

**Universidad de San Carlos de Guatemala**

**Facultad de Ingeniería**

**Escuela de Ciencias y Sistemas**

**Software Avanzado - Sección P**

**Ing. Marco Aldana**

**Aux. Fernando Paz**



## Práctica 1 - SOAP Y REST

Nombre	Registro Académico
Fernando Josué Tello Valiente	201800714

# Explicación de la solución

Repositorio del proyecto: [https://github.com/FerJoTello/sa\\_practica1](https://github.com/FerJoTello/sa_practica1)

La aplicación fue elaborada en Next.js versión 13.4.4 que consiste en un sistema web donde se hace consumo de API's REST y SOAP. Es posible clonar el proyecto y ejecutarlo con NodeJS versión 18.14.0.

Las API's utilizadas fueron las siguientes:

- <https://gorest.co.in/>: API Rest que permite realizar peticiones basadas en recuperación de usuarios. Esta API es pública para consumir métodos GET, sin embargo es necesario registrarse para generar un Token de Acceso para consumir métodos POST, PUT y DELETE, por lo que fue necesario registrarse.
- <http://webservicex.oorsprong.org/websamples.countryinfo/CountryInfoService.wso>: API que se basa en protocolo SOAP que permite recuperar la información de países como su nombre y su código ISO.
- <https://www.dataaccess.com/webservicesserver/NumberConversion.wso>: API SOAP que permite convertir un número a letras devolviendo el nombre del número que se provea.

Las API's basadas en SOAP fueron halladas en el siguiente enlace:

<https://documenter.getpostman.com/view/8854915/Szf26WHn>.

## Consumos REST

Para visualizar el funcionamiento de los servicios REST se elaboró una vista donde se indica las funcionalidades disponibles y qué método está utilizando.

Práctica 1 - Software Avanzado

REST SOAP

REST

Se utilizó la API <https://gorest.co.in/> para consumir 2 endpoints. 1 con un método GET para enlistar usuarios registrados, y 1 con un método POST para publicar usuarios nuevos.

### Método GET

En la siguiente table se enlistar la información recuperada del siguiente endpoint: [https://gorest.co.in/public/v2/users?page=numeroPagina&per\\_page=cantidadPorPagina](https://gorest.co.in/public/v2/users?page=numeroPagina&per_page=cantidadPorPagina), donde se indican los parámetros de la cantidad de elementos que se desean buscar por cada llamada a la API (limit) y qué conjunto de datos se desea consultar (offset).

ID	Nombre	Correo	Género	Estado
2574384	Gauraang Mishra DVM	dvm_gauraang_mish...	female	inactive
2574383	Trisha Patil	patil_trisha@crona.e...	female	active
2574382	Chandan Bhat	chandan_bhat@berg...	male	inactive
2574381	Prasanna Adiga	adiga_prasanna@ha...	male	inactive
2574380	Purushottam Sethi	sethi_purushottam@...	male	inactive
2574379	Aagam Mishra	mishra_aagam@kon...	male	inactive
2574378	Meghnad Iyengar	iyengar_meghnad@l...	male	active
2574377	Ravi Varrier	ravi_varrier@schmitt...	female	active
2574376	Swapnil Nayar Sr.	nayar_sr_swapnil@gl...	female	active
2574375	Ms. Ghanaaand Dutta	ms_dutta_ghanaana...	male	inactive

Filas por página: 10 1-10 de 100 < > >>

### Método POST

Con el siguiente formulario es posible publicar un usuario nuevo y registrarlo a través del siguiente endpoint: <https://gorest.co.in/public/v2/users> con el método POST.

Para consumir este endpoint fue necesario generar un token de acceso en el sitio <https://gorest.co.in/>. Una vez ha sido creado el token es necesario proporcionarlo en los headers de autorización para consumir el servicio.

