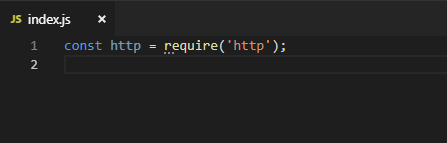
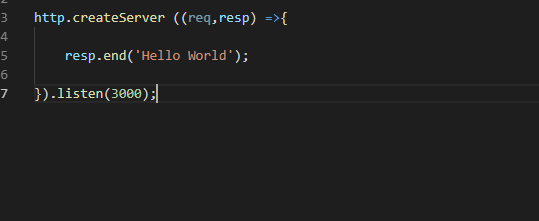
Crear un servidor Node.js

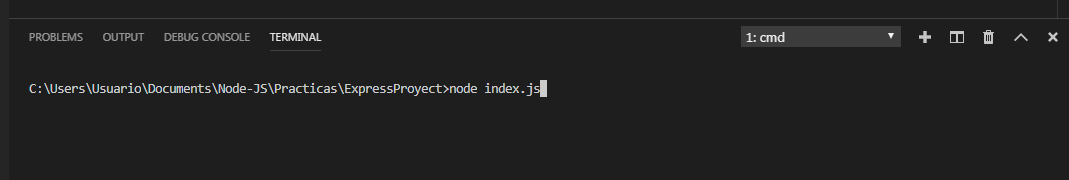
Importamos el modulo http al proyecto



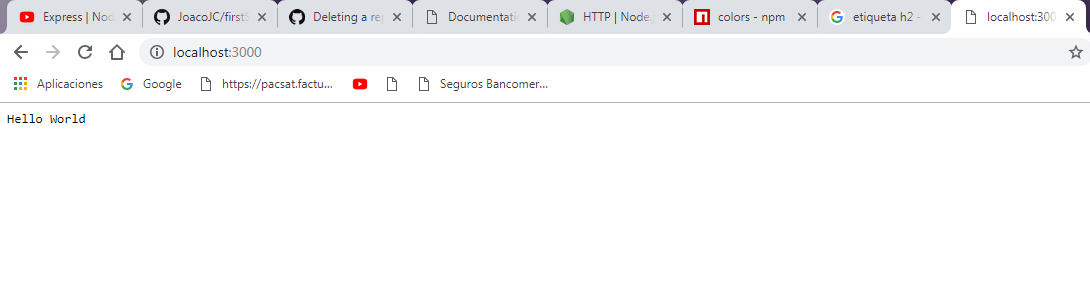
Creamos un servidor escuchando por el puerto 3000



Ejecutar el servidor con el comando **node index.js**

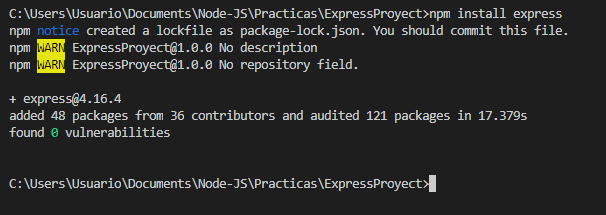


El servidor fue levantado correctamente

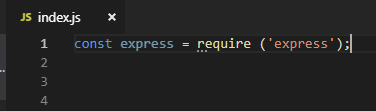


Crear un servidor Express

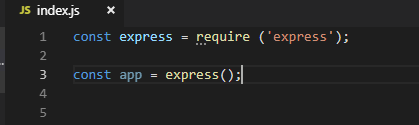
Instalar express con el comando **npm install express**



