



ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



Día, Fecha:

Jueves, 12/09/2024

Hora de inicio:

10:40 - 12:20

Introducción a la Programación y Computación 2 [P]

Denilson Florentín de León Aguilar



ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



Travesía por el Desarrollo Web: Desde el Frontend hasta las Nubes

IMPARTIDO POR: **Jonathan Valiente**

 **19** de septiembre
10:40 - 12:20 HORAS

 Google Meet



/ecysFIUSA
C



<https://dti-ecys.org>

Recordatorio Grabar

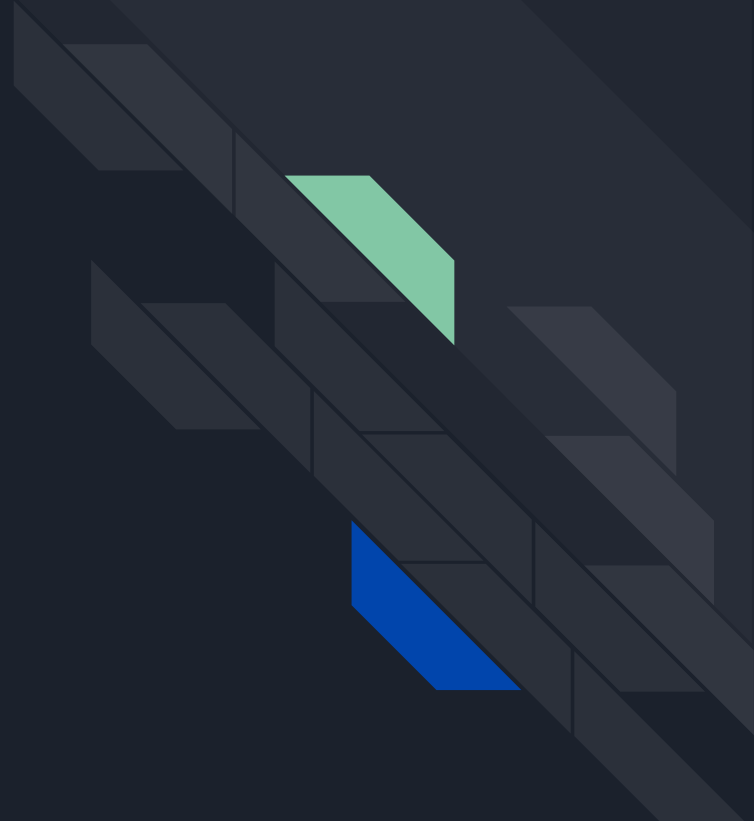




Contenido clase 8

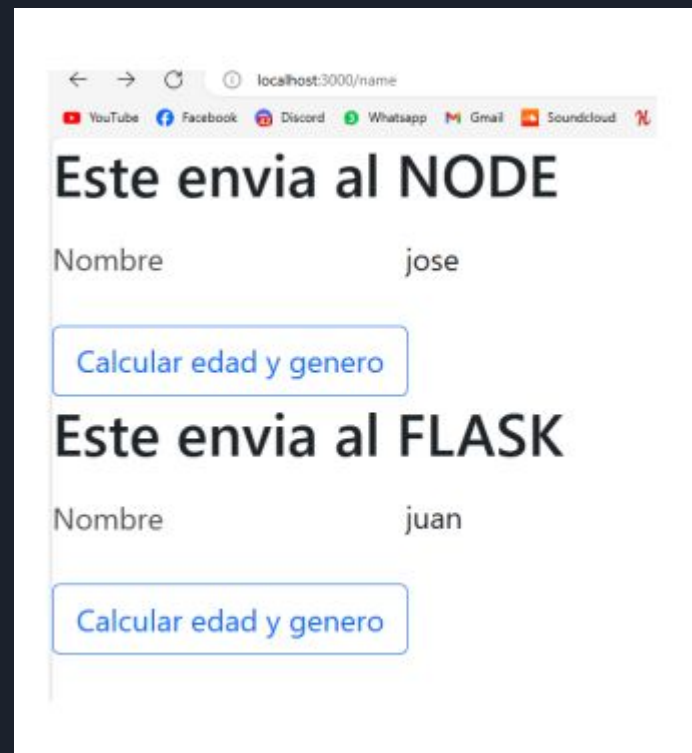
- Lectura Proyecto 2
 - Archivo de ejemplo
- ¿Qué es el Frontend?
- ¿Qué es el backend?
- ¿Qué tecnologías se pueden usar en cada una?
- Acceso a Datos web
 - Internet como origen de datos
 - Protocolo HTTP
 - Códigos de Error
 - Peticiones HTTP en Python
 - Librería en Python para realizar peticiones HTTP
- EJEMPLO:
 - Frontend de formulario.
 - Backend que genera código Graphviz

Frontend



¿Qué es el Frontend de una aplicación?

El Frontend se refiere a la parte de un sitio web o aplicación que los usuarios interactúan directamente. Incluye todo lo que ven y con lo que interactúan en sus navegadores o dispositivos, como la interfaz de usuario, el diseño, los estilos, las animaciones y las funcionalidades.



