

**Back End**

**Carrera  
Programador  
full-stack**

***PRIMERAS RESPUESTAS  
DESDE EL BACKEND***

# Organización de los datos

Una forma de organizar las **variables** y **funciones** es a través de **objetos**.

Encapsulan **datos** y **comportamiento**

Son los objetos nativos en JavaScript

Su “clase” es de tipo “objeto” y son de estructura libre

# Qué es JSON?

(JavaScript Object Notation)

- Organizado
- Fácil de acceder

//objeto 'profesor' con dos atributos

```
{  
  "nombre": "Mauri",  
  "materia": "Back",  
}
```

Se compone de

Registros (objetos): { }

Propiedades: "clave" : "valor"

Arreglos: [ ] (Contienen objetos)



No confundir un Objeto JSON con una función, tiene llaves, pero no tiene parámetros, ni código, ni la palabra **function** antes

# Tipo de datos soportados

Tipo de Dato:

- String
- Número
- Objetos
- Arrays
- Booleano
- Null



# Agregar y acceder a miembros

```
let curso = {  
  "nombre": "full-stack",  
  "alumno": [  
    {  
      "nombre": "juan",  
      "apellido": "perez"  
    },  
    {  
      "nombre": "maria",  
      "apellido": "garcia"  
    }  
  ]  
}
```

# Agregar y acceder a miembros

Podemos leer los miembros:

```
let nombre = curso.nombre;  
let primerApellido = curso.alumno[0].apellido;
```

Les podemos agregar miembros

```
curso.lugar = "Madariaga";
```

# Claves

Las claves pueden o no ir entre comillas

```
{ valor: 4 }
```

O

```
{ "valor": 4 } //Good practice!,
```

evita problemas si tenemos un campo que es una palabra reservada

```
{  
    "error": 4,  
    "if": 6  
}
```

# Valores en JSON

Los valores pueden ser de los siguientes tipos:

```
{  
  "cadena": "texto",  
  "numero": 5,  
  "otroObjeto": {...},  
  "arreglo": [5, "a", 1],  
  "verdadero": true,  
  "nada" : null  
}
```

Notar uso de comillas  
en *value* sólo para  
cadenas de texto



## JSON

```
{  
  "type": "Phone",  
  "price": 120,  
  "inStock": true,  
  "features": [  
    "HD Camera",  
    "Wi-Fi",  
    "LTE"  
  ]  
}
```

JSON.parse()



JSON.stringify()



## JavaScript Object

```
{  
  type: "Phone",  
  price: 120,  
  inStock: true,  
  features: [  
    "HD Camera",  
    "Wi-Fi",  
    "LTE"  
  ]  
}
```

# Ventajas

- Super - Liviano para transferir
- Datos auto-descriptivos
- Legible por el humano
- Fácil de adoptar por los lenguajes orientados a objetos

## { JSON }

```
{
  - Contacts: [
    - {
      FirstName: "Demis",
      LastName: "Bellot",
      Email: "demis.bellot@gmail.com"
    },
    - {
      FirstName: "Steve",
      LastName: "Jobs",
      Email: "steve@apple.com"
    },
    - {
      FirstName: "Steve",
      LastName: "Ballmer",
      Email: "steve@microsoft.com"
    },
    - {
      FirstName: "Eric",
      LastName: "Schmidt",
      Email: "eric@google.com"
    },
    - {
      FirstName: "Larry",
      LastName: "Ellison",
      Email: "larry@oracle.com"
    }
  ]
}
```

## <xml />

```
<ContactsResponse xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <Contacts>
    <Contact>
      <Email>demis.bellot@gmail.com</Email>
      <FirstName>Demis</FirstName>
      <LastName>Bellot</LastName>
    </Contact>
    <Contact>
      <Email>steve@apple.com</Email>
      <FirstName>Steve</FirstName>
      <LastName>Jobs</LastName>
    </Contact>
    <Contact>
      <Email>steve@microsoft.com</Email>
      <FirstName>Steve</FirstName>
      <LastName>Ballmer</LastName>
    </Contact>
    <Contact>
      <Email>eric@google.com</Email>
      <FirstName>Eric</FirstName>
      <LastName>Schmidt</LastName>
    </Contact>
    <Contact>
      <Email>larry@oracle.com</Email>
      <FirstName>Larry</FirstName>
      <LastName>Ellison</LastName>
    </Contact>
  </Contacts>
</ContactsResponse>
```