Nombre:

Nombre

Email:

Email

Género:

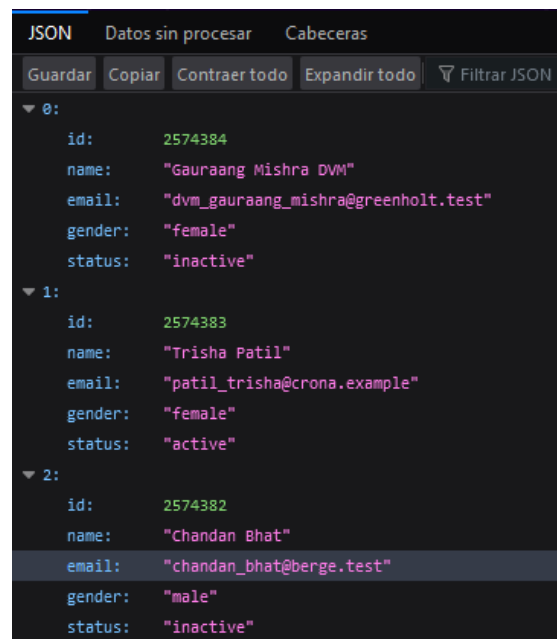
Género

Estado:

Estado

Crear usuario

Para poner en práctica el método GET del servicio REST se consume un endpoint que consulta una cantidad de usuarios indicando un paginado a través de la siguiente ruta: [https://gorest.co.in/public/v2/users?page=1per\\_page=10](https://gorest.co.in/public/v2/users?page=1per_page=10) donde los parámetros page y per\_page pueden ser modificados para consultar e implementar un paginado. La respuesta JSON de este servicio sigue el siguiente formato de ejemplo:



Estos datos son mostrados en una tabla mostrando de forma ordenada la información que devuelve el servicio.

## Método GET

En la siguiente tabla se enlista la información recuperada del siguiente endpoint: [https://gorest.co.in/public/v2/users?page=numeroPagina&per\\_page=cantidadPorPagina](https://gorest.co.in/public/v2/users?page=numeroPagina&per_page=cantidadPorPagina), donde se indican los parámetros de la cantidad de elementos que se desean buscar por cada llamada a la API (limit) y qué conjunto de datos se desea consultar (offset).

ID	Nombre	Correo	Género	Estado
2574384	Gauraang Mishra DVM	dvm_gauraang_mish...	female	inactive
2574383	Trisha Patil	patil_trisha@crona.e...	female	active
2574382	Chandan Bhat	chandan_bhat@berg...	male	inactive
2574381	Prasanna Adiga	adiga_prasanna@ha...	male	inactive
2574380	Purushottam Sethi	sethi_purushottam@...	male	inactive
2574379	Aagam Mishra	mishra_aagam@kon...	male	inactive
2574378	Meghnad Iyengar	iyengar_meghnad@l...	male	active
2574377	Ravi Varrier	ravi_varrier@schmitt...	female	active
2574376	Swapnil Nayar Sr.	nayar_sr_swapnil@gi...	female	active
2574375	Ms. Ghanaanand Dutta	ms_dutta_ghanaana...	male	inactive

En el archivo **UserTable.jsx** del código fuente se puede encontrar cuál es la forma de consumo, implementando el método *fetch* nativo de Javascript dentro de un componente React que hace una petición HTTP-GET a la ruta indicada con los parámetros detallados de “página” y “porPágina” que definen la cantidad de usuarios que se desea solicitar.

```
UserTable.jsx X
tarea1 > src > app > rest > UserTable.jsx > UserTable
You, 38 minutes ago | 1 author (You)
1 'use client';
2
3 import { useEffect, useState } from "react"
4 import DataTable from 'react-data-table-component'
5
6 export default function UserTable() {
7   const [users, setusers] = useState([])
8   const [perPage, setPerPage] = useState(10)
9   const [TotalRows, setTotalRows] = useState(100)
10  const [Loading, setLoading] = useState(0)
11
12
13  async function handleBusqueda(pagina, porPagina = null) {
14    setLoading(true);
15    const fetchResponse = await fetch(`https://gorest.co.in/public/v2/users?page=${pagina}&per_page=${porPagina || perPage}`)
16    const jsonResponse = await fetchResponse.json()
17    setusers(jsonResponse)
18    setLoading(false)
19  }
20
21  You, 38 minutes ago • add source code
22  useEffect(() => {
23    const initComponents = async () => {
24      handleBusqueda(1)
25    }
26    initComponents()
27  }, [])
```

A su vez, para aplicar el método POST se consume un endpoint que permite crear un usuario por medio de un formulario.

