UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

Diplomatura en programación web full stack con React JS

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 4: NodeJS Intermedio

Unidad 1: API Rest con JSON

Presentación:

En esta unidad vamos a aprender acerca de cómo se desarrolla un servidor API Rest utilizando para la comunicación el formato JSON.

Objetivos:

Que los participantes:

Aprenda a desarrollar un servidor API Rest con JSON.

Bloques temáticos:

1. Introducción a JSON
2. Métodos HTTP
3. Introducción a API Rest
4. Creación de una API Rest en Express
5. Ejemplo
6. Trabajo Práctico

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

** El MEC es el modelo de E-learning colaborativo de nuestro Centro.*

Tomen nota:

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Introducción a JSON

Es un formato de datos basado en texto que sigue la sintaxis de objeto de JavaScript, popularizado por Douglas Crockford. Aunque es muy parecido a la sintaxis de objeto literal de JavaScript, puede ser utilizado independientemente de JavaScript, y muchos ambientes de programación poseen la capacidad de leer (analizar y procesar) y generar JSON.

Los JSON son cadenas - útiles cuando se quiere transmitir datos a través de una red. Debe ser convertido a un objeto nativo de JavaScript cuando se requiera acceder a sus datos. Ésto no es un problema, dado que JavaScript posee un objeto global JSON que tiene los métodos disponibles para convertir entre ellos.

JSON es una cadena cuyo formato recuerda al de los objetos literales JavaScript. Es posible incluir los mismos tipos de datos básicos dentro de un JSON que en un objeto estándar de JavaScript - cadenas, números, vectores, booleanos, y otros literales de objeto.

Ejemplo de JSON

Analicemos un documento JSON de MercadoLibre

```
{
    "id": "ITEM_CONDITION",
    "name": "Condición del ítem",
    "tags": {
      "hidden": true
    },
    "hierarchy": "ITEM",
    "relevance": 2,
    "value_type": "list",
    "values": [
      {
        "id": "2230284",
        "name": "Nuevo"
      },
      {
        "id": "2230581",
        "name": "Usado"
      },
      {
        "id": "2230582",
        "name": "Reacondicionado"
      }
    ],
    "attribute_group_id": "OTHERS",
    "attribute_group_name": "Otros"
  }
```

En este documento se pueden apreciar un objeto el cual contiene varias propiedades. Los tipos de propiedades incluyen casi todos los tipos de datos disponibles:

- Cadena (String)
- Entero (Integer)

- Booleano
- Objeto
- Vector

Veamos algunos ejemplos de los tipos de datos indicados

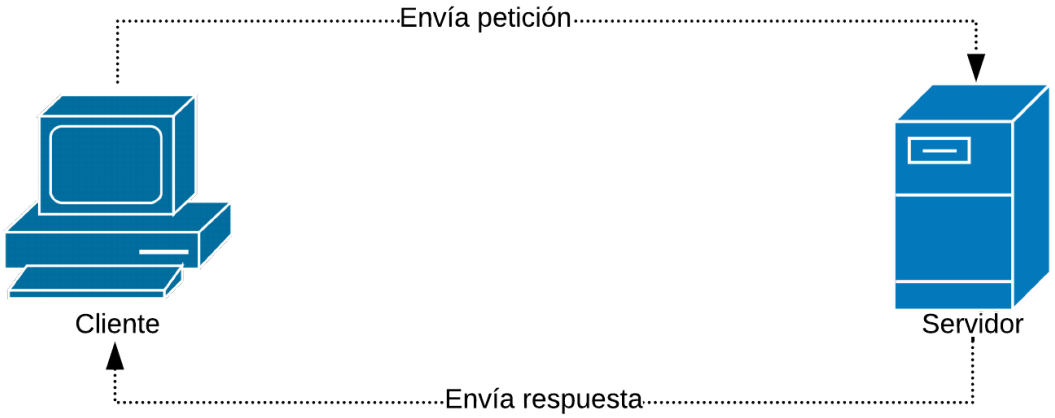
Tipo de dato	Propiedad
String	name, hierarchy, value_type, id, etc
Entero	relevance
Booleano	hidden
Objeto	`{"hidden": true }` `{ "id": "2230284","name": "Nuevo"}`

