

Los controladores



Su responsabilidad es la de **atender** los distintos **pedidos** del cliente y generar **comunicación** entre las **vistas** y los **modelos**.

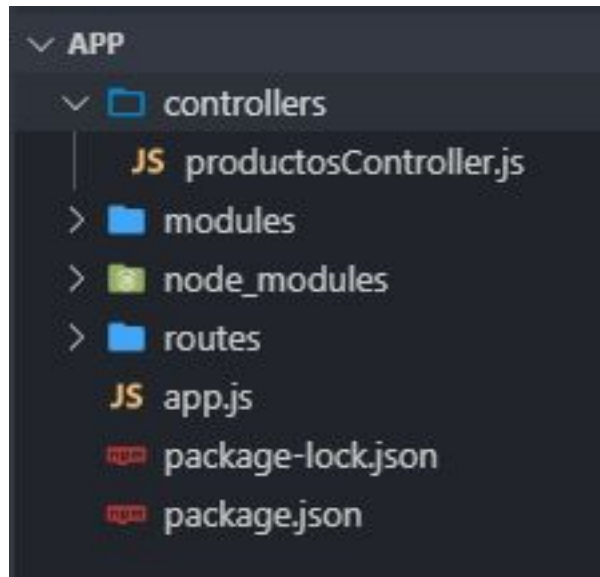


Arquitectura de **archivos**

En la carpeta raíz del proyecto crearemos la carpeta **controllers**. Dentro almacenaremos un controlador por cada recurso o entidad que tenga la aplicación.

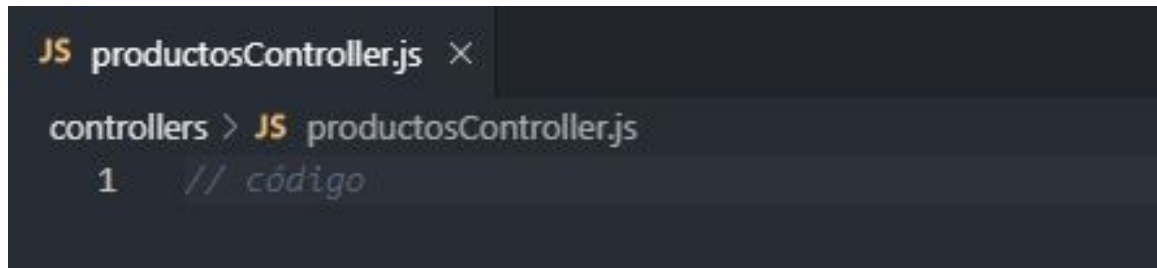
Las entidades pueden ser cosas como usuarios, productos, carritos de compras, etc.

Cada controlador será un **módulo** que exportaremos, para luego requerirlo en donde lo necesitemos.



Creando un controlador

Crearemos un archivo para manejar nuestro recurso. Para nombrar los archivos se estila empezar con el **nombre** del **recurso** seguido de la palabra **Controller** usando el estilo *camelCase*.



```
JS productosController.js X
controllers > JS productosController.js
1 // código
```

Creando un controlador

Lo primero que hay que hacer es definir una variable en el archivo del controlador del recurso y asignarle un objeto literal.

```
{ }  const controlador = {};
```

Dentro del objeto, iremos definiendo los métodos que se encargarán de manejar cada pedido en particular.

```
{ }  const controlador = {  
    index: //mostrar listado de productos,  
    show: //mostrar detalle de un producto,  
    create: //enviar datos para agregar un producto,  
};
```

La vieja forma

Hasta el momento, la **lógica** de cómo manejar cada request que llegaba la veníamos escribiendo en el archivo de rutas de cada recurso, en donde definíamos una **URL** y un **callback** que se encargaba de manejar esa petición.

```
{  
  // código del archivo productos.js en la carpeta routes  
  router.get('/', (req, res) => {  
    res.send('Index de productos');  
  });  
}
```

La nueva forma

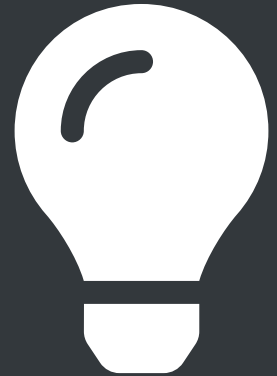
Al trabajar con el patrón **MVC**, podemos apoyarnos en sus reglas y dividir las responsabilidades de nuestros archivos. Siguiendo con esa línea, serán los **métodos** de cada controlador los que **recibirán** esos datos y **enviarán** la información correspondiente.

De modo que quitaremos el callback que habíamos definido en las rutas y lo escribiremos en el método index del controlador de productos.

```
{  
  const controlador = {  
    index: (req, res) => {  
      res.send('Index de productos');  
    },  
  };  
}
```



Como **buena** práctica se suelen usar **nombres descriptivos** y **lógicos** para nombrar a los **métodos** de un **controlador**.



Implementar un controlador

Primero hay que hacer visible todo el código que definimos en el controlador. Para eso, exportaremos la variable en la última línea del archivo.

```
{ } module.exports = controlador;
```

Para empezar a usar los métodos que definimos, debemos requerir el módulo dentro del archivo de ruteo del recurso, en este caso `productos.js`, dentro de la carpeta `routes`.

```
{ } const productosController = require('../controllers/productosController');
```



Para ir una **carpeta** hacia **atrás** cuando indicamos dónde está alojado un **archivo**, debemos escribir dos puntos, seguidos de una barra `../`, al principio de la **ruta**.



Implementar un **controlador**

Con nuestro módulo ya visible en el archivo es momento de terminar de configurar aquella ruta a la que le quitamos el callback.

La misma va a seguir recibiendo ese callback, pero ahora no va a ser su responsabilidad definirlo. Para eso llamaremos al método **index** del controlador de productos y le pasaremos ese método como segundo parámetro. **Al ser un callback no le escribimos los paréntesis.**

```
{ } router.get('/', productosController.index);
```

El código final

En el archivo del controlador, escribimos los métodos de cada pedido y exportamos el objeto literal resultante.

```
{  
  const controlador = {  
    index: (req, res) => {  
      res.send('Index de productos');  
    },  
  };  
  module.exports = controlador;  
}
```

En el archivo de rutas, llamamos a nuestro controlador y en cada ruta ejecutamos el método que corresponda.

```
{  
  const productosController = require('../controllers/productosController');  
  router.get('/', productosController.index);  
}
```

DigitalHouse>
Coding School