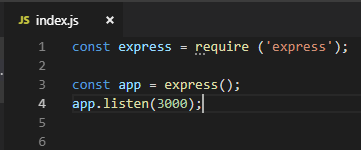
Importar el modulo express



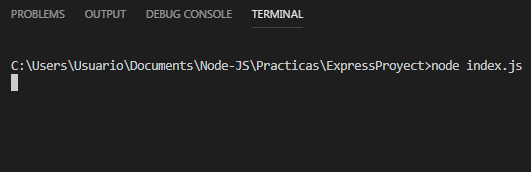
Inicializar express

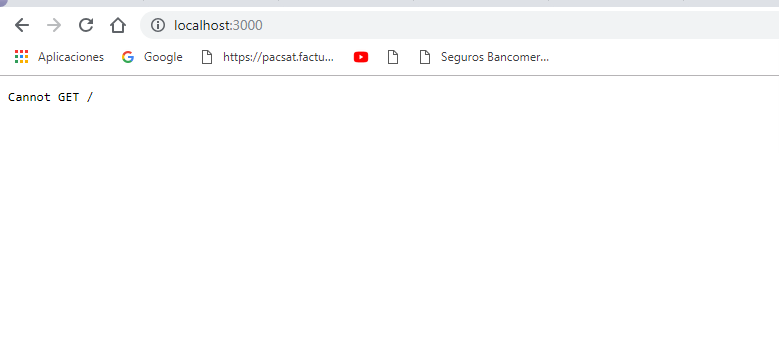


Crear el servidor escuchando por el puerto 3000

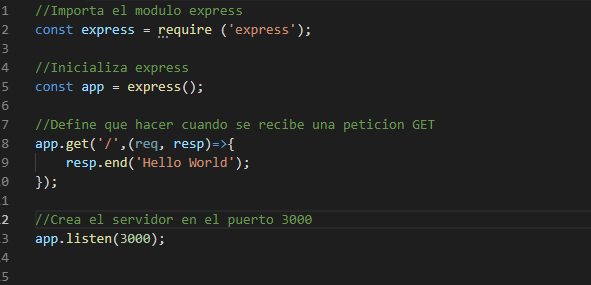


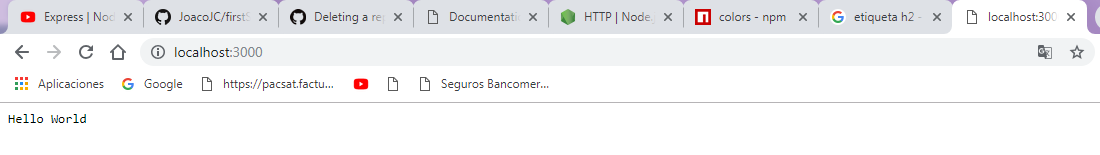
Levantamos el servidor con el comando node index.js



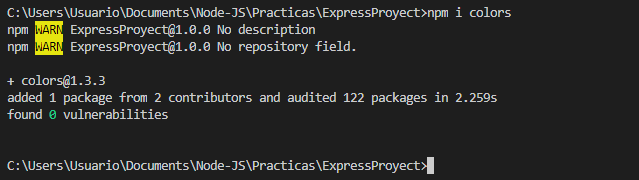


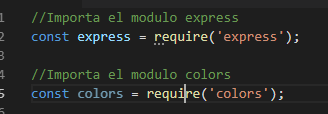
Definimos que hacer cuando se recibe un GET