¿Qué es el Frontend de una aplicación?

El Frontend se refiere a la parte de un sitio web o aplicación que los usuarios interactúan directamente. Incluye todo lo que ven y con lo que interactúan en sus navegadores o dispositivos, como la interfaz de usuario, el diseño, los estilos, las animaciones y las funcionalidades.

```
<p>Este es otro párrafo.</p>

<!-- Párrafos -->
<p>Este es un párrafo.</p>
<p>Este es otro párrafo.</p>

<h1>Multiplicación de Dos Números</h1>

<label for="numero1">Número 1:</label>
<input type="number" id="numero1" />

<label for="numero2">Número 2:</label>
<input type="number" id="numero2" />

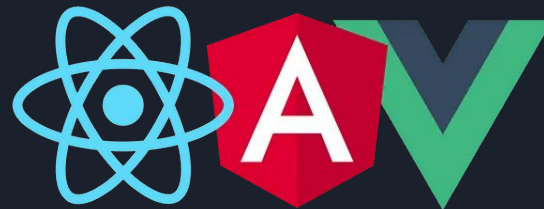
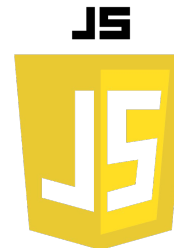
<button onclick="multiplicar()">Multiplicar</button>

<p id="resultado">Resultado: </p>

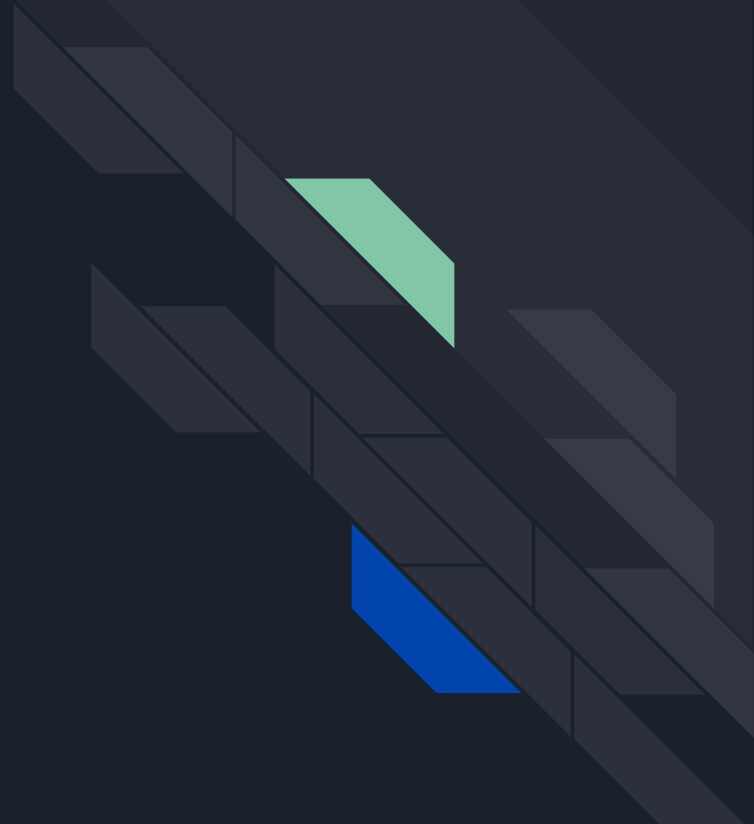
<script src="/scripts/script.js"></script>
</body>
```

¿Qué tecnologías se pueden usar en el Frontend?

En el Frontend, las tecnologías más comunes incluyen HTML, CSS y JavaScript, junto con frameworks y bibliotecas como React.js, Angular, Vue.js, Bootstrap, entre otros.

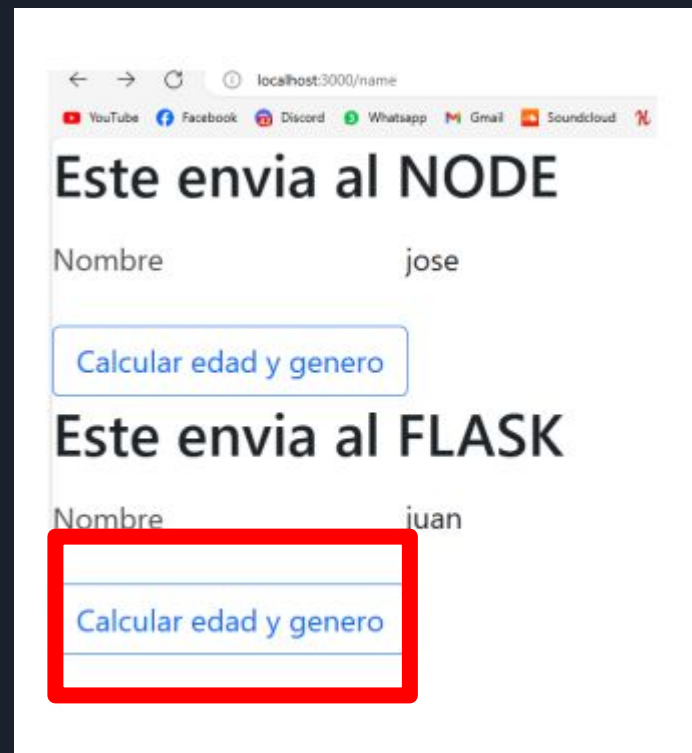


Backend



¿Qué es el Backend?