# Back End

## Carrera Programador full-stack

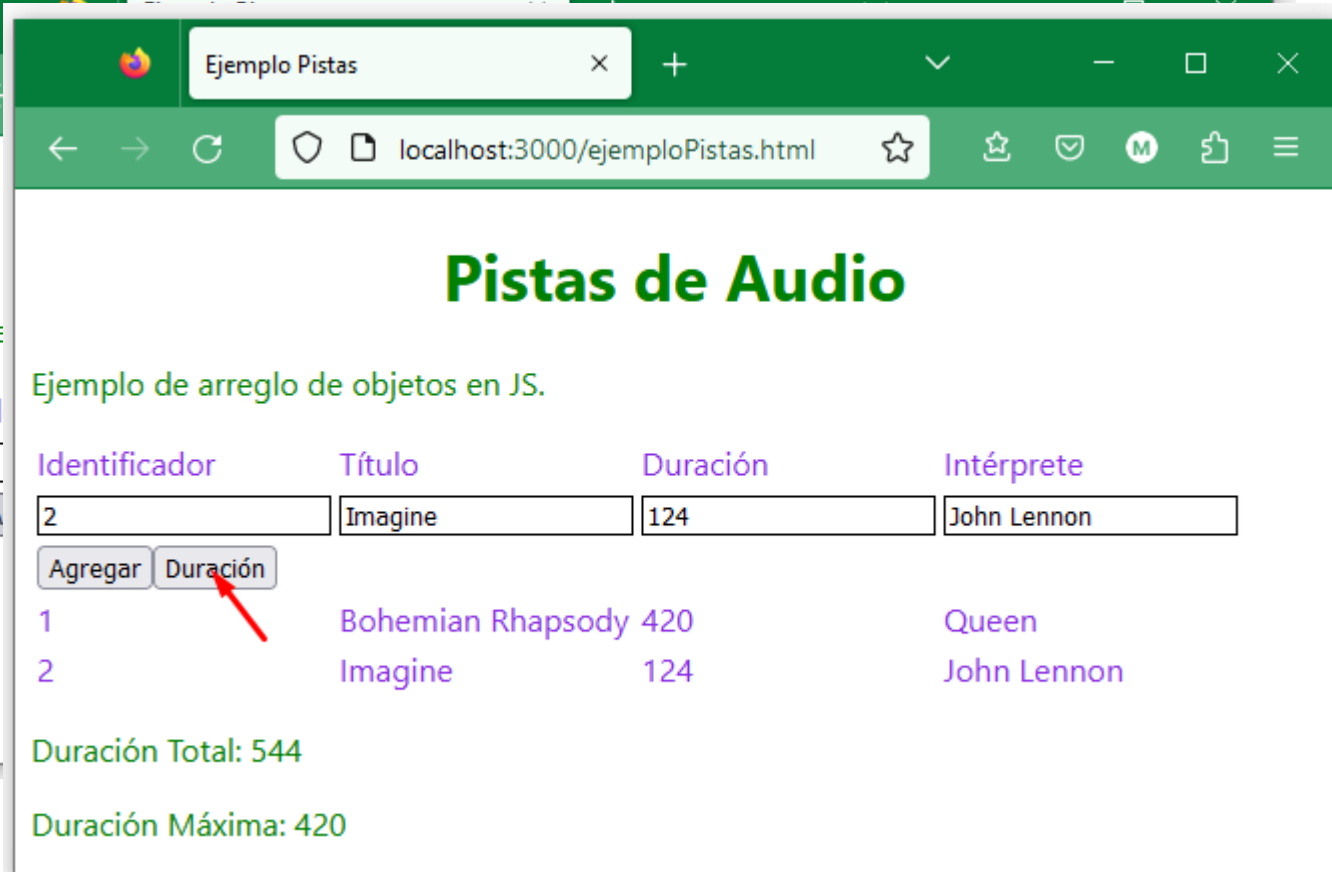
*Ejemplo con JSON*

# Lista de audios

Hagamos una lista de audios (una librería de canciones) que tenga los datos de las pistas como en el ejemplo de POO.