Vector	`[`
	`{"id": "2230284","name": "Nuevo"},`
	`{"id": "2230581","name": "Usado"},`
	`{"id": "2230582","name": "Reacondicionado"}`
	`]`

2. Métodos HTTP

Peticiones HTTP

La comunicación por medio de HTTP, se centra en el concepto de ciclo de petición-respuesta.



En la cual el cliente envía una petición y el servidor retorna una respuesta.

Para que la petición del cliente sea válida, debe incluir:

- URL (Uniform Resource Locator)
- Método
- Encabezados
- Cuerpo de la petición

URL

Se convirtió en una forma fácil para que el cliente le indique al servidor con que desea interactuar, llamado recursos.

Método

El método de la petición (Ej: GET, POST, PUT, DELETE) le indica al servidor que clase de acción desea realizar el cliente sobre el recurso.

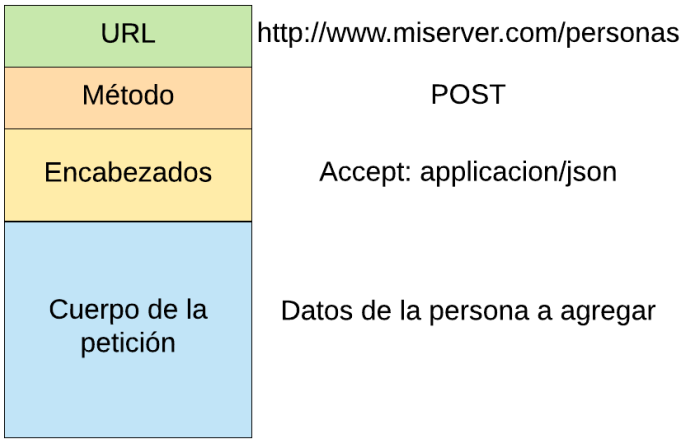
- GET: le solicita al servidor que retorne uno o varios recursos.
- POST: le indica al servidor que debe crear un nuevo recurso.
- PUT: le indica al servidor que debe actualizar un recurso existente.
- DELETE: le indica al servidor que borre un recurso.

Encabezados

Proveen información extra en las peticiones. Generalmente utilizado para indicar el formato de respuesta que se espera del servidor (xml, json, html) y enviar datos de autenticación.

Cuerpo de la petición

Contiene la información que el cliente desea enviar al servidor. Para los métodos POST y PUT, se envía la información con la que el cliente desea crear un recurso (POST) o actualizar un recurso (PUT).



Petición HTTP

Respuestas HTTP

La respuesta del servidor ante la petición del cliente, incluye:

- Código de estado (status code)
- Encabezados
- Cuerpo de la respuesta

Código de estado

Es un código numérico preestablecido que le indica al cliente el resultado de la operación. Entre ellos podemos destacar:

- 200 operación realizada de manera correcta (sin errores)
- 201 recurso creado
- 204 recurso borrado
- 400 petición inválida
- 401 sin autorización (el usuario no se ha autenticado)
- 403 no tiene permisos sobre el recurso (se ha autenticado, pero no tiene suficientes permisos para realizar la operación solicitada en el recurso)