El Backend es la parte de un sitio web o aplicación que los usuarios no ven directamente, pero que es esencial para su funcionamiento. Se encarga de la lógica de negocio, la gestión de datos, la seguridad, la autenticación de usuarios y la comunicación con el servidor y la base de datos.



¿Qué es el Backend?

Del ejemplo anterior, al presionar el botón, se realizaba una conexión hacia el backend, donde realizaba una llamada a una API externa y luego retornaba un resultado.

```
@app.route('/recibirNombre', methods=['GET'])
def recibir_nombre():
    try:
        name = request.args.get('name')
        country = 'GT'

        api_url_genero = f'https://api.genderize.io?name={name}&country_id={country}'
        api_url_edad = f'https://api.agify.io?name={name}&country_id={country}'

        result_genero = requests.get(api_url_genero).json()
        result_edad = requests.get(api_url_edad).json()

        result_final = {
            'resultados_edad': result_edad,
            'resultados_genero': result_genero
        }

        return jsonify(result_final), 200
    except Exception as e:
        return f'Error: {str(e)}', 400
```

¿Qué tecnologías se pueden usar en el Backend?

En el Backend, las tecnologías varían según el lenguaje de programación utilizado. Algunas opciones populares son Python con frameworks como Django o Flask, JavaScript con Node.js y Express.js, Ruby on Rails, PHP con Laravel, Java con Spring, entre otros. Además, se suelen utilizar bases de datos como MySQL, PostgreSQL, MongoDB, entre otras, para almacenar y gestionar los datos del sitio o aplicación.

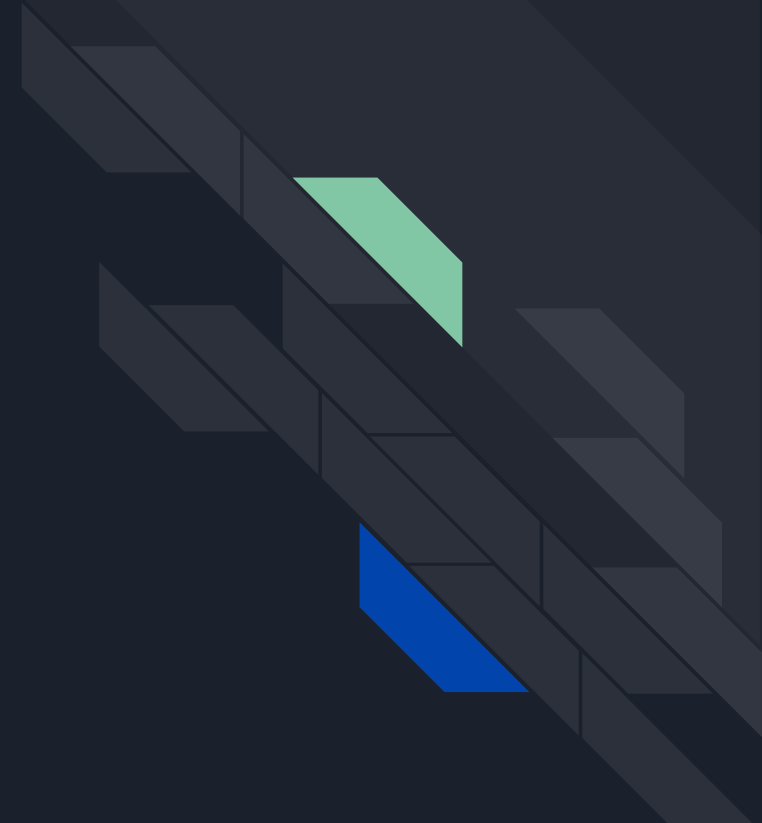


¿Qué tecnologías se pueden usar en el Backend?

