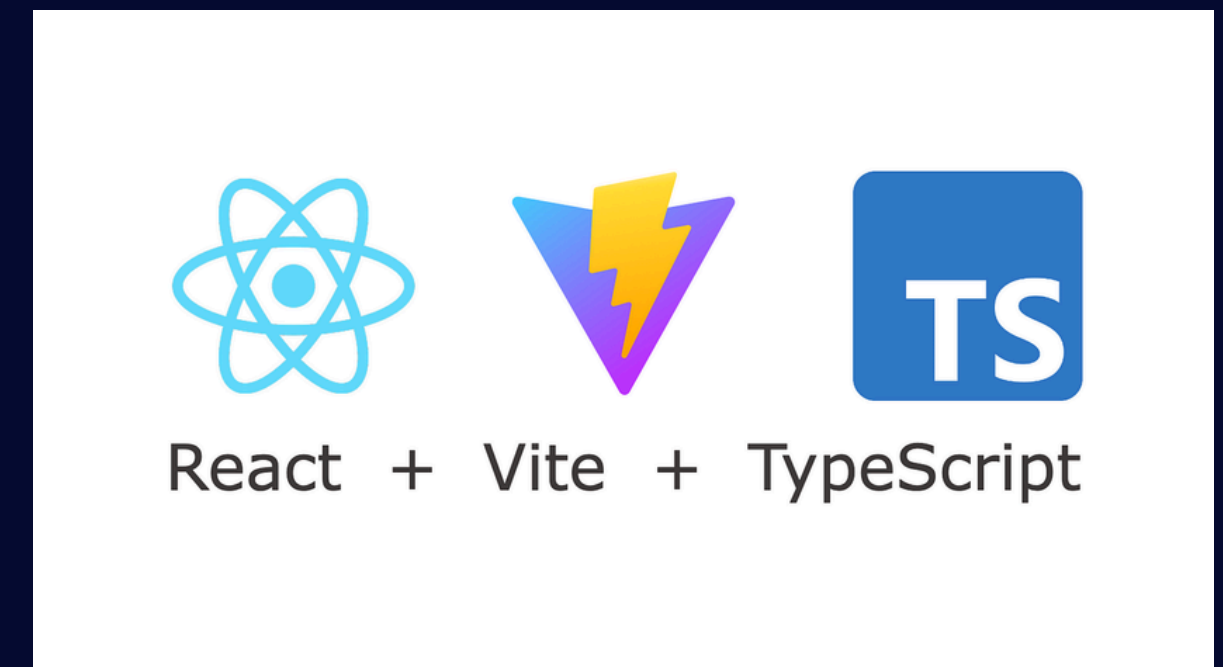


¿POR QUÉ USAR REACT EN EL FRONTEND?

- Componentes reutilizables → ahorra tiempo y esfuerzo.
- Actualizaciones rápidas → solo cambia lo que se necesita en pantalla.
- Fácil integración con APIs → ideal para consumir el backend en Go.
- Amplia comunidad → muchos recursos, librerías y soporte.

¿QUÉ ES VITE?

- Herramienta moderna de desarrollo para crear proyectos en React (y otros frameworks).
- Muy rápida: usa un servidor de desarrollo optimizado y recarga instantánea.
- Configuración mínima: en segundos tienes un proyecto listo para empezar.
- Soporta TypeScript



CREAR UN PROYECTO CON VITE

```
# Crear un nuevo proyecto con Vite  
npm create vite@latest mi-proyecto
```

```
# Entrar al proyecto  
cd mi-proyecto
```

```
# Instalar dependencias  
npm install
```


```
# Ejecutar en modo desarrollo  
npm run dev
```

HOOKS EN REACT

- **¿Qué son?** → Funciones especiales que permiten a los componentes usar características de React (estado, ciclo de vida, etc.).
- **Por qué existen** → Antes solo se podían usar estas funciones en class components, ahora también en function components.
- **Ejemplos principales:**
 - **useState** → manejar valores que cambian.
 - **useEffect** → ejecutar código cuando algo cambia o al montar el componente.
- **Ventaja** → hacen el código más simple, reutilizable y fácil de leer.

