CLI Common line interface

NPM Node Package Manager

1 crear las carpetas

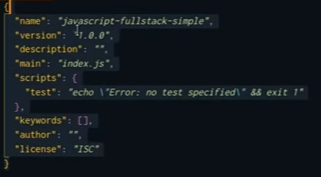


2 Crear un archivo package.json con la descripción del proyecto

npm init -y







3 instalar



npm i express mongoose morgan multer dotenv cross-env cors fs-extra

Express: framework nodejs servidor

Mongoose: modulo de conexión bdd

Morgan: modulo ir viendo por consola en el servidor

Multer: subir imágenes al servidor

Dotenv: variables de entorno, api

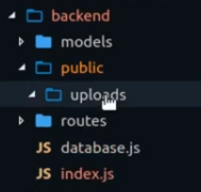
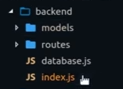
Cross-env: definir en que entorno estamos

Cors: permite enviar y recibir daros de los dos servidores

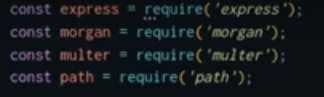


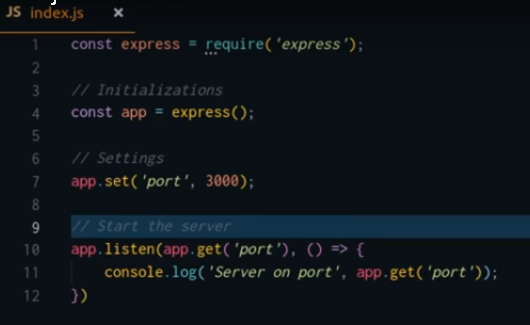
Extra: copiar archivos

4 crear la estructura del proyecto



configurar el server con express





Probar en consola (el servidor)



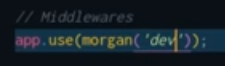
Morgan (muestra por consola detalles de ejecución del servidor)



Ejecutar Morgan



Esto es el valido para corregir error



***instalar la dependencia para desarrollo***

**- nodemon** (permite ejecutar código si se realiza un cambio y reinica el servidor automaticamente)

npm i nodemon -D



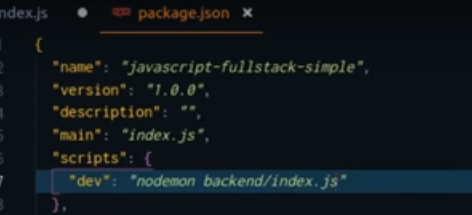
Ejecutar nodemon (automatiza el servidor, ejm reinicio y otros)



Si presenta error



Mejor opción para ejecutar nodemon



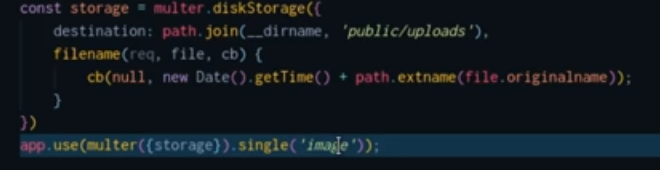
"dev":"nodemon backend/index.js"

Ejecutar la configuración anterior



npm run dev

* **Configurar multer (para subir archivos)**



//multer(subir imagenes)

const storage=multer.diskStorage({

    destination: path.join(\_\_dirname,'public/upload'),

    filename(req,file,cb){

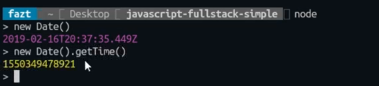
        cb(null,new Date().getTime()+path.extname(file.originalname))

    }

})

app.use(multer({storage}).single('image'));

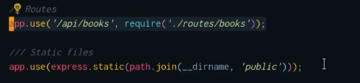
Ejecutar comandos de node (funciones para capturar la fecha y su código para dar como nombre del archivo)



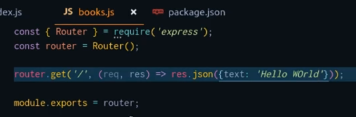
* **Para entender los datos que se envían del frontendde los formularios y json que recibe el backend**