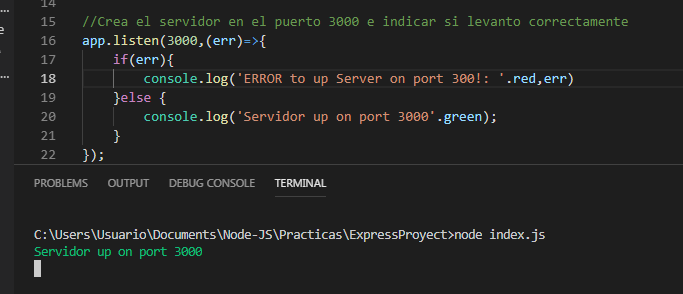


Importamos el modulo colores para poner colores a los mensajes de la consola

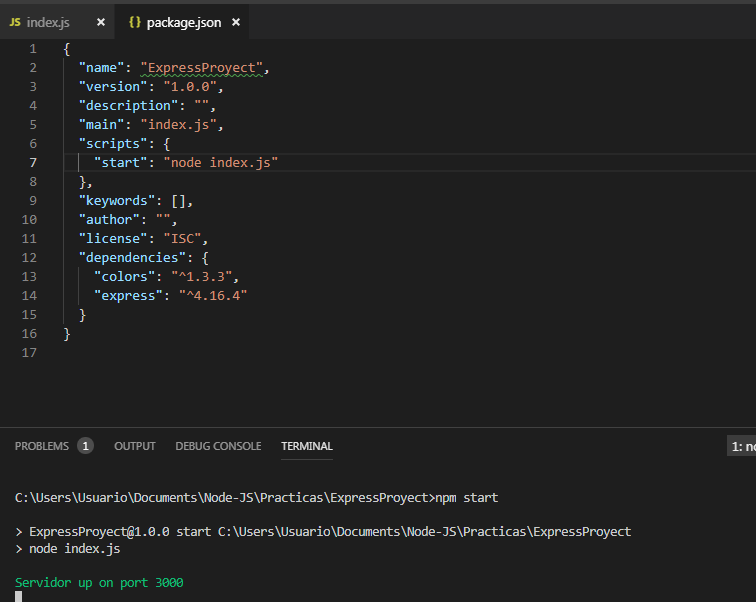




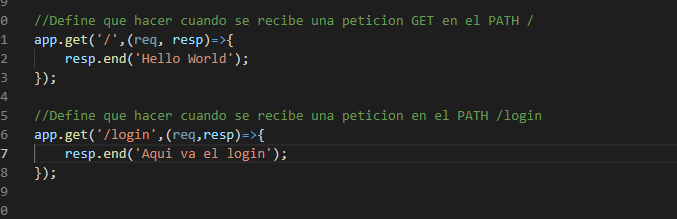
Agregamos una función asíncrona para verificar si el servidor levantó correctamente



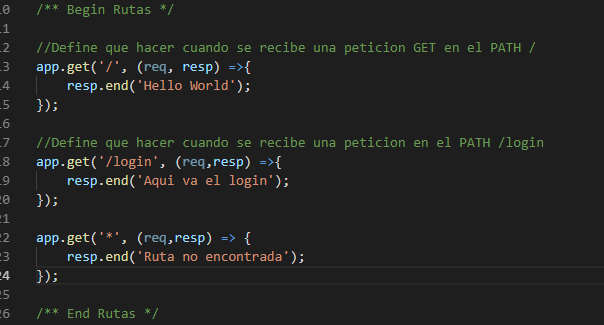
Agregar una forma abreviada de iniciar el servidor



Agregamos una ruta /login



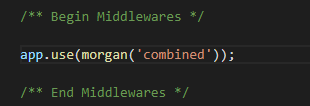
Agregamos una ruta que es cualquiera y regresa un error si no existe



Agregamos Midlewares encargados de realizar acciones intermedias antes de responder la petición