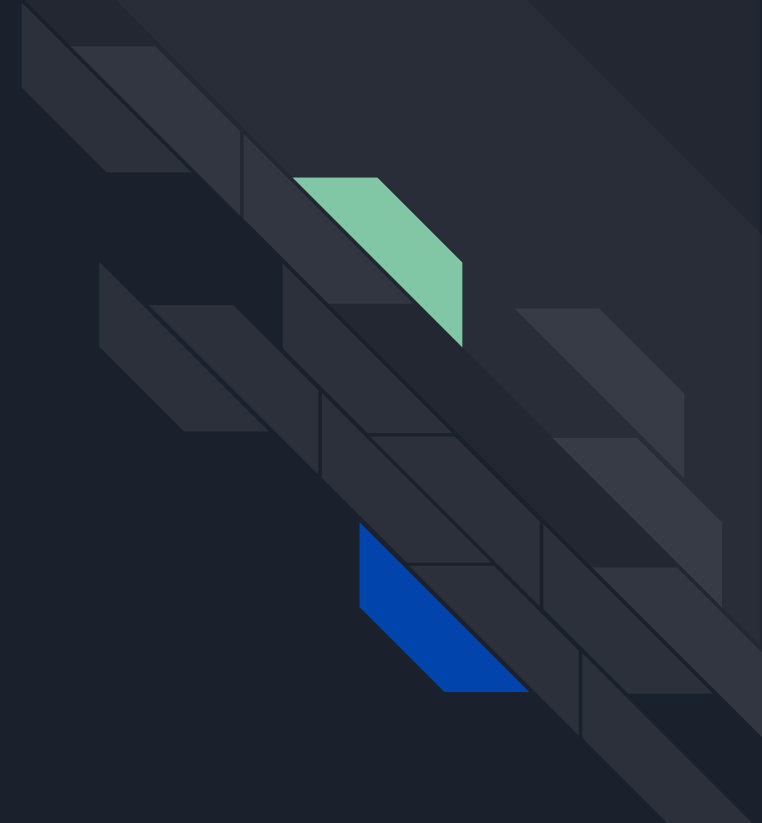
En el Backend, las tecnologías varían según el lenguaje de programación utilizado. Algunas opciones populares son Python con frameworks como Django o Flask, JavaScript con Node.js y Express.js, Ruby on Rails, PHP con Laravel, Java con Spring, entre otros. Además, se suelen utilizar bases de datos como MySQL, PostgreSQL, MongoDB, entre otras, para almacenar y gestionar los datos del sitio o aplicación.




Acceso a datos web



Conexión HTTP





¿Qué componentes base
tiene una comunicación
en la red?

HTTP TYPE
DIRECCION IP
PUERTO / PORT
METODO / METHOD
ENDPOINT
PACKAGE/PAQUETE

TCP / IP
https://developer.mozilla.org/
80
GET
en-US/docs/Web/HTTP/Methods
BODY {nombre_param = valor, pais = 'GT'}

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods?nombre_param=valor&&pais=GT

¿Qué componentes base
tiene una comunicación
en la red?

IP Address classes

Address Class	High-Order Bits	First Octet Address Range	Number of Bits in the Network Address	Number of Networks	Number of Hosts per Network
Class A	0	0-127	8	126	16,777,216
Class B	10	128-191	16	16,384	65,536
Class C	110	192-223	24	2,097,152	254
Class D	1110	224-239	28	N/A	N/A

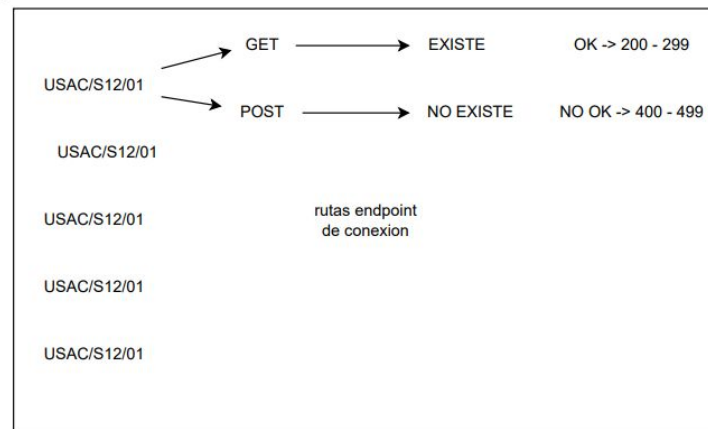
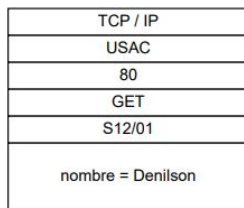
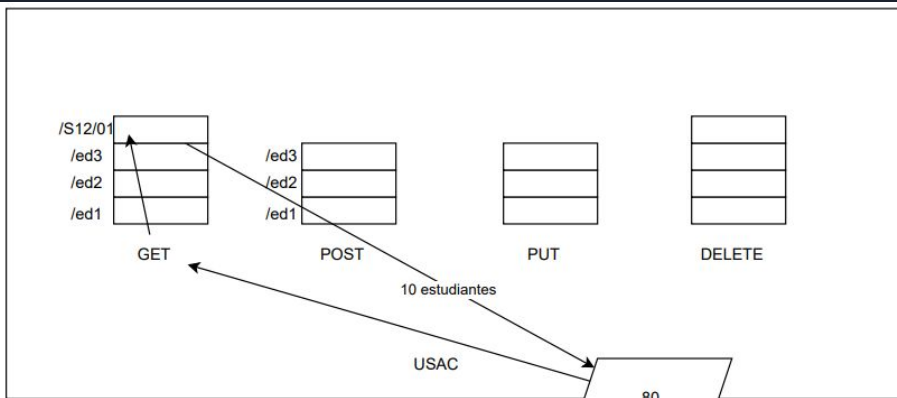
Address Class	Range of IP addresses	
Class A	1.0.0.0	127.255.255.255
Class B	128.0.0.0	191.255.255.255
Class C	192.0.0.0	223.255.255.255
Class D	224.0.0.0	239.255.255.255

¿Qué componentes base tiene una comunicación en la red?