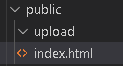


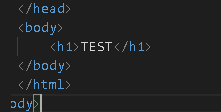
**--routes**











Conectar Mongo db

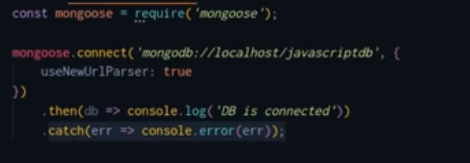
Iniciar el servidor de bdd



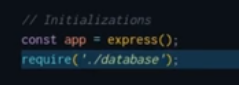
Conectar

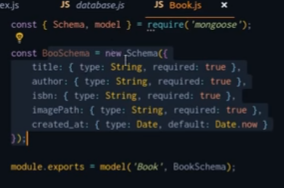


En database.js



Requerir el modulo en index.js





**Variables de entorno**

*Crear el archivo .env para guardar los parámetros, y agregar las variables*

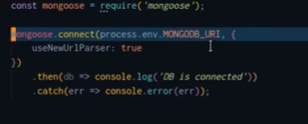




Requerir dotenv al incio de index.js para manejar las variables de entorno



Utilizar las variables





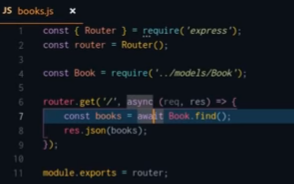
*Modificar el script de inicio para validar se inicia en desarrollo o producción*

**

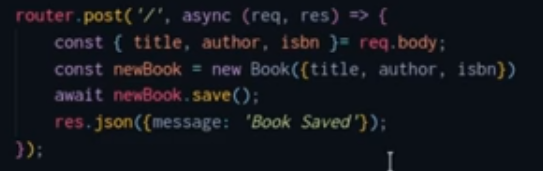
Validad si estas en desarrollo



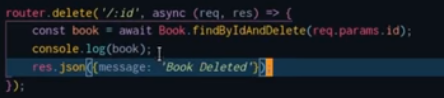
Trabajar con book



Crear controlador ruta para agregar nuevo libro



Controlador para eliminar



FRONTEND

Instalar



**Configurar webpack**

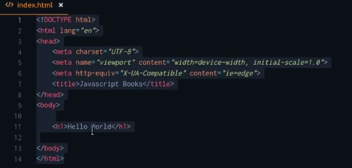
-crear el archivo en frontend



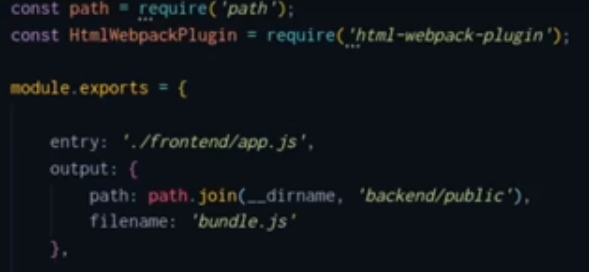
-ejecutar webpack

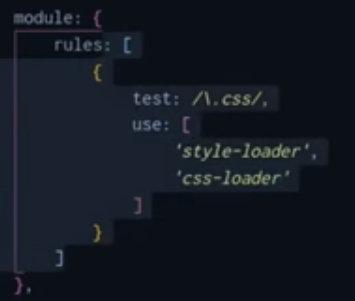


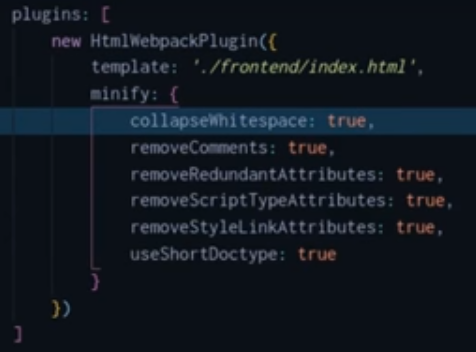
-crear la interfaz en el index del frontend



* Webpack.config







* Crear un archivo para estilos

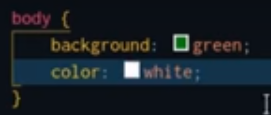


* Código de app.js

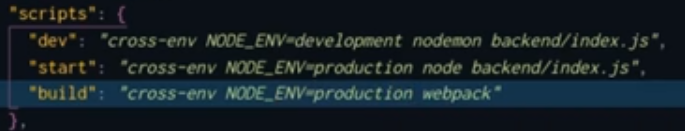
Requerir el css desde el app.js pq webpack pone de manera automática al html