- 404 no encontrado
- 500 error interno del servidor

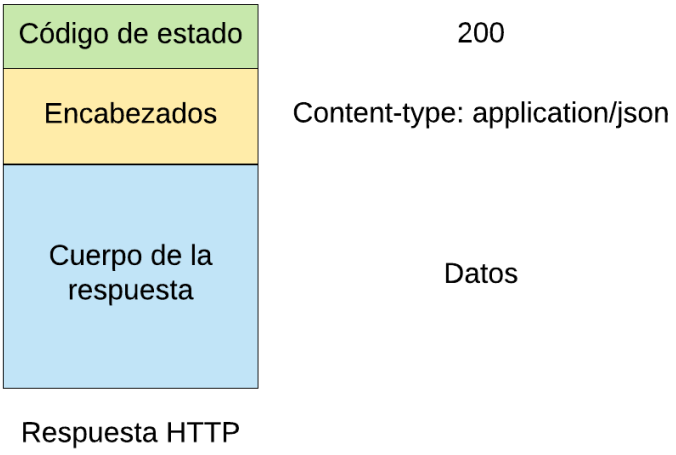
El servidor retorna un único código de estado por respuesta.

Encabezados

Información extra que el servidor le entrega al cliente con respecto a la respuesta.

Cuerpo de la respuesta

Los datos que retorna el servidor al cliente. Por ejemplo, cuando el cliente le solicita un recurso, el servidor retorna en el cuerpo de la petición el recurso solicitado.

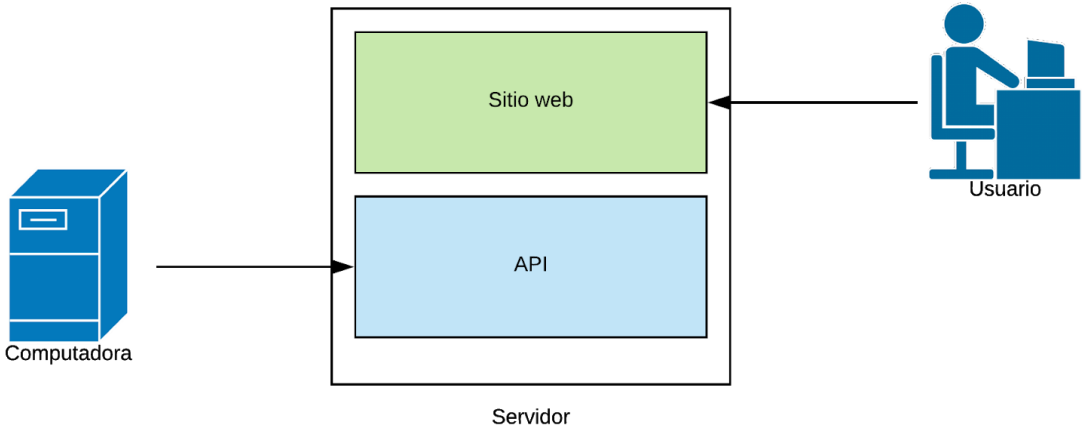


3. Introducción a API Rest

Conceptos de API REST

API significa Interfaz de Programación de Aplicación (Application Programming Interfaces).

Una API es una herramienta que hace que un sitio web sea consumible por una computadora. A través de él, una computadora puede ver y editar datos, tal como lo hace una persona cargando páginas y enviando información por medio de formularios.



Cuando un sistema se enlaza a través de una API, decimos que los mismos se encuentran integrados. Un lado es el servidor que publica la información por medio de la API, y el otro lado es el cliente que consume y manipula la API.

4. Creación de una API Rest en Express

Conceptos previos

Formato de datos

Tanto el cliente como el servidor deben poder interpretar los datos que entre ellos intercambian, es por ello que se utilizan standards de codificación de datos para que ambos se puedan entender. Los dos formatos de datos más utilizados son JSON y XML.

JSON

Es un formato simple que se basa en dividir los datos en clave-valor.

```
{
      "curso": "NodeJS nivel Intermedio"
}
```

XML

Tiene una estructura de árbol compuesta por bloques. El bloque principal es llamado nodo, y a partir de él se crean nodos hijos (como si fueran ramas). El nombre del nodo nos indica el atributo (clave) y la información contenida en el dato el valor.

```
<curso>NodeJS nivel intermedio</curso>
```

Uso del formato de datos

A través de los encabezados, el cliente le puede indicar al servidor que formato formato de dato utiliza para codificar la información, e indicar en qué formato de dato espera la respuesta. Generalmente para los servicios REST, el formato de datos utilizado es JSON.

