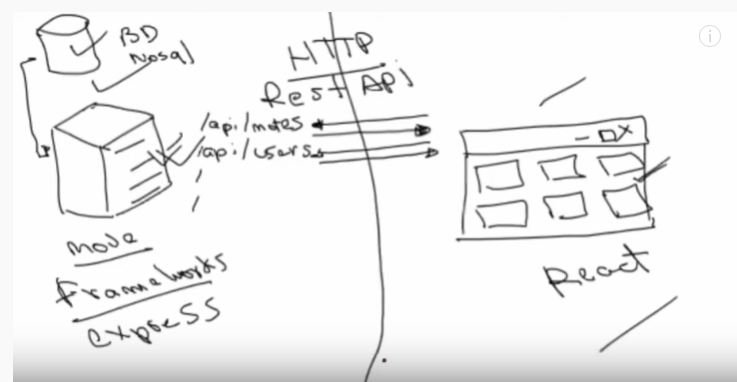
**DOCUMENTACIÓN DE CURSO FULL-STACK CON ARQUITECTURA MERN**

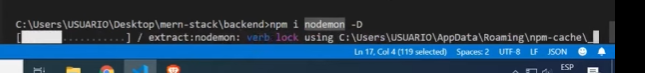
* **PRIMERA CLASE**



* Modelo de como estaría toda la pagina web completa tanto con su backend conectado ala base de datos y los dos conectados con el fronted.
* **SEGUNDA CLASE**

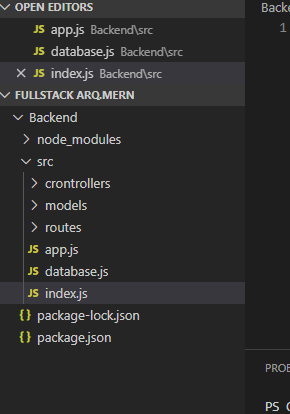


* Dependencias principales
  + **Express:** es para poder interactuar con el servidor y también se maneja mucho en node ya que es un framework
  + **Mongose:** es para conectar el backend con la base de datos y también para poder tener un poco la base de datos relacionales ya que en mongo es no-relacianol
  + **Cors:** es para se puedan comunicar tanto el backend con el fronted
  + **Dotenv:** es para las variables de entorno

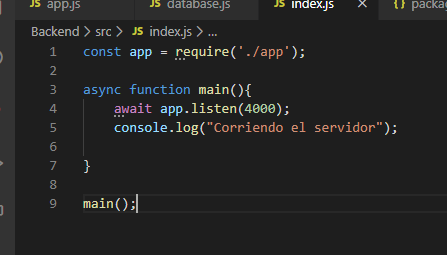
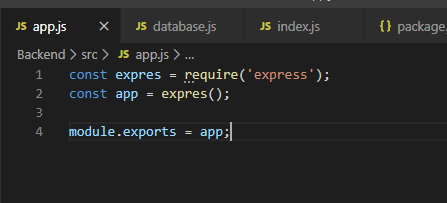


* Dependencia secundaria para poder trabajar más cómodo
  + **Nodemon:** es para que el servidor se recargue solo sin necesidad de hacerlo nosotros mimos

ESTRUCTURA DEL BACKEND

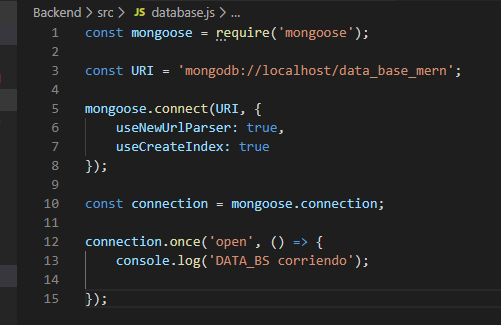


INCIALISAMOS EL SERVIDOR



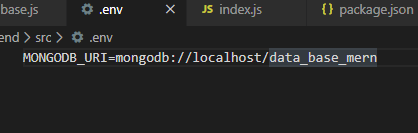
* **TERCERA CLASE**

CONECTAMOS EL SERVIOR CON LA BASE DE DATOS QUE ES MONGO\_DB

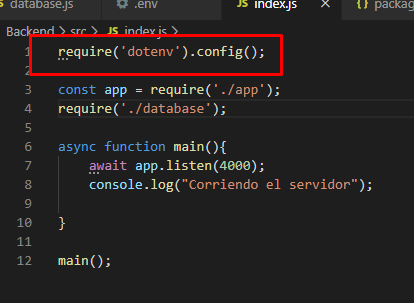


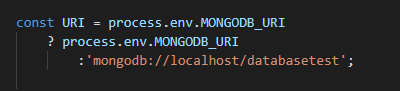
* En la línea **5** pasamos la base de datos y la configuración por defecto para no tener problemas con la base datos al tener su conexión
* Línea **10** y **12** en la **10** se hace la conexión y en la **12** validamos si conecto correctamente con **once** y verificamos con un console.log