## Método POST

Con el siguiente formulario es posible publicar un usuario nuevo y registrarlo a través del siguiente endpoint:

<https://gorest.co.in/public/v2/users> con el método **POST**.

Para consumir este endpoint fue necesario generar un token de acceso en el sitio <https://gorest.co.in/>. Una vez ha sido creado el token es necesario proporcionarlo en los headers de autorización para consumir el servicio.

Nombre:	Email:
<input type="text" value="Fernando"/>	<input type="text" value="fernando@fakemail.com"/>
Género:	Estado:
<input type="text" value="male"/>	<input type="text" value="active"/>
<input type="button" value="Crear usuario"/>	

Al enviar la información, el servicio nos devuelve una respuesta con el detalle del usuario que se está creando, incluyendo un ID, el cual fue el que la API le asignó cuando se registró. A continuación se muestra un ejemplo al realizar una petición con *curl*.

```
ferjo@DESKTOP-SDIQ885 MINGW64 ~/Documents_unsynced/sa/tareal/tareal (master)
$ curl -i -H "Accept:application/json" -H "Content-Type:application/json" -H "Authorization: Bearer 40e9a3afedfc7e7d1d59bdf49ee824266375d2d4335f64c9a106b7418a86257" -XPOST "https://gorest.co.in/public/v2/users" -d '{"name":"Tena11 Ramakrishna", "gender":"male", "email":"tena1111111.ramakrishna@15ce.com", "status":"active"}'
HTTP/2 201
date: Tue, 06 Jun 2023 02:28:21 GMT
content-type: application/json; charset=utf-8
location: https://gorest.co.in/public/v2/users/2576530
cache-control: max-age=0, private, must-revalidate
etag: W/"363dbc3c701d09cc852c0e093483070d"
referrer-policy: strict-origin-when-cross-origin
vary: Origin
x-content-type-options: nosniff
x-download-options: noopen
x-frame-options: SAMEORIGIN
x-permitted-cross-domain-policies: none
x-ratelimit-limit: 90
x-ratelimit-remaining: 89
x-ratelimit-reset: 1
x-request-id: 65ff08bc-45b0-4abd-b463-f5d2542e5556
x-runtime: 0.083169
x-xss-protection: 0
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=FqwlqvB0rnAPz28t0rfqkxq28STfml7kpD0RjhaDy0y3zn2cu0H223pyw%2FLuy18aentV%28onBj2na5pan581p0g4qg7LzRmhf0q8sh19bsJEq8300x0edvc0WouJEpy5ok8D7j"}, {"group":"cf-nel", "max_age":604800}], "group":"cf-nel", "max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 7d2d38ac0be3b3e6-MIA
alt-avg: h3="443"; ma=80480
{"id":"2576530","name":"Tena11 Ramakrishna","email":"tena1111111.ramakrishna@15ce.com","gender":"male","status":"active"}
```

En la vista de la página web esto es desplegado de la siguiente manera después de haber llenado los campos y presionar el botón **Crear usuario**.

## Método POST

Con el siguiente formulario es posible publicar un usuario nuevo y registrarlo a través del siguiente endpoint:

<https://gorest.co.in/public/v2/users> con el método **POST**.

Para consumir este endpoint fue necesario generar un token de acceso en el sitio <https://gorest.co.in/>. Una vez ha sido creado el token es necesario proporcionarlo en los headers de autorización para consumir el servicio.