- Content-type: por medio de este encabezado, el cliente le indica al servidor el formato en el cual está enviando la información.

- Accept: por medio de este encabezado, el cliente le indica al servidor qué tipo de formato es el que espera en la respuesta.

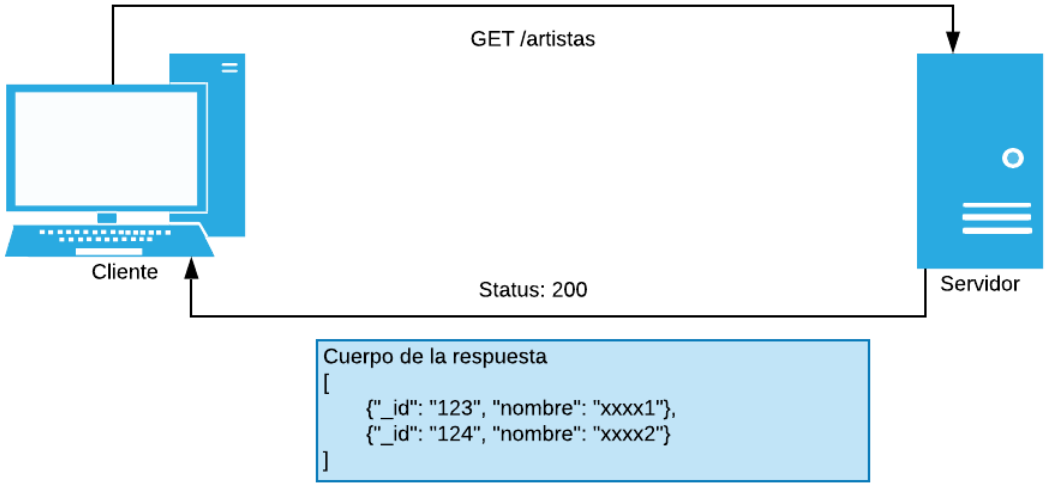
Creación de una API Rest con Express

Ya hemos visto cómo es la comunicación entre el cliente y servidor de una API REST, ahora vamos a ver un ejemplo de cómo se manejan los recursos, para ello vamos a tomar el ejemplo de los recursos para administrar una persona.

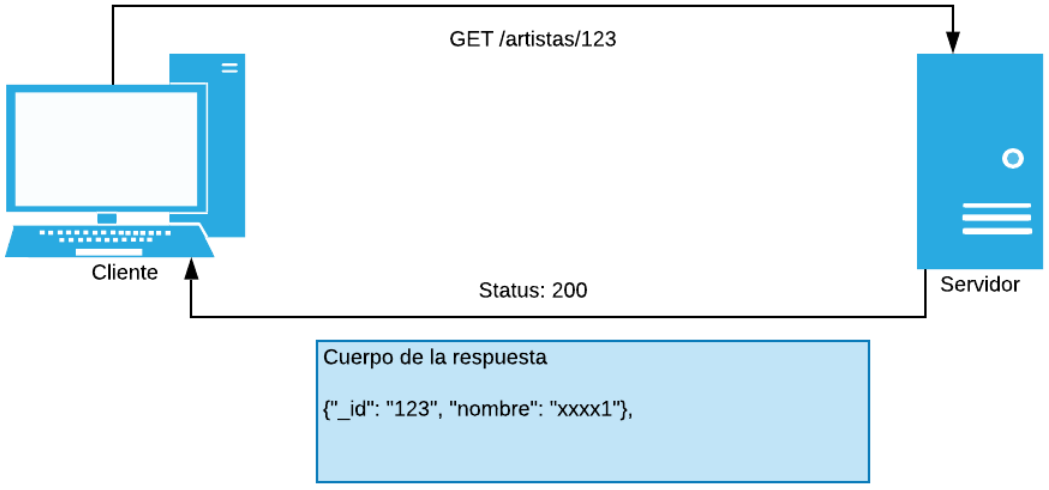
Método	URL	Acción
GET	/artistas	Lista los artistas existentes
POST	/artistas	Agrega un nuevo artistas
GET	/artistas/1	Retorna el artistas con identificador 1
GET	/artistas/20	Retorna el artistas con identificador 20
PUT	/artistas/3	Actualiza el artistas con identificador 3
DELETE	/artistas/8	Borra el artistas con identificador 8