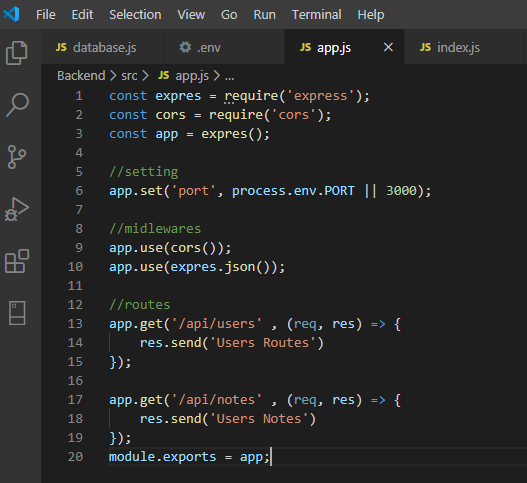
CREACION DE LAS VARIABLES DE ENTORNO

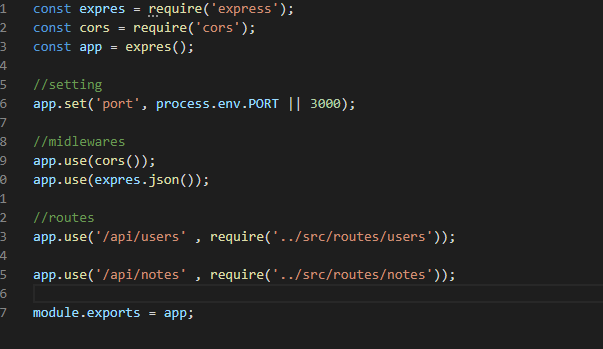


* Creamos el archivo **.env** en donde ponemos nuestra variable local

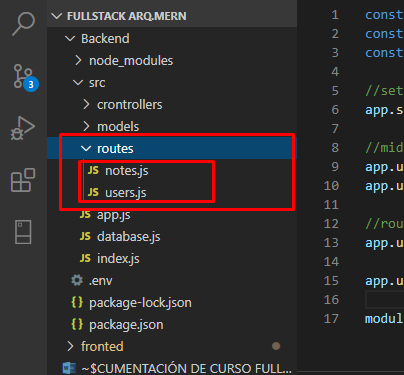




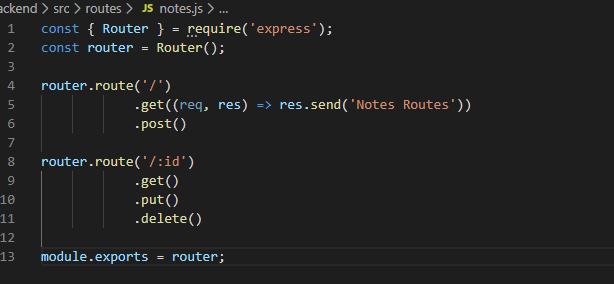
* Explicacion de la imagen: eso es como un **if** y con el **else**  si es el **process.env.MONGODB\_URI**  es **true** se hace lo que esta después del **?**  o si no entraría al **else** se ejecuta lo que es después del los **:**
* Para tener acceso a esa variable la tenemos por medio de **dotenv** que lo estamos requiriendo en la línea **1** en el archivo **index.js**
* Y lo utilizamos en el archivo **database.js** que es donde está la conexión a la base de datos.
* **CUARTA CLASE**
* En la clase 4 agregamos nuevas cosas al archivo **app.js** que son el **setting** el numero de la variable con su puerto que es **process.env.PORT** que es que coja de por defecto de el entorno o el 3000 que estamos poniendo, en **midlewares**  en la línea  **9 y 10** agregamos el **cors** que es para que tenga interacción con el servidor y el **express.json** es para que cuando el fronted cuando datos **json** los sepa identicar, y en **routes** dos diferentes rutas para poder navegar.
* **QUINTA CLASE**



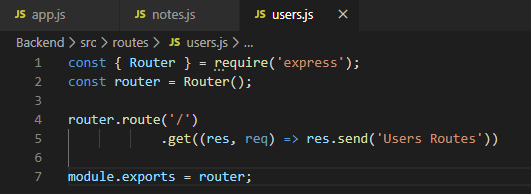
* Agregamos los routes aparta apara cada uno que son **users y notes**



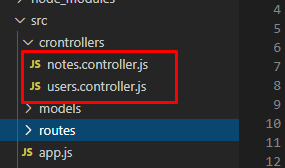
* También agregamos sus dos diferentes archivos para sus dos rutas que tenemos en la aplicación