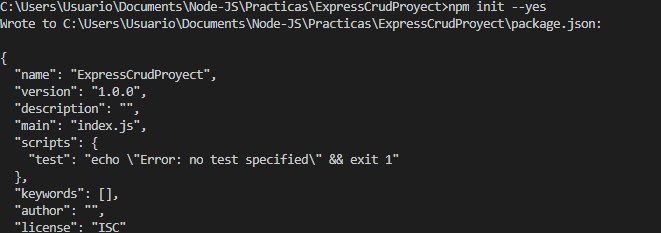
Agregamos Midlwares utilizando MORGAN

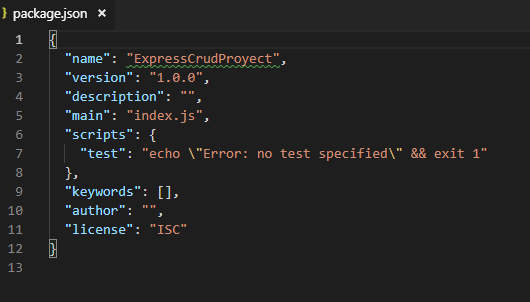


Agregar configuraciones a la aplicación

**Aplicación CRUD** **con NodeJS, Express y MySQL**

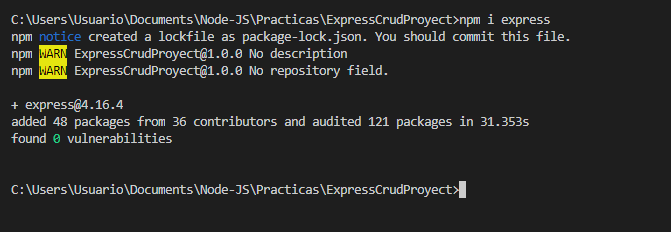
1.- Inicializamos el proyecto con el comando **npm init –yes** para crear el archivo de configuración **package.json** el cual es un archivo que contiene información acerca del proyecto así como las dependencias que se vallan agregando



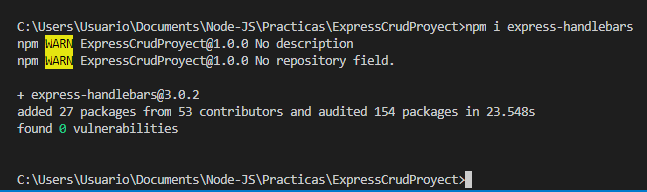


2.- Instalar los módulos necesarios para el proyecto

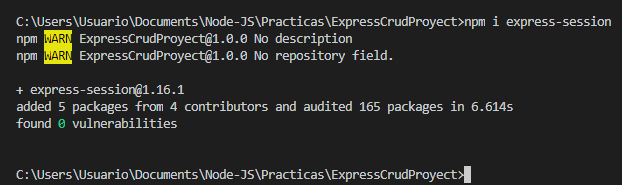
* Instalamos un framework para el back-end en este caso será **Express** lo instalamos con el comando **npm i express**



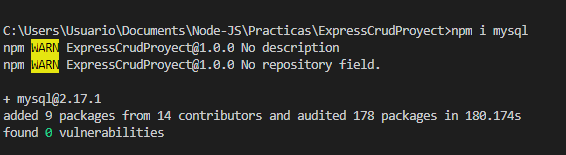
* Instalamos un complemento de express llamado **express-handlebars** es la integración del motor de plantillas en express. Se pueden utilizar también otras opciones como **jade, pug, ejs, jinja, etc.** Pero en este caso instalamos express-handle bars con el comando **npm i express-handlebars**



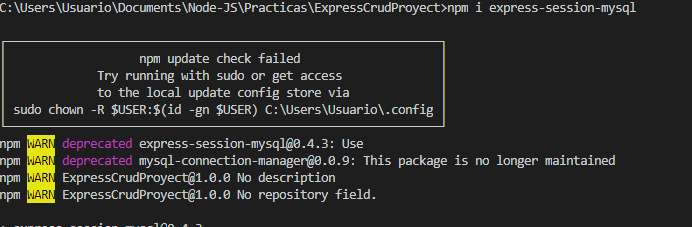
* El siguiente modulo a instalar es un módulo que nos ayudará a controlar las sesiones en express denominado **express-session** administra las sesiones de la app nos será útil para autenticar a un usuario, lo instalamos con el comando **npm i express-session**



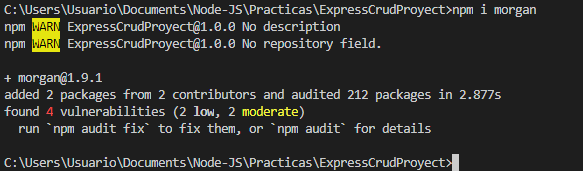
* Otro modulo que necesitaremos será el modulo para conectarnos a la base de datos Mysql para instalarlo escribimos el comando **npm i mysql**