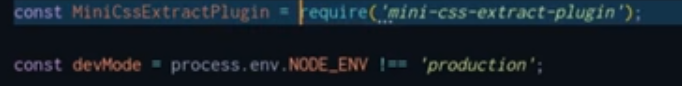
* Código de style.css



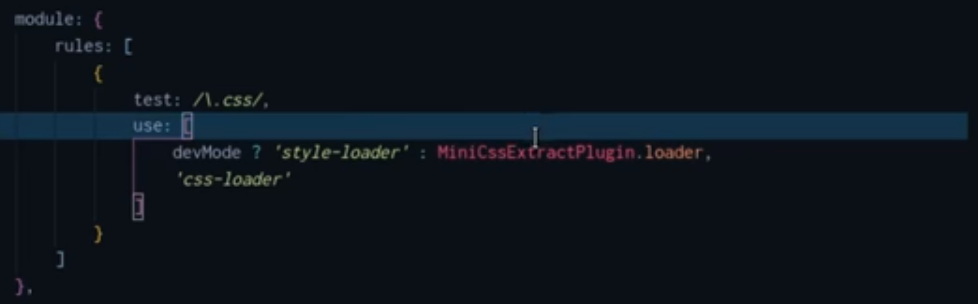
* Agregar el package json para ejecutar en producción



* Agregar estas constantes para evaluar si se está en producción y para extraer el css



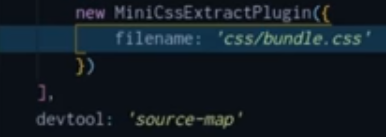
* Modificar module para especificar si extrae el css decuerdo a si producción o desarrollo



* Agregar en plugins el archivo que generara el plugin que extrae el css



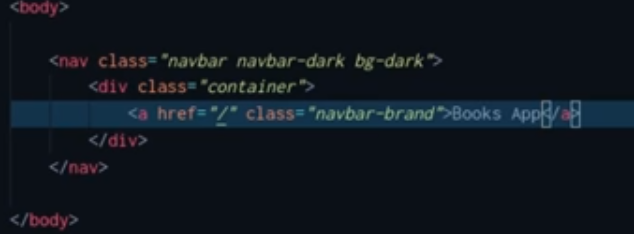
* Agregar un plugin para q muestre en q línea ocurre un error



* Para ignorar los cambios realizados en el frontend



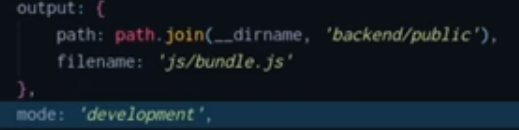
Creacion de la interfaz



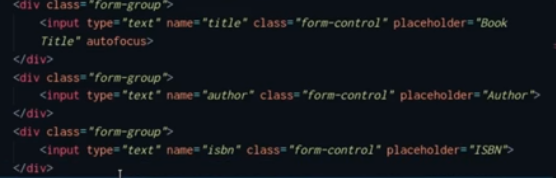
Generar servidor desarrollo front

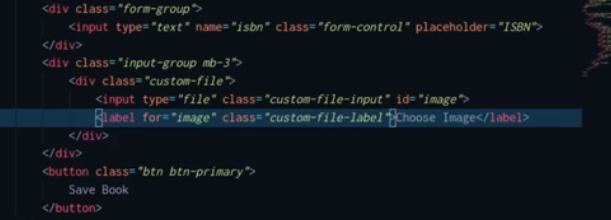


Asiganr el modo de ejecución config-webpack

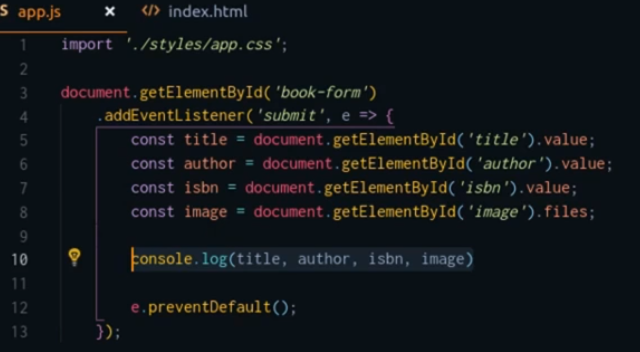






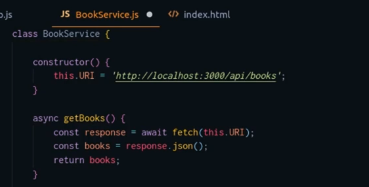


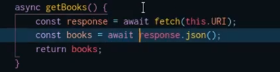
Capturar lo datos de la interfaz

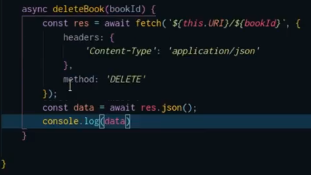


Crear el archivo bookservice

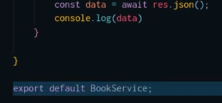








Exportar el servicio



Importar el servicio el app.js

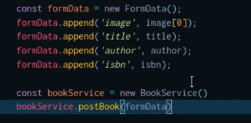


Modificar el index.js del backend para agregar coords q permite que se comuniquen los dos servidores