**Manejar un arreglo de objetos en JS**

# Objetivo final



**Pistas de Audio**

Ejemplo de arreglo de objetos en JS.

Identificador	Título	Duración	Intérprete
<input type="text" value="2"/>	<input type="text" value="Imagine"/>	<input type="text" value="124"/>	<input type="text" value="John Lennon"/>
<div><input type="button" value="Agregar"/> <input type="button" value="Duración"/></div>			
1	Bohemian Rhapsody	420	Queen
2	Imagine	124	John Lennon

Duración Total: 544

Duración Máxima: 420

# Modelo de datos

```
{"identificador": 1, "titulo": "titulo 1", "duracion": 120, "interprete": "interprete 1"}
```

```
{"identificador": 2, "titulo": "titulo 2", "duracion": 125, "interprete": "interprete 1"}
```

```
{"identificador": 3, "titulo": "titulo 3", "duracion": 112, "interprete": "interprete 2"}
```

```
{"identificador": 4, "titulo": "titulo 4", "duracion": 210, "interprete": "interprete 1"}
```

```
{"identificador": 5, "titulo": "titulo 5", "duracion": 220, "interprete": "interprete 3"}
```

```
{"identificador": 6, "titulo": "titulo 6", "duracion": 180, "interprete": "interprete 2"}
```

```
{"identificador": 7, "titulo": "titulo 7", "duracion": 130, "interprete": "interprete 3"}
```

```
{"identificador": 8, "titulo": "titulo 8", "duracion": 150, "interprete": "interprete 1"}
```

Vamos a simular una **petición** y su consiguiente **respuesta** del backend. Para una prueba rápida lo haremos de la siguiente manera:

Crearemos en el **servicio** un método **getTracks()** en reemplazo de **getHello()**. El nuevo método retornará una lista de canciones. Por ahora usaremos como base de datos un arreglo en memoria.

A estas alturas ustedes recuerdan que debemos crear una **interfaz** que describa la entidad **track** (cada canción).

Luego, en el controlador, reemplazaremos **getHello()** por el método que acabamos de crear. Examinaremos en el navegador la ruta `/api...` nuestras canciones aparecerán allí.

TS app.service.ts X

src &gt; TS app.service.ts &gt; iTrack &gt; artist

```
1  import { Injectable } from '@nestjs/common';
2  export interface iTrack {
3    id: number;
4    title: string;
5    duration: number;
6    artist: string;
7  }
8
9  > export const tracks: iTrack[] = [ ...
70 ];
71
72 @Injectable()
73 export class AppService {
74   getTracks(): any {
75     return tracks;
76   }
77 }
```