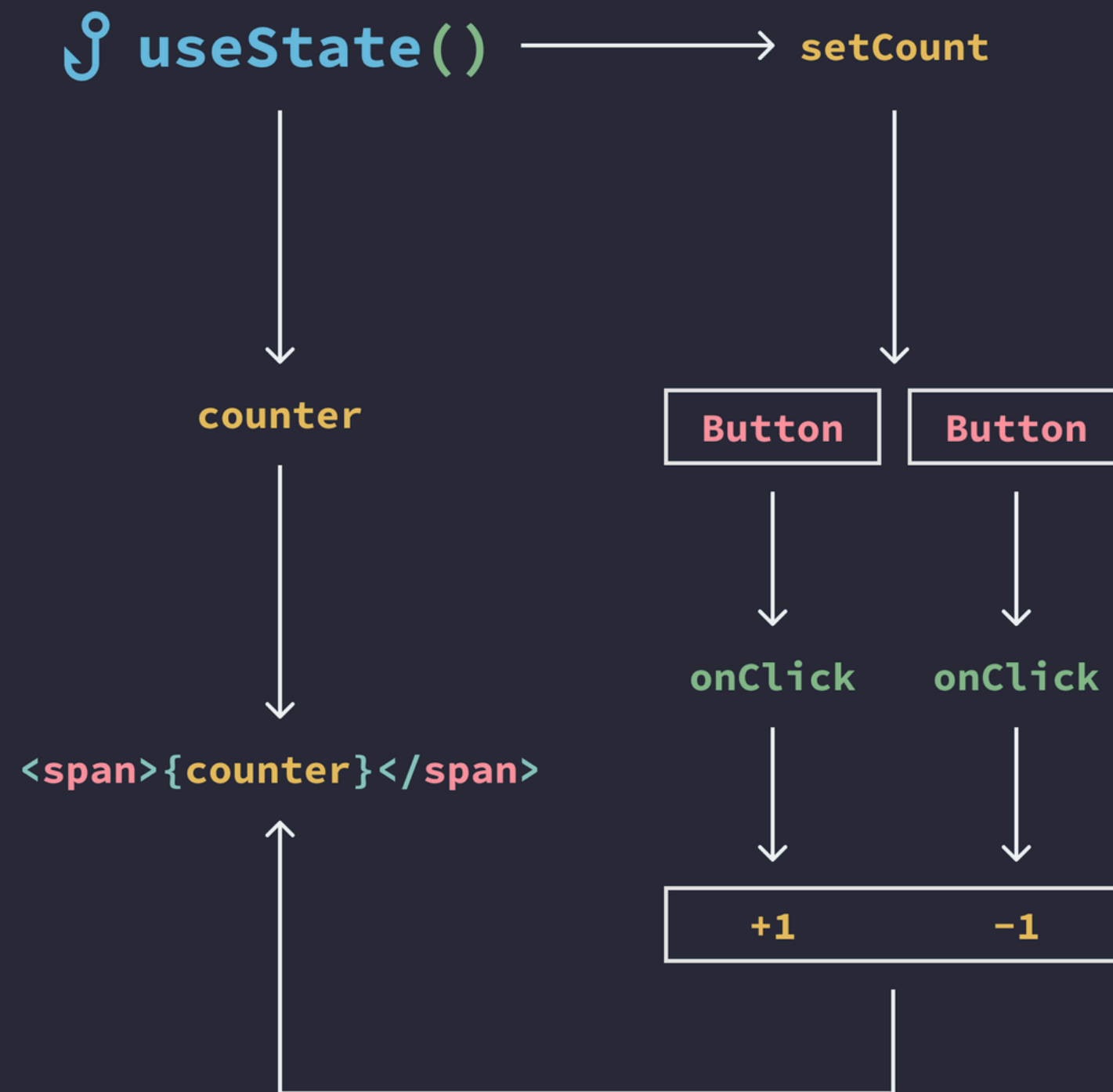
USESTATE

- **¿Qué es?** → Un Hook que permite a un componente guardar y actualizar valores.
- **Por qué es útil** → Hace que la interfaz cambie cuando cambian los datos.



```
1 import React, { useState } from 'react';  
2  
3 const App = () => {  
4  
5   const [mssg, setMssg] = useState('This is a functional component!');  
6  
7   return <h1>{mssg}</h1>  
8 }
```

EJEMPLO



USEEFFECT

- **¿Qué es?** → Un Hook que permite ejecutar *efectos secundarios* en un componente.
- **Ejemplos de efectos secundarios:**
 - Llamadas a una API.
 - Suscripciones (ej. WebSockets).
 - Actualizar el título de la página.
- Se ejecuta después de que React renderiza el componente.

```
const Counter = () => {  
  const [counter, setCounter] = useState(0);  
  
  useEffect(() => {  
    const s = setInterval(() => {  
      setCounter(c => c + 1);  
    }, 1000);  
  
    return () => clearInterval(s)  
  }, [counter]);  
  
  return (  
    <div style={{textAlign: 'center'}}>  
      <h1>Counter: {counter}</h1>  
    </div>  
  )  
};
```

FETCH API

- **¿Qué es?** → Una forma nativa de JavaScript para pedir datos a un servidor.
- **Cómo funciona** → Envía una petición HTTP y recibe una respuesta (generalmente en JSON).
- **Por qué usarla aquí** → Nos permite conectar React con el backend en Go.

```
fetch('https://example.com/api/data')  
  .then((response) => response.json())  
  .then((data) => console.log(data));
```


LIBRERÍAS QUE SUMAN AL FRONTEND



NO SIS UL CL N CO CO

- HTML y CSS → la base de toda página web.
- React → librería para construir interfaces dinámicas y modernas.
- Vite → herramienta rápida para iniciar proyectos.
- Hooks (useState, useEffect) → permiten manejar datos y efectos en componentes.
- Fetch API → conecta el frontend con el backend.
- Librerías extras → ayudan a mejorar la experiencia visual.



GRACIAS