

Express

BACKEND EXPRESS JS

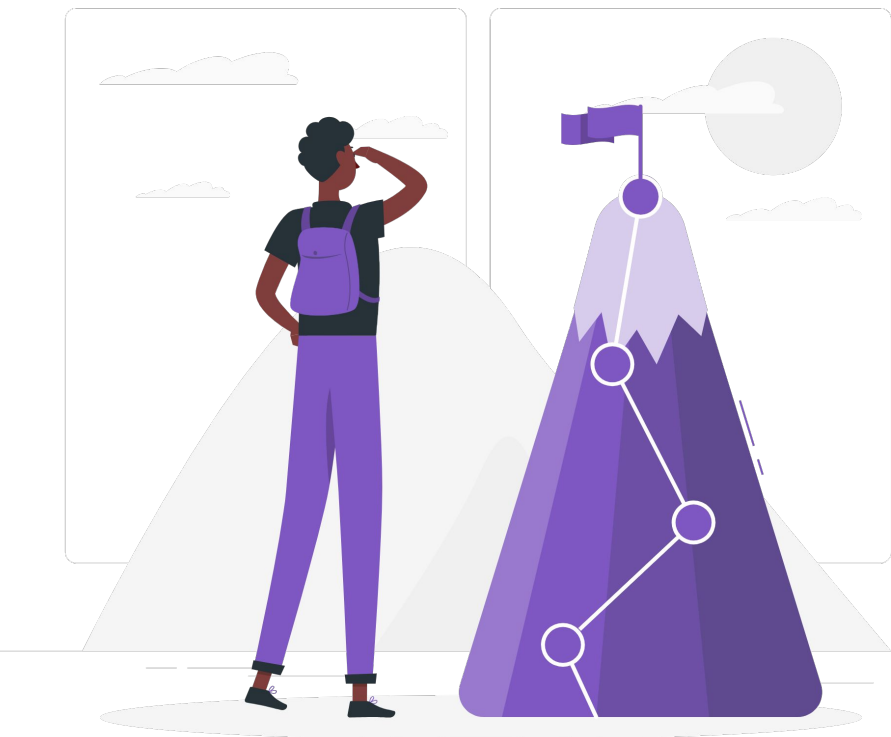
DEV.FX
DESARROLLAMOS(PERSONAS);

Elaborado por: César Guerra

www.cesarquerra.mx



Buy me a coffee



Objetivos de la Sesión

- Aprender a comprobar que tenemos los requerimientos para ejecutar Express JS.
- Codificar nuestro primer servidor web con Express JS.
- Entender la estructura de un servidor web con Express JS.
- Aprender a manejar rutas de API REST en Express.

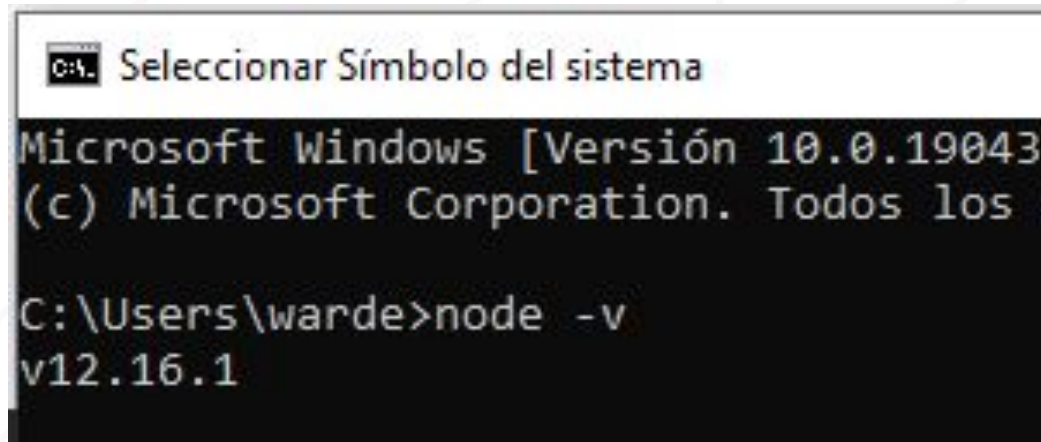
Express

PRERREQUISITOS

DEV.F.
DESARROLLAMOS(PERSONAS);

dev

Comprobar Instalación



```
C:\> Seleccionar Símbolo del sistema

Microsoft Windows [Versión 10.0.19043]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\warde>node -v
v12.16.1
```

Ir a una línea de comandos, y ejecutar el comando:

node -v

Deberemos poder ver la versión de node.js que tenemos instalada.

En caso de no tenerlo... Descarga de Node.js

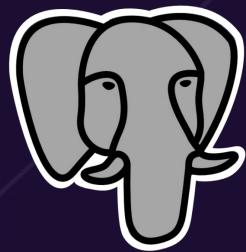


Descargar
la última
versión que
tenga LTS

<https://nodejs.org>

¡ Vamos al Código !





PostgreSQL

Ejercicios #5

APIs con Express

DEV.F.
DESARROLLAMOS(PERSONAS);

Ejercicios Intro a Express (1)

1. **Agrega un endpoint `/api/` que responda a una petición de tipo GET con un código de estado 200 y el siguiente json:**

```
{ 'mensaje': 'hola mundo' }
```
2. **Agrega un endpoint `/api/suma` que responda a una petición de tipo GET con la suma de dos números que reciba mediante las queries `num1` y `num2`. El servidor debe responder con un código de estado 200 y un json como el siguiente:**

```
{ 'resultado': 7 }
```
3. **Agrega un endpoint `/api/usuario/` que responda a una petición de tipo GET con el nombre que sea recibido a través de params. El servidor debe responder con un código de estado 200 y un json como este:**

```
{ 'usuario': 'cesar' }
```
4. **Agrega un endpoint `/api/swapi` que responda a una petición de tipo GET con el personaje solicitado de <https://swapi.dev/>. El cliente debe mandar el número de personaje mediante params. La respuesta del servidor debe lucir algo así**

```
{ 'personaje': {  
  'name': 'Luke Skywalker',  
  ...,  
}}
```


Ejercicios Intro a Express (2)

5. Agrega un endpoint `/api/body` que responda a una petición de tipo PUT con el body que el cliente envíe al hacer la petición. Ejemplo: cliente envía un body desde postman o insomnia que luce como este:

```
{ "nombre": "César", "ocupacion": "Sensei" }
```

Entonces, el servidor debe responder con un objeto idéntico al que envía el cliente, junto con un status de respuesta 200.

6. Vuelve a hacer el ejercicio 2 pero enviando `num1` y `num2` desde el body, a través de una petición POST que responda con el status 200

7. Crea un endpoint para una petición de tipo DELETE donde envíes un ID (un número cualquiera) a través de params.

Si el param contiene el ID 3, entonces responde con un status 200 y el mensaje "se ha eliminado el objeto con ID 3", de lo contrario, si envían cualquier otro número como ID, responde con un status 404 y el mensaje "No se encontró el objeto con el ID especificado".



¡GRACIAS!



César Guerra



www.cesarguerra.mx