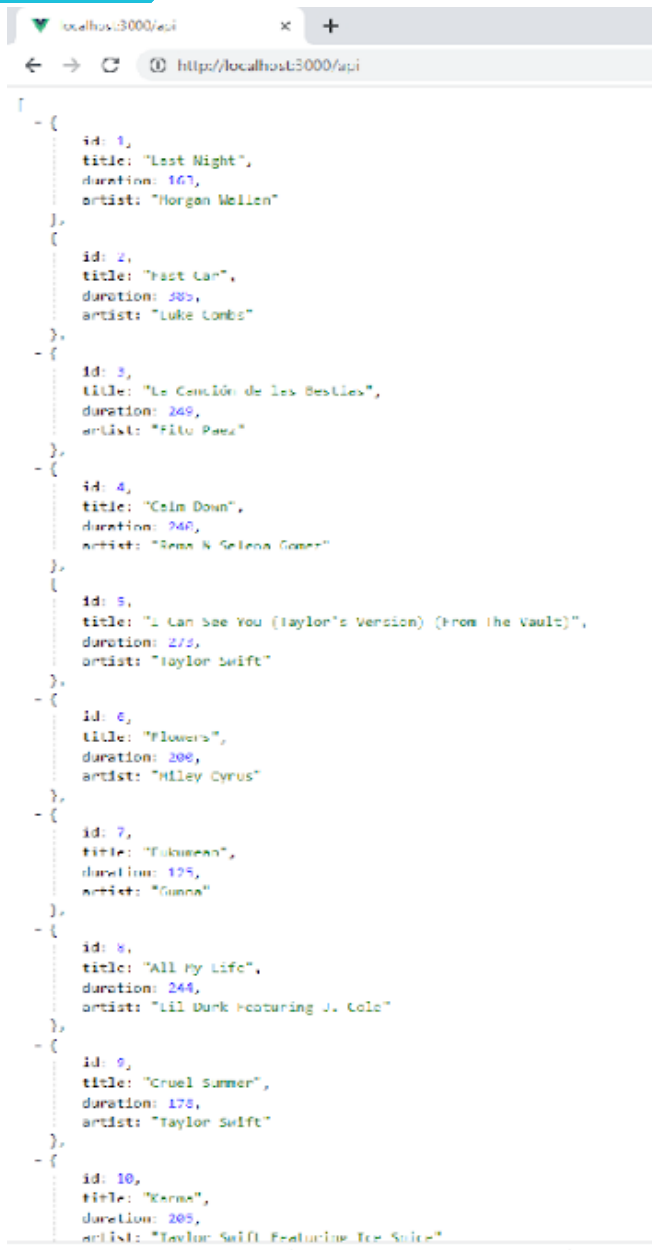


TS app.service.ts

TS app.controller.ts X

src &gt; TS app.controller.ts &gt; ...

```
1  import { Controller, Get } from '@nestjs/common';
2  import { AppService } from '../app.service';
3  import { iTrack } from '../app.service';
4
5  @Controller()
6  export class AppController {
7      constructor(private readonly appService: AppService) {}
8
9      @Get('/tracks')
10     getTracks(): iTrack[] {
11         return this.appService.getTracks();
12     }
13 }
```



```
[
  {
    id: 1,
    title: "Last Night",
    duration: 167,
    artist: "Horgan Wallen"
  },
  {
    id: 2,
    title: "Fast Car",
    duration: 385,
    artist: "Luke Combs"
  },
  {
    id: 3,
    title: "La Cumbia de los Bestias",
    duration: 249,
    artist: "Fito Páez"
  },
  {
    id: 4,
    title: "Calm Down",
    duration: 246,
    artist: "Rema & Selena Gomez"
  },
  {
    id: 5,
    title: "I Can See You (Taylor's Version) (From the Vault)",
    duration: 272,
    artist: "Taylor Swift"
  },
  {
    id: 6,
    title: "Flowers",
    duration: 208,
    artist: "Miley Cyrus"
  },
  {
    id: 7,
    title: "Fukumean",
    duration: 175,
    artist: "Gunna"
  },
  {
    id: 8,
    title: "All My Life",
    duration: 244,
    artist: "Lil Durk Featuring J. Cole"
  },
  {
    id: 9,
    title: "Cruel Summer",
    duration: 178,
    artist: "Taylor Swift"
  },
  {
    id: 10,
    title: "Karma",
    duration: 205,
    artist: "Taylor Swift Featuring Ice Spice"
  }
]
```

Esto debería servirles para entender la mecánica de una **petición** (*request*) desde que entra hasta que se envía la **respuesta** (*response*) desde el backend.

Cabe aclarar que hemos tomado ciertos atajos pedagógicos colocando la interfaz y el objeto de datos dentro del propio servicio.

Vamos a ir corrigiendo estas malas prácticas a medida que avancemos en el aprendizaje de Nest JS y de los conceptos generales de backend.