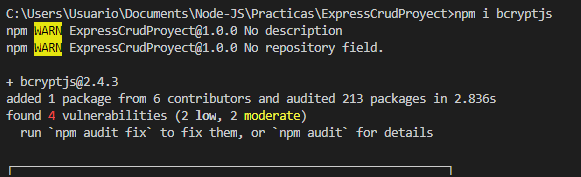
* El siguiente modulo a instalar nos ayudará a controlar las sesiones dentro de la Base de Datos en lugar del servidor ideal para producción para instalarlo escribimos el comando **npm i express-mysql-session**



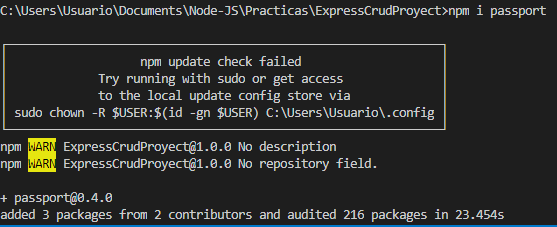
* Instalaremos un módulo que nos ayudará a crear logs o mensajes de que es lo que las aplicaciones cliente le están solicitando al servidor para esto instalaremos Morgan con el siguiente comando **npm i Morgan**



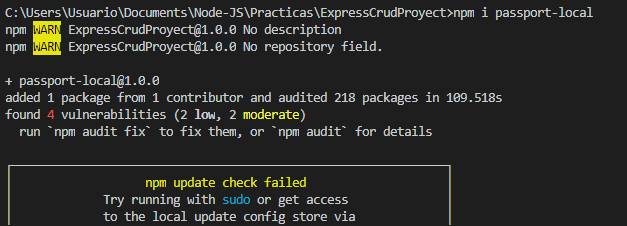
* El siguiente modulo será para cifrar las contraseñas de los usuarios antes de guardarlas en la base de datos el módulo será bcryptjs lo instalamos con el comando **npm i bcryptjs**



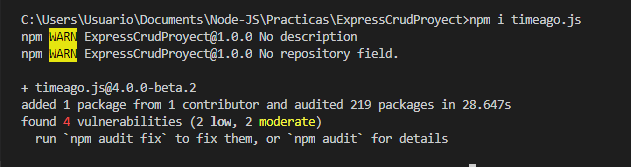
* Otro módulo que instalaremos será utilizado para autenticar y manejar el proceso de login de un usuario en nuestra aplicación el cual será Passport lo instalamos con el comando **npm i Passport**



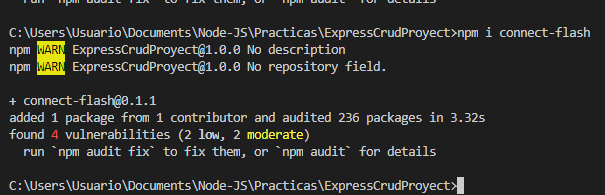
* El siguiente modulo complementa al anterior el cual será utilizado para realizar la autenticación de manera local es decir desde nuestra propia base de datos el módulo a instalar es Passport-local lo instalamos escribiendo el comando  **npm i Passport-local**



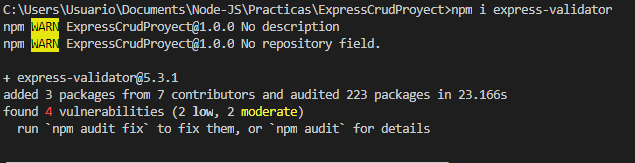
* Instalaremos un módulo que nos convertirá las fechas en un formato de: 2minutes ago, 2 hours ago, etc. El módulo será timeago.js lo instalamos con el comando **npm i timeago.js**



* El siguiente módulo nos permitirá intercambiar datos entre las vistas, denominado connect-flash lo usaremos para mostrar mensajes de error y éxito cuando el usuario realice una operación lo instalamos con el comando **npm i connect-flash**



