

Agencia de
Aprendizaje
a lo largo
de la vida

Desarrollo Fullstack



Les damos la bienvenida

Vamos a comenzar a grabar la clase

A horizontal blue arrow pointing to the right, with a white circle at the tail, a yellow circle in the middle, and another white circle at the head. The yellow circle contains the word 'Extra'.

Extra

Node JS

- Controladores
- ENV



NodeJS

Controllers

Controladores

Son una de las **capas de MVC** (modelo-vista-controlador) y ayuda a separar nuestra aplicación en capas.

Hasta ahora la respuesta a un ENDPOINT se encontraba dentro de la ruta:



```
router.get('/home', (req, res) => res.send("Página de Home"));
```

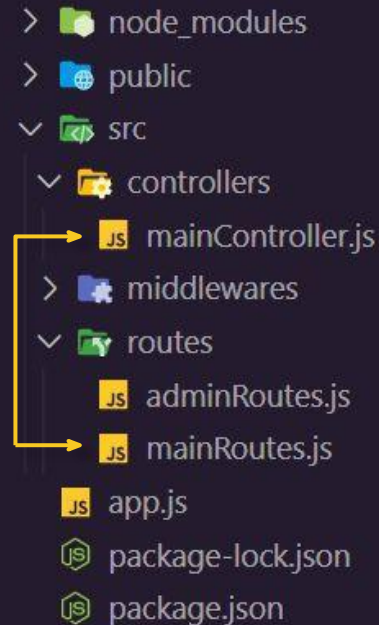
Pero, ¿qué pasa cuando tengo muchas rutas y sus respuestas son más complejas?

Controladores

Lo mejor en estos casos es separar el código para que sea más **escalable** y **legible**.

mainController.js

```
module.exports = {  
  home: (req, res) => res.send("Página de Home"),  
  contact: (req, res) => res.send("Página de Contacto"),  
  about: (req, res) => res.send("Página Sobre Nosotros")  
}
```



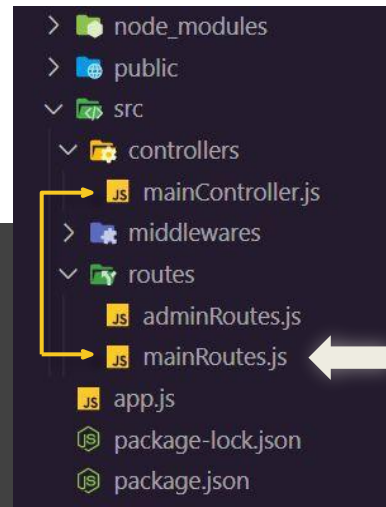
```
> node_modules  
> public  
✓ src  
  ✓ controllers  
    JS mainController.js  
  > middlewares  
  ✓ routes  
    JS adminRoutes.js  
    JS mainRoutes.js  
  JS app.js  
  package-lock.json  
  package.json
```

Controladores

Una vez que **tenemos nuestros controladores** los usamos en el archivo de rutas donde antes colocábamos la **lógica de respuesta**.

mainRoutes.js

```
const mainController = require('../controllers/mainController.js');  
/* MAIN ROUTES */  
router.get('/home', mainController.home);  
router.get('/contact', mainController.contact);  
router.get('/about', mainController.about);  
  
module.exports = router;
```



De esta manera
logramos dividir la
lógica de nuestra
aplicación.



Datos protegidos

En una aplicación a veces necesitamos utilizar **valores** o **constantes** que **NO** se expongan en nuestro código.

Para ello se utiliza algo conocido como **.env**

.env

.env

Comencemos instalando la dependencia **dotenv**:

Consola

```
npm install dotenv
```

Luego **creamos un archivo** en la raíz del proyecto, llamado **.env** y escribimos **un valor que deseamos que sea “secreto” como el puerto** a utilizar en la APP.

.env

```
PORT=3000
```

▼ NODE_SERVER_EXPRESS

> node_modules

> public

> src

≡ .env

JS app.js

JS package-lock.json

JS package.json

.env

Ahora podemos utilizar esa variable en nuestro archivo **app.js**

Es importante tener en cuenta que el `require('dotenv').config()` **debe llamarse antes de cualquier otro código de la aplicación**, que **necesite acceder** a las variables de entorno definidas en el **archivo .env**. Por eso, **a menudo se coloca al principio del archivo principal** de la aplicación, como `app.js` o `index.js`.

De esta manera veremos que seguimos utilizando el mismo **puerto** pero ahora lo **lee desde un archivo oculto**.

```
Servidor corriendo en http://localhost:3000
```

Este enfoque es muy útil por ejemplo para guardar las credenciales de acceso a una base de datos que **no deben ser expuestas**.

app.js

```
/* Requerimos la dependencia*/  
require('dotenv').config();  
  
/* Leemos la constante*/  
const PORT = process.env.PORT;
```

Herramientas que utilizamos en clases



VSCoDe+plugins



No te olvides de dar el presente

Recordá:

- **Revisar la Cartelera de Novedades.**
- **Hacer tus consultas en el Foro.**
- **Realizá los ejercicios obligatorios.**

Todo en el Aula Virtual.