Los métodos POST y PUT son con los cuales el cliente envía contenido en el cuerpo de la petición (ya que debe enviar los datos de la persona, ya sea para crear una nueva POST, o para actualizar una existente PUT).

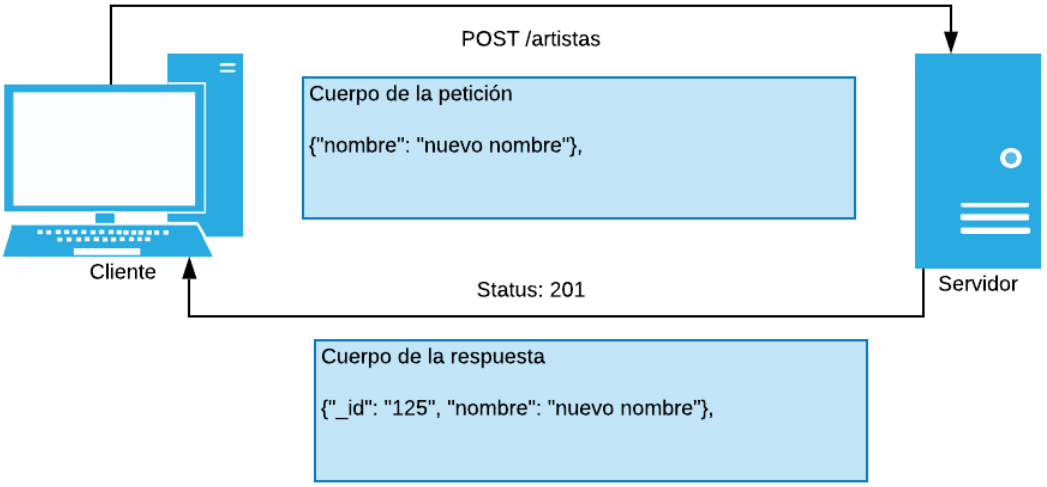
Ejemplo de solicitud de todas los artistas



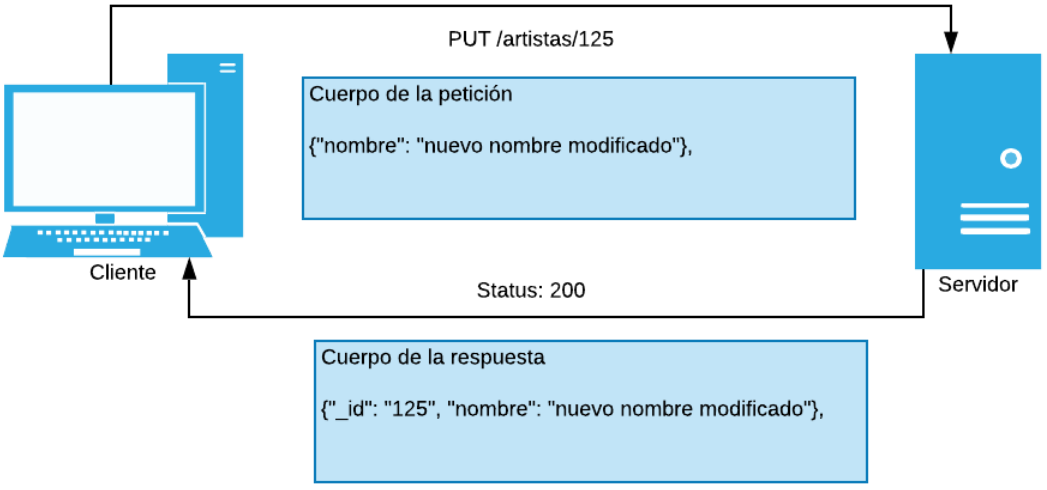
Ejemplo de solicitud del artista con identificador 123