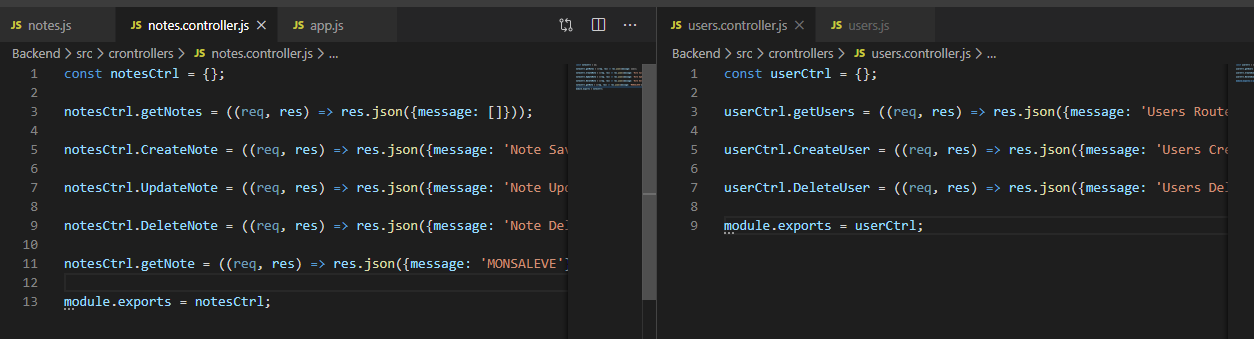
* Las rutas que tenemos en notes



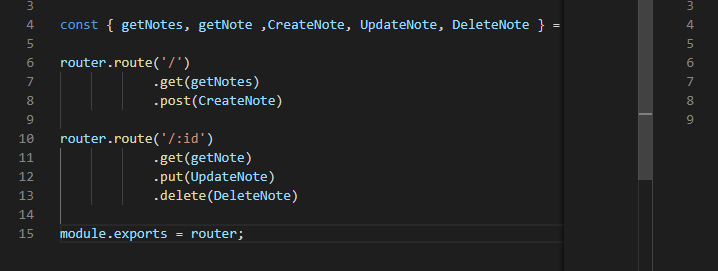
* Las rutas que tenemos en users.
* **SEXTA CLASE**



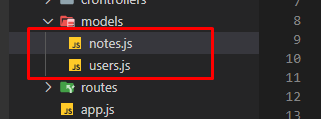
* Agregamos dos archivos ala carpeta controllers **note.controllers.js**  y **notes.controllers.js**.



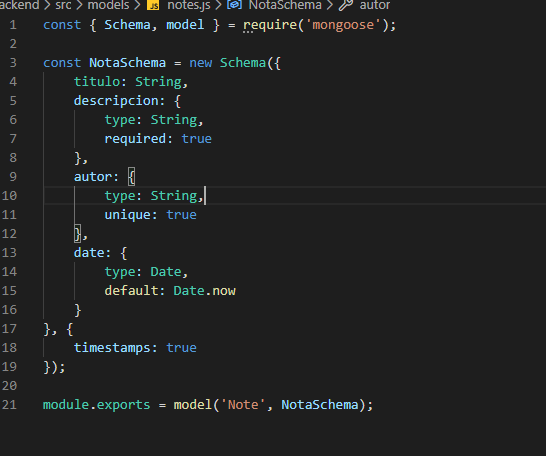
* Ponemos toda la lógica de cada de las peticiones HTTP más utilizadas en nuestra pagina web que sería **users** que esta al lado derecho y **notes** ala izquierda, exportamos los dos que serian **notesCtrl**  y **useCtrl** utilizamos en cada uno de sus routes de users y notes.



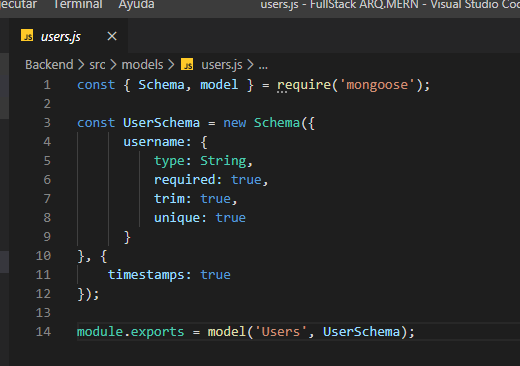
* Para utilizarlos seria requiriéndolos y por medio de una constante llamando cada uno de sus clase o propiedades que tiene el **notesCtrl** para el **router notes** .
* **SEPTIMA CLASE**



* Agregamos a la carpeta **models** dos archivos **notes.js** y **users.js.**