Class of IP v4 Address

Class in IP v4 Address	Range of Values ONLY in 1 st OCTET	Classful Subnet Mask ‘/’ or CIDR or Prefix Notation	Default Subnet Mask Dotted Decimal
A	1 – 126	/8	255.0.0.0
B	128 – 191	/16	255.255.0.0
C	192 – 223	/24	255.255.255.0
D	224 – 239	Not Applicable	Not Applicable
E	240 – 255	Not Applicable	Not Applicable

¿Qué
componentes
base tiene una
comunicación
en la red?





Dirección y sockets

La combinación de una dirección URL/IPv4/IPv6 junto con un puerto se conoce como socket (IP:puerto) o simplemente dirección y puerto. Esta combinación es utilizada para identificar de manera única un servicio o aplicación en un servidor. Por ejemplo, "example.com:8080" indica que se está accediendo al servicio en el puerto 8080 del dominio example.com.

```
<h2>Este envia al FLASK</h2>  
<form action='http://localhost:5000/recibirNombre' method='GET'>  
  <div class="mb-3 row">
```

Peticiones HTTP en Python





Peticiones HTTP en Python

En Python, las solicitudes HTTP pueden realizarse utilizando bibliotecas como requests, que proporciona una interfaz sencilla y elegante para enviar y recibir solicitudes HTTP. Esta biblioteca permite realizar solicitudes HTTP GET, POST, PUT, DELETE y otras, así como manejar encabezados, cookies, autenticación, y mucho más.

Primero se importa:

```
from flask import Flask, render_template, send_from_directory, request, jsonify
import requests

app = Flask(__name__)
```

Peticiones HTTP en Python

Quedaría así el código de un Endpoint

```
@app.route('/recibirNombre', methods=['GET'])
def recibir_nombre():
    try:
        name = request.args.get('name')
        country = 'GT'

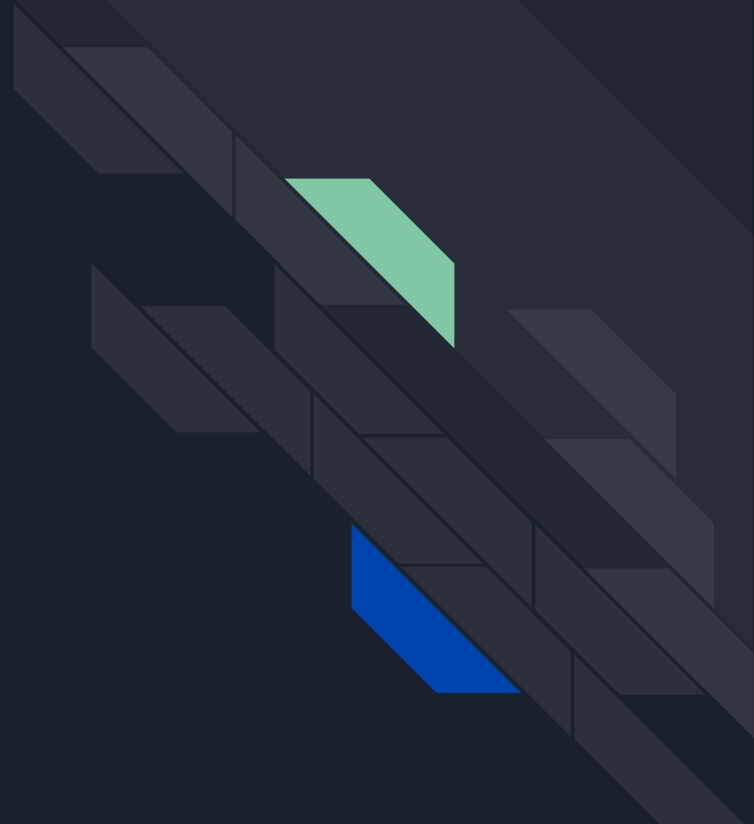
        api_url_genero = f'https://api.genderize.io?name={name}&country_id={country}'
        api_url_edad = f'https://api.agify.io?name={name}&country_id={country}'

        result_genero = requests.get(api_url_genero).json()
        result_edad = requests.get(api_url_edad).json()

        result_final = {
            'resultados_edad': result_edad,
            'resultados_genero': result_genero
        }

        return jsonify(result_final), 200
    except Exception as e:
        return f'Error: {str(e)}', 400
```

Errores HTTP en Python





Códigos de Estado HTTP

Los códigos de estado HTTP son mensajes que indican el resultado de una solicitud HTTP. Estos códigos son devueltos por el servidor web en respuesta a una solicitud del cliente. Algunos ejemplos comunes de códigos de estado incluyen 200 (OK), que indica que la solicitud se completó correctamente, y 404 (No Encontrado), que indica que el recurso solicitado no se pudo encontrar en el servidor.