Nombre:	Email:
<input type="text" value="Nombre"/>	<input type="text" value="Email"/>
Género:	Estado:
<input type="text" value="Género"/>	<input type="text" value="Estado"/>
<input type="button" value="Crear usuario"/>	
Usuario creado con ID: 2576198	

De igual manera, la implementación del consumo del método HTTP-POST se puede encontrar en el archivo **UserForm.jsx**, donde se vuelve a usar el método fetch pero se especifica el método y los parámetros que el endpoint necesita para funcionar, como los headers de autorización y el body.

```
UserForm.jsx X
tarea1 > src > app > rest > UserForm.jsx > UserForm

3  import { useState } from 'react';
4
5  const TOKEN = process.env.NEXT_PUBLIC_REST_API_TOKEN
6
7  const UserForm = () => {
8    const [ID, setID] = useState(null)
9    const [formData, setFormData] = useState({
10      name: '',
11      gender: '',
12      email: '',
13      status: '' | You, 1 hour ago • add source code ...
14    });
15
16    async function publicarUsuario() {
17      const fetchResponse = await fetch('https://gorest.co.in/public/v2/users', {
18        method: 'POST',
19        mode: 'cors',
20        headers: {
21          "Authorization": `Bearer ${TOKEN}`,
22          "Content-Type": 'application/json'
23        },
24        body: JSON.stringify({
25          "name": formData.name,
26          "gender": formData.gender,
27          "email": formData.email,
28          "status": formData.status,
29        })
30      })
31      return await fetchResponse.json()
32    }
33  }
```

# Consumos SOAP

Para visualizar el funcionamiento de los servicios SOAP se elaboró una vista donde se indica las funcionalidades disponibles y qué método está utilizando.

Práctica 1 - Software AvanzadoRESTSOAP

## SOAP

La documentación utilizada para consumir los endpoints con el protocolo SOAP fue hallada en : <https://documenter.getpostman.com/view/8854915/Szf26WHnIc8589ab-0fc5-493c-9792-93e5d427e608>

### Método GET

En la siguiente tabla se enlista la información recuperada del siguiente endpoint: <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso> describiendo la información de países y su código ISO.

Código ISO	Nombre
AX	Aland Islands
AF	Afghanistan
AL	Albania
DZ	Algeria
AS	American Samoa
AD	Andorra
AO	Angola
AI	Anguilla
AQ	Antarctica
AG	Antigua & Barbuda

Filas por página: 101-10 de 246<>>I

### Método POST

Con el siguiente formulario se realiza una petición POST con el protocolo SOAP que devuelve el nombre (en inglés) del número que se proporcione.  
El endpoint es el siguiente: <https://www.dataaccess.com/webservices/server/NumberConversion.wso>.

Numero:

0

Consultar

Para poner en práctica el método GET del servicio SOAP se consume un endpoint que consulta un listado de países indicando su nombre y su código ISO a través del siguiente enlace: <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso>. La respuesta XML de este servicio sigue el siguiente formato de ejemplo:

Estos datos son mostrados en una tabla mostrando de forma ordenada la información que devuelve el servicio.

## Método GET

En la siguiente tabla se enlista la información recuperada del siguiente endpoint: <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso> describiendo la información de países y su código ISO.

Código ISO	Nombre
AX	Aland Islands
AF	Afghanistan
AL	Albania
DZ	Algeria
AS	American Samoa
AD	Andorra
AO	Angola
AI	Anguilla
AQ	Antarctica
AG	Antigua & Barbuda