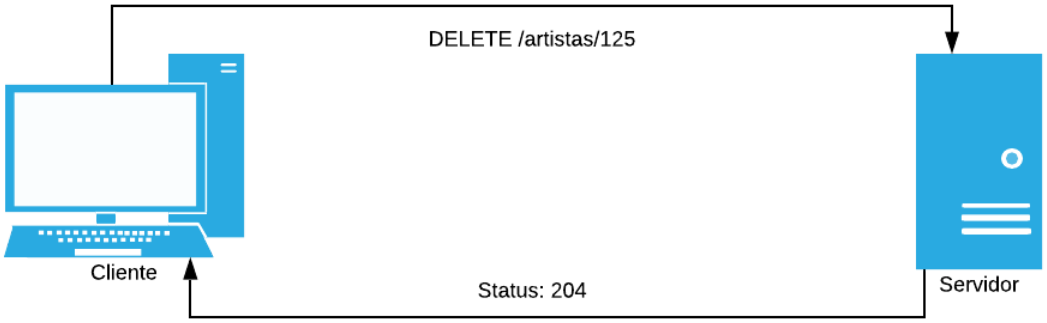
Ejemplo de solicitud de creación de un artista



Ejemplo de solicitud de modificación del artista con identificador 125



Ejemplo de solicitud de eliminación del artista con identificador 125



API REST con recursos enlazados

Si queremos agregar un recurso que depende de otro, por ejemplo, si queremos agregar una canción a un recurso artista, debemos tener una forma de poder manejar dicha asociación de una manera clara y sencilla. Para ello podemos agregar recursos que dependan de otro recurso, esto lo manejamos a través de la URL a la cual se realiza la petición.

Tomemos como ejemplo la administración de las canciones de un artista. Para solicitar todos los recursos canciones asociados al recurso artista con identificador 8, podríamos realizar la siguiente petición:

GET /artistas/8/canciones

Con la cual estamos solicitando todas las canciones, del recurso artista, cuyo identificador (de artista) es el 8.

La administración total de las canciones quedaría:

Método	URL	Acción
GET	/artistas/<id>/canciones	Lista los canciones existentes para el artista especificado

POST	/artistas/<id>/canciones	Agrega una nueva canción al artista especificado
GET	/artistas/<id>/canciones/3	Retorna la canción con identificador 3 para ela artista especificado
PUT	/artistas/<id>/canciones/3	Actualiza la canción con identificador 3 para el artista especificado
DELETE	/artistas/<id>/canciones/4	Elimina la canción con identificador 4 para el artista especificado

5. Ejemplo

Hemos creado una aplicación de ejemplo en Express que implementa la API REST que hemos utilizado en esta unidad (/artistas y /artistas/<id>/canciones) a la cual puedes acceder en:

https://github.com/cursos-utn/nodejs-intermedio/tree/m1u4_cancion_hbs

6. Trabajo Práctico

Utilizar los conceptos aprendidos en esta unidad para crear una pequeña aplicación de temática libre.

Bibliografía utilizada y sugerida

Express (n. d.) Recuperado de: https://expressjs.com/es/

MDN - JavaScript (n.d.) Recuperado de:

https://developer.mozilla.org/es/docs/Web/JavaScript

NodeJs (n. d.) Recuperado de: https://nodejs.org/es/

NodeJS Documentacion (n. d.) Recuperado de: https://nodejs.org/es/docs/

NPM (n. d.) Recuperado de: https://www.npmjs.com/package/page

Wikipedia - NodeJS (n. d.) Recuperado de: https://es.wikipedia.org/wiki/Node.js

Lo que vimos:

API Rest con JSON.

Lo que viene:

Middleware, sesiones y jwt .