Existen los más comunes, estados:

- 200: Ok
- 300: Redirección
- 400: Error del cliente
- 500: Error de servidor



Errores HTTP

Códigos de Estado 200 (Éxito)

- **200 OK:** Indica que la solicitud ha tenido éxito.
- **201 Created:** Indica que la solicitud ha tenido éxito y se ha creado un nuevo recurso como resultado.
- **204 No Content:** Indica que la solicitud ha tenido éxito pero no hay contenido para enviar en la respuesta.

Códigos de Estado 300 (Redirección)

- **301 Moved Permanently:** Indica que el recurso solicitado ha sido movido permanentemente a una nueva ubicación.
- **302 Found:** Indica que el recurso solicitado se ha encontrado temporalmente en una nueva ubicación.
- **304 Not Modified:** Indica que la versión del recurso solicitado no ha sido modificada desde la



Errores HTTP

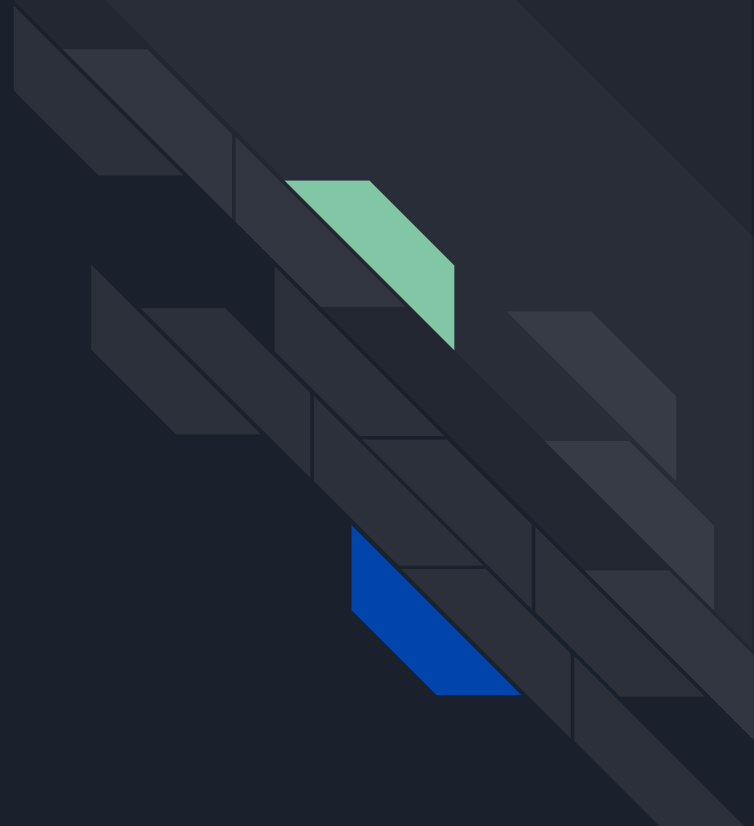
Códigos de Estado 400 (Error del Cliente)

- **400 Bad Request:** Indica que la solicitud del cliente no se pudo entender por parte del servidor debido a una sintaxis incorrecta.
- **401 Unauthorized:** Indica que se requiere autenticación para acceder al recurso solicitado.
- **404 Not Found:** Indica que el servidor no pudo encontrar el recurso solicitado.

Códigos de Estado 500 (Error del Servidor)

- **500 Internal Server Error:** Indica que se ha producido un error interno en el servidor y no se puede completar la solicitud.
- **502 Bad Gateway:** Indica que el servidor actuó como una puerta de enlace o proxy y recibió una respuesta no válida del servidor ascendente.
- **503 Service Unavailable:** Indica que el servidor no está listo para manejar la solicitud debido a una sobrecarga temporal o mantenimiento del servidor.