Filas por página: 101-10 de 246<>>I

En el archivo **obtenerPaíses.js** del código fuente se puede encontrar cuál es la forma de consumo, usando la biblioteca de Node llamada *axios* que permite hacer peticiones y procesar la respuesta de forma práctica.

```
obtenerPaíses.js X
tarea1 > src > app > soap > obtenerPaíses.js > obtenerPaíses
You, 2 hours ago | 1 author (You)
1 import axios from 'axios';
2 import { parseStringPromise } from 'xml2js';
3
4 const url = 'http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso'; // URL del servicio SOAP
5
6 const xmlData = `<?xml version="1.0" encoding="utf-8"?>
7 <soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
8   <soap12:Body>
9     <ListOfCountryNamesByName xmlns="http://www.oorsprong.org/websamples.countryinfo">
10       </ListOfCountryNamesByName>
11     </soap12:Body>
12   </soap12:Envelope>`;
13
14 export default async function obtenerPaíses() {
15   try {
16     const { data } = await axios.post(url, xmlData, {
17       headers: {
18         'Content-Type': 'text/xml; charset=utf-8'
19       }
20     });
21     const parsedData = await parseStringPromise(data);
22     // countries es un arreglo de objetos que representa la lista de países
23     const countries = parsedData['soap:Envelope']['soap:Body'][0]['m:ListOfCountryNamesByNameResponse'][0]['m:ListOfCountryNamesByNameResult'][0]['m:tCountryCodeAndName'];
24     return countries;
25   } catch (error) {
26     console.log(error);
27     return null;
28   }
29 }
```

A su vez, para aplicar el método POST se consume un endpoint que permite convertir un número a letras devolviendo el nombre del número en palabras (en inglés).

## Método POST

Con el siguiente formulario se realiza una petición POST con el protocolo SOAP que devuelve el nombre (en inglés) del número que se proporcione.

El endpoint es el siguiente: <https://www.dataaccess.com/webservicesserver/NumberConversion.wso>.

Numero:

Consultar

Después de proporcionar un número y presionar el botón **Consultar** se despliega la respuesta del servicio indicando el nombre del número en inglés.



## Método POST

Con el siguiente formulario se realiza una petición POST con el protocolo SOAP que devuelve el nombre (en inglés) del número que se proporcione.

El endpoint es el siguiente: <https://www.dataaccess.com/webservicesserver/NumberConversion.wso>.

Numero:

Consultar

Respuesta en palabras: three thousand four hundred and fifty eight

De la misma manera, la implementación del consumo del método SOAP-POST se puede encontrar en el archivo **api/soap/route.js**, dónde Next funciona como un backend y realiza la petición SOAP resolviendo así los problemas de CORS que se presentan al consumir los endpoints desde un cliente web (navegador). Después de recuperar la información de la petición con axios nuevamente, se procesa para acceder al dato de la conversión del número y queda disponible para consultas desde el cliente.

```
route.js x
tarea1 > src > app > api > soap > route.js > GET > parsedData
You, 2 hours ago | 1 author (You)
1 import axios from 'axios';
2 import { NextResponse } from "next/server";
3 import { parseStringPromise } from 'xml2js';
4
5 const API_URL = 'https://www.dataaccess.com/webservicesserver/NumberConversion.wso'; // URL del servicio SOAP
6
7 export async function GET(request) {
8   try {
9     const { searchParams } = new URL(request.url);
10    const numero = searchParams.get('numero');
11
12    const xmlData = `<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
13      <soap:Body>
14        <NumberToWords xmlns="http://www.dataaccess.com/webservicesserver/">
15          <ubiNum>${numero}</ubiNum>
16        </NumberToWords>
17      </soap:Body>
18    </soap:Envelope>`;
19
20    const { data } = await axios.post(API_URL, xmlData, {
21      headers: {
22        'Content-Type': 'text/xml; charset=utf-8'
23      }
24    });
25    const parsedData = await parseStringPromise(data)
26
27    return NextResponse.json({ numeroAPalabras: parsedData['soap:Envelope']['soap:Body'][0]['m:NumberToWordsResponse'][0]['m:NumberToWordsResult'][0] });
28  } catch (error) {
29    console.log(error)
30    return NextResponse.json({ error: "Se dio un problema no controlado" }, { status: 400 });
31  }
32 }
33 }
```