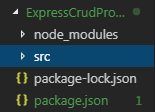
* Instalaremos un módulo que nos ayudará a validar los datos que el usuario nos envía desde la aplicación cliente, el cual será express-validator lo instalamos con el comando **npm i express-validator**



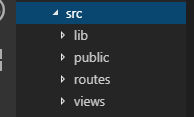
* Instalamos un módulo que nos ayudará con el desarrollo llamado nodemon lo instalamos con el comando **npm i nodemon –D**

3.- Empezamos a crear la estructura del proyecto

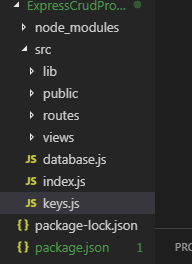
* Generamos una carpeta llamada src donde almacenaremos todo el código



* Generamos las siguientes carpetas dentro de la carpte src:
  + Lib
  + Public
  + Routes
  + Views

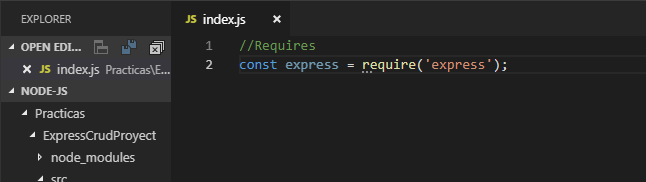


* Iniciamos con la generación de los archivos:
  + Index.js en src archivo encargado de arrancar la aplicación
  + Database.js en src archivo encargado de crear la conexión a mysql
  + Keys.js en src archivo donde almacenaremos palabras clave para la aplicación como conexión a la bd, contraseña, dirección de donde está la bd, etc.

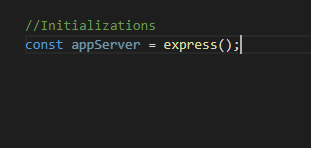


4.- Arrancamos editando el archivo index.js

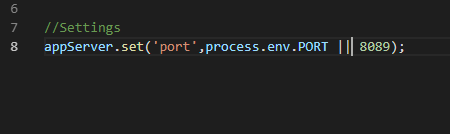
* Empezamos requiriendo el módulo de express



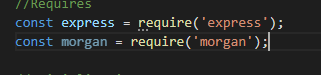
* Inicializamos express



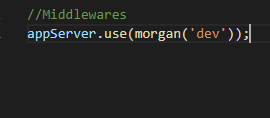
* Definimos el puerto que tomará el servidor, si se encuentra un puerto disponible lo toma y si no toma el 8089



* Hacemos uso de los middlewares como el modulo Morgan para generar logs de las peticiones
  + Requerimos Morgan



* + Hacemos uso de Morgan pasándole como parámetro **dev** que nos ayudará a imprimir ciertos datos de las peticiones en consola



* Inicializamos el server en el puerto determinado

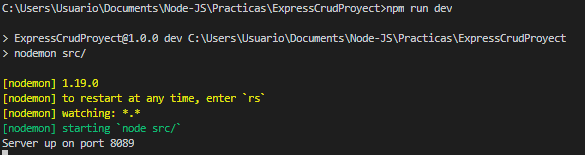


5.- Editamos el archivo package.json para configurar unos comandos

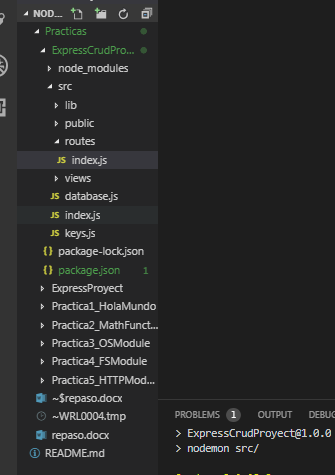
* Generamos el comando dev para ejecutar el comando **nodemon src/** para que busque el archivo **index.js** y lo ejecute