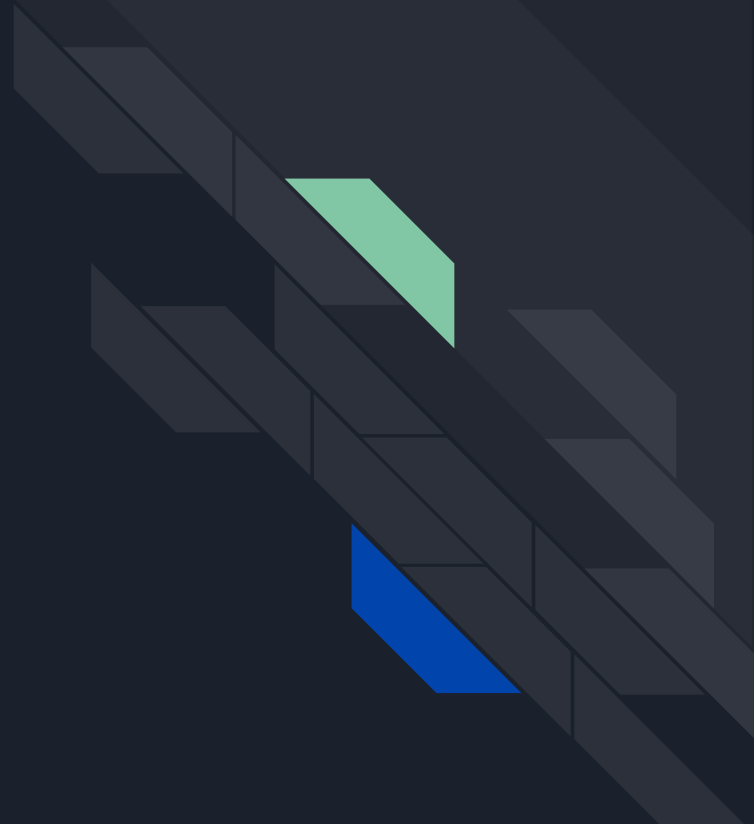
Lectura Práctica 2



Recordatorio Captura



Ejemplos



Comunicación entre servicios

<https://genderize.io/documentation#localization>

<https://agify.io/documentation#localization>

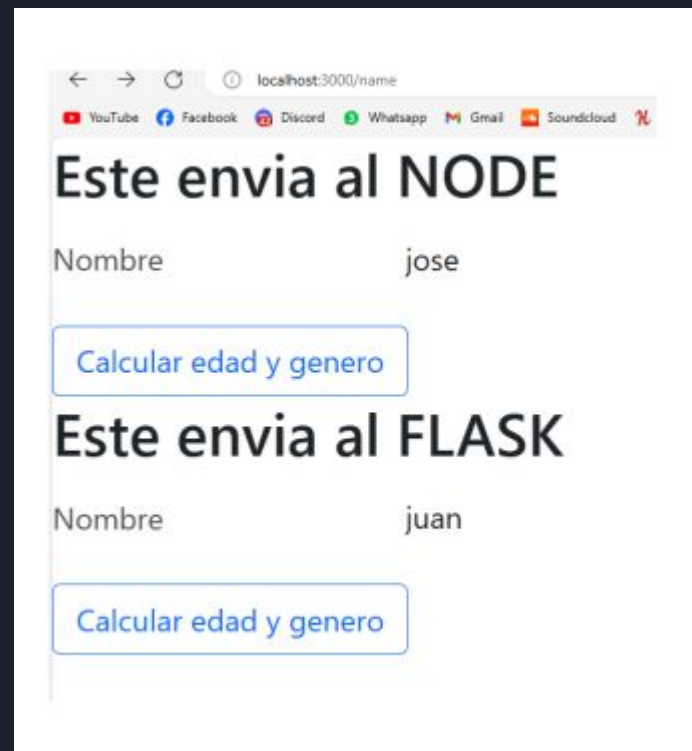
Flask y Node

¿Qué son?

¿En qué se diferencian?

¿Para qué son?

¿Cómo funcionan?



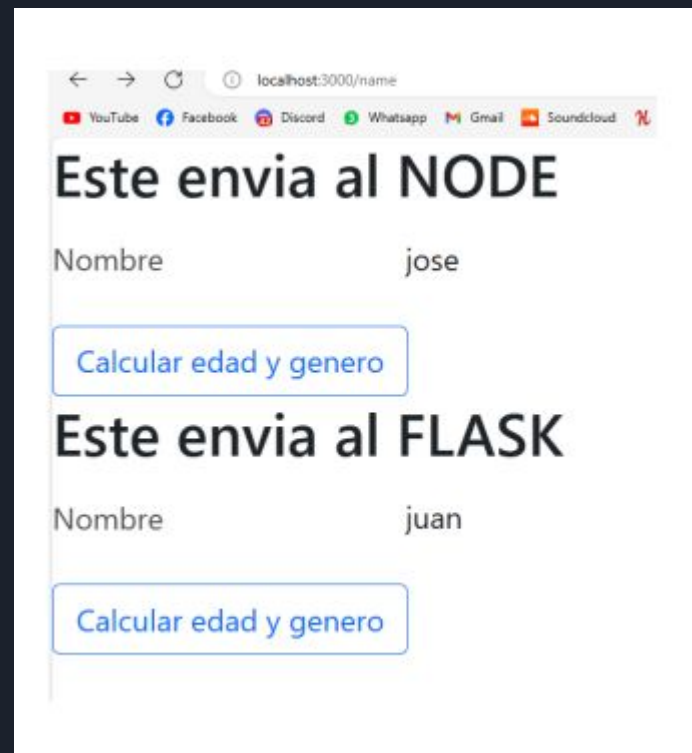
Flujo

Tener una página web principal, en este ejemplo se levantará en un servidor node.js

(localhost:3000 para la pag web)

