



UNIVERSIDAD POLITECNICA
INTERDISCIPLINARIA EN INGENIERIA
Y TECNOLOGIAS AVANZADAS.



PRACTICA 1

Materia :Aplicaciones distribuidas

Profesor : Noe Sierra Romero

TITULO :

Servicios ResBul usando NodeJS

FECHA : 18 de Febrero del 2026. Hecho por :

Flores Hernandez Lluvia Nahivy
Boleta : 2022640090

Crear una serie de servicios web usando un servidor NodeJS

Inciso i. mascaracteres

The screenshot shows the Postman interface. At the top, it says "http://localhost:3000/mascaracteres". Below that, a POST request is being made to the same URL. The "Body" tab is selected, showing the following JSON payload:

```
1 {  
2   "cadena1": "Hola mundo",  
3   "cadena2": "anita"  
4 }
```

At the bottom, the response is shown as a 200 OK status with a response time of 5 ms and a body size of 261 B. The response body is:

```
{ } JSON ▾ > Preview | Visualize | ▾  
1 {  
2   "resultado": "Hola mundo"  
3 }
```

Se valida el código que regresa la cadena de caracteres mas larga o con mas caracteres.

Inciso ii. menoscaracteres

The screenshot shows the Postman interface. At the top, it says "Overview" and "POST http://localhost:3000/i". Below that, the URL "http://localhost:3000/menoscaracteres" is entered. The "Body" tab is selected, showing the following JSON payload:

```
1 {  
2   "cadena1": "Hola mundo",  
3   "cadena2": "anita"  
4 }
```

Below the body, the response is shown as "200 OK" with a green status bar. The response body is also JSON:

```
{ } JSON ▾ ▶ Preview ▾ Visualize ▾  
1 {  
2   "resultado": "anita"  
3 }
```

En este inciso se valida que regrese la cadena de menos valor o con menos caracteres.

Inciso iii. numcaracteres

The screenshot shows a Postman interface with the following details:

Request URL: `http://localhost:3000/numcaracteres`

Method: POST

Body (JSON):

```
1 {
2   |
3   "cadena": "Anita lava la tina"
4 }
```

Response: 200 OK

Body (JSON):

```
1 {
2   |
3   "resultado": 18
4 }
```

Below the main window, there is a navigation bar with icons for Cloud View, Console, Terminal, Runner, Vault, and others.

Se regresara el numero total de caracteres que hay en la oración.

Inciso iv. palindroma

The screenshot shows a Postman interface for a POST request to `http://localhost:3000/palindroma`. The request body is a JSON object with a single key-value pair: `"cadena": "Anita lava la tina"`. The response status is `200 OK`, with a response body showing a JSON object with a key `"resultado": true`.

```
HTTP http://localhost:3000/palindroma
Save Share
POST http://localhost:3000/palindroma Send
Docs Params Auth Headers (9) Body Scripts Tests Settings Cookies
raw JSON Schema Beautify
1 {
2   "cadena": "Anita lava la tina"
3 }
4 }
```

Body 200 OK 8 ms 253 B

```
[] JSON Preview Visualize
1 {
2   "resultado": true
3 }
```

Cloud View Console Terminal Runner Vault ?

Imprime true si es una palabra palindroma o false si no lo es.

Inciso v. concat

The screenshot shows the Postman application interface. At the top, it says "Overview" and "POST http://localhost:3000/concat". Below that, the URL "http://localhost:3000/concat" is entered in the "Send" button's input field. The "Body" tab is selected, showing the following JSON payload:

```
1 {  
2   "cadena1": "Hola mundo",  
3   "cadena2": "anita"  
4 }
```

Below the body, the "Headers" section shows "Content-Type: application/json". The "Test Results" section shows a successful response with status code 200 OK, 5 ms duration, and 266 B size. The response body is:

```
{ } JSON ▾ ▶ Preview ▾ Visualize ▾  
1 {  
2   "resultado": "Hola mundoanita"  
3 }
```

Regresa una oración, uniendo la cadena 1 y 2 de información.

Inciso vi.applysha256

The screenshot shows the Postman interface. At the top, it says "Overview" and "POST http://localhost:3000/i". Below that, the URL "http://localhost:3000/applysha256" is entered. The "Body" tab is selected, showing the following JSON payload:

```
1 {  
2   "cadena": "hola mundo"  
3 }  
4 
```

Below the body, the "Test Results" section shows a successful response:

200 OK • 6 ms • 341 B • [View Response](#)

The response body is also shown in JSON format:

```
{ } JSON ▾ ▶ Preview ▾ Visualize ▾  
1 {  
2   "original": "hola mundo",  
3   "encriptada": "0b894166d3336435c800bea36ff21b29eaa801a52f584c006c49289a0dcf6e2f"  
4 }  
5 
```

Para poder codificar una palabra fue necesario agregar la libreria de "crypto", una vez ingresara la palabra regresara codificada.

Inciso vii. verifysha256

The screenshot shows the Postman interface. At the top, it says "Overview" and "POST http://localhost:3000/". Below that, the URL "http://localhost:3000/verifysha256" is entered. The "Body" tab is selected, showing the following JSON payload:

```
1 {  
2   "cadenaNormal": "secreto123",  
3   "cadenaEncriptada": "1960d37651016928e7e2528172c7225c52c0350414df302914041d5a7122112e"  
4 }
```

At the bottom, the response is shown as "200 OK" with a status of "17 ms" and "254 B". The response body is:

```
{ } JSON ▾ ▶ Preview ▶ Visualize ▾
```

```
1 {  
2   "resultado": false  
3 }
```

Para verifysha256 se vuelve lo contrario a applysha256, se ingresa una para encriptada y devolvera la palabra original.

Conclusión

Los servicios web desde hace ya un par de años han sido cruciales para nuestra comunicación y como todo lo demás tambien sigue evolucionando, aqui solo se muestran ejemplos simples, pero la dinamica es la misma para problemas mucho mas complejos.