Llamar al servicio de app.js y probar el evento post



Cambiar el routes/books.js para guardar la ruta de la imagen

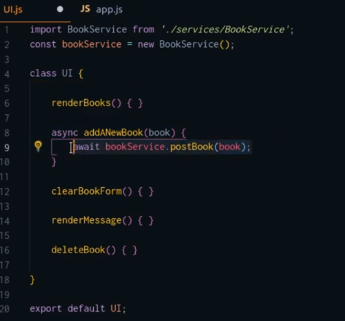


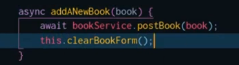
Crear la clase UI.JS

Quitar el servicio de app.js y pasarlo a UI



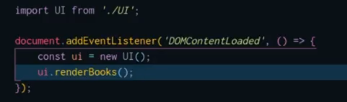
En Ui importar el servicio y generar los métodos





  
En app.js importar UI y llamar al método Add

Cargar los datos al cargar la pagina app.js

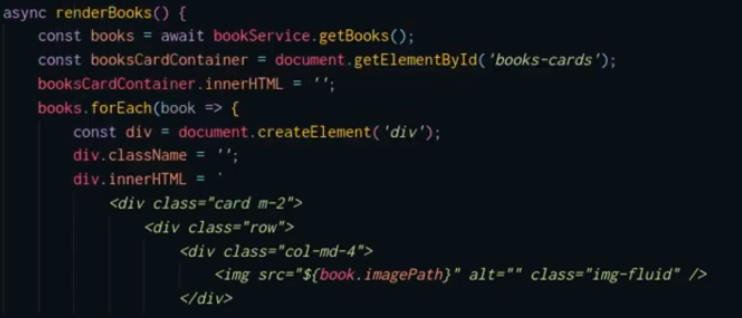




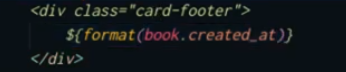
Importar format en UI

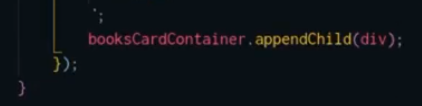


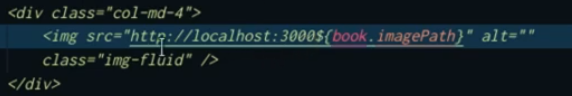
Método render

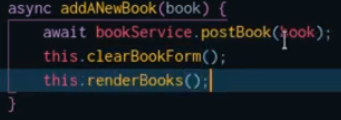






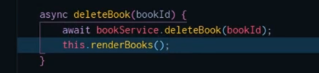




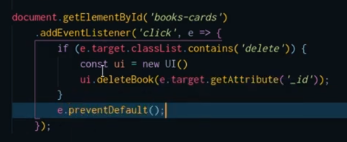


Eliminar registro

Actulizar el método delete UI



Captutar el evento delete en app.js y llamar a UI para eleminiar

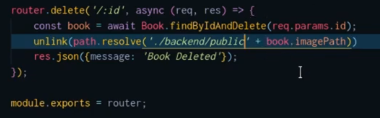


Eliminisr la imagen del servidor en backend/book.js

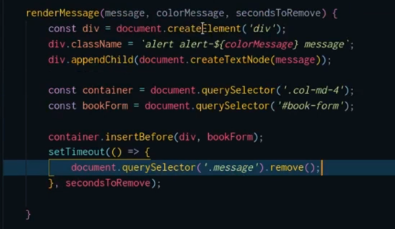
-agregar los



- eliminar la imagen



Crear div para mensajes en UI



Llamar a los mensajes app.js

Luego de agregar



Luego de eliminar



Configuración Proxy GIT

git config --global http.proxy http://vsotog:654123@172.33.0.10:8080

git config --global --get http.proxy

git config --global --unset http.proxy