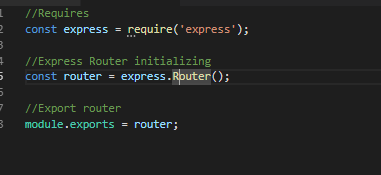
* Ejecutamos el comando con **npm run dev**



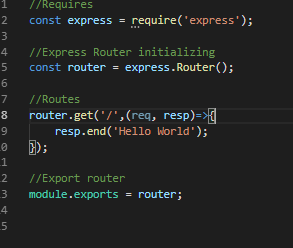
6.- Creamos un archivo index.js en la carpeta routes utilizado para almacenar todas las rutas principales de la aplicación



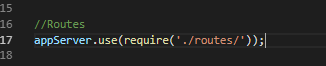
* Requerimos express, lo inicializamos llamando su método Router y lo exportamos el modulo



* Creamos una ruta de ejemplo ‘Hello World’

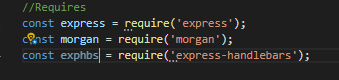


* Hacemos uso de la ruta definida en el archivo index.js de la ruta principal con el método use

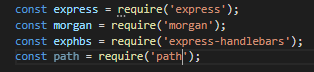


7.- Iniciamos a hacer uso del motor de plantillas **express-handlebars**

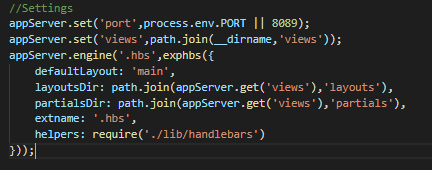
* Requerimos el modulo express-handlebars en el archivo index.js principal



* Configurar el renderizador de vistas en la aplicación
  + Requerimos el módulo PATH



* Le mandamos un objeto con todas las definiciones que utilizaremos
  + Defaultlayout: Es la plantilla principal de handlebars
  + layoutsDir: Ruta de donde tomará los layouts vistas
  + partialsDir: Ruta de donde tomara los partials, es la parte que debe estar en todas las vistas, navbar, footer, header, etc.
  + Extname: extensión que tendrán nuestros archivos handlebars
  + Helpers: ruta de donde tomará los helpers para mostrar las vistas

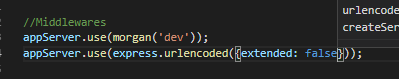


* Le indicamos al appServer que renderizador va a utilizar

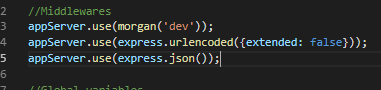


8.- Agregamos un midleware más **express.urlencoded** para poder aceptar desde los formularios los datos que envían los usuarios desde el app cliente, también añadimos para aceptar JSON

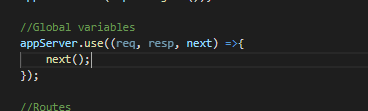
* Añadimos el midleware **express.urlencoded**



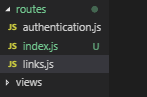
* Añadimos el midleware **json**



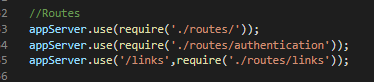
9.- Añadimos a las variables globales una función que tomará los datos enviados por el usuario y una función para continuar con el flujo del código



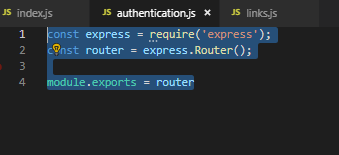
10.- Agregamos dos archivos al directorio routes **authentication.js** y **links.js**

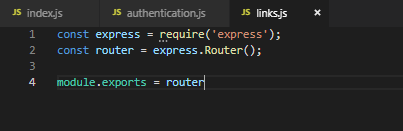


11.- Importamos los archivos creados en el archivo principal **index.js**



12.- Requerimos Express en cada uno de los archivos y los exportamos





13.- Le indicamos al server la carpeta public