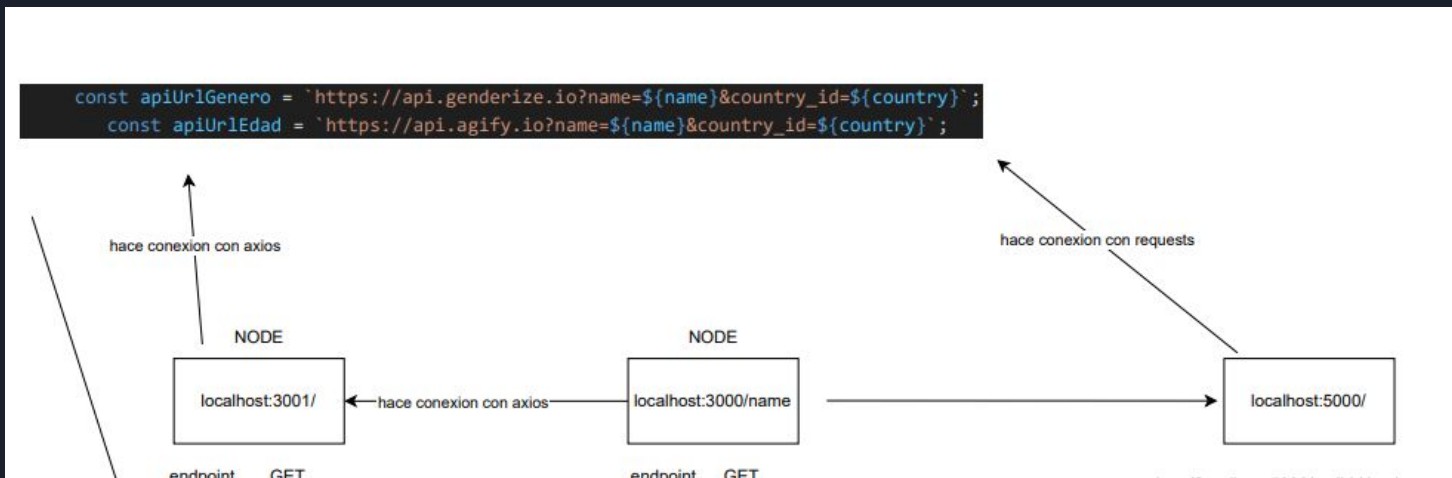
El primer botón enviará el nombre a otro servicio (servidor) en node en el puerto localhost:3001 al endpoint indicado

Mientras que el otro lo mandará a un servicio de flask al localhost:5000 a un endpoint con el mismo nombre que el anterior



Flujo

Tanto el servidor de node en localhost:3001
como el de flask en localhost:5000 van a
comunicarse con otro servicio API público



Flujo - Código Flask

importamos flask
y requests

```
pip install Flask graphviz
```

```
from flask import Flask, render_template, send_from_directory, request, jsonify
import requests
app = Flask(__name__)
@app.route('/recibirNombre', methods=['GET'])
def recibir_nombre():
    try:
        name = request.args.get('name')
        country = 'GT'

        api_url_genero = f'https://api.genderize.io?name={name}&country_id={country}'
        api_url_edad = f'https://api.agify.io?name={name}&country_id={country}'

        result_genero = requests.get(api_url_genero).json()
        result_edad = requests.get(api_url_edad).json()

        result_final = {
            'resultados_edad': result_edad,
            'resultados_genero': result_genero
        }

        return jsonify(result_final), 200
    except Exception as e:
        return f'Error: {str(e)}', 400

if __name__ == '__main__':
    app.run(debug=True)
```

Flujo - Código Node servicio

importamos axios
y express

```
const express = require('express');
const axios = require('axios');
const app = express();
const port = 3001;

// Post para recibir informacion delicada
// Delete para indicar que se debe de eliminar algo
// GET para obtener informacion
app.get("/recibirNombre", async (req, res) => {
  try{
    const name = req.query.name;
    const country = "GT";

    const apiUrlGenero = `https://api.genderize.io?name=${name}&country_id=${country}`;
    const apiUrlEdad = `https://api.agify.io?name=${name}&country_id=${country}`;

    const resultGenero = await axios.get(apiUrlGenero);
    const resultEdad = await axios.get(apiUrlEdad);

    const resultFinal = {
      'resultados_edad': resultEdad.data,
      'resultados_genero': resultGenero.data
    }

    res.status(200).json(resultFinal);
  } catch(error) {
    res.status(400).send("HAy error: " + error.message);
  }
});

app.listen(port, () => {
  console.log("Servidor en el puerto: ", port);
});
```

Flujo - Salida de las API de flask y node

retornan un json

```
1  {  
2    "resultados_edad": {  
3      "age": 48,  
4      "count": 229,  
5      "country_id": "GT",  
6      "name": "juan"  
7    },  
8    "resultados_genero": {  
9      "count": 9039,  
10     "country_id": "GT",  
11     "gender": "male",  
12     "name": "juan",  
13     "probability": 1  
14   }  
15 }
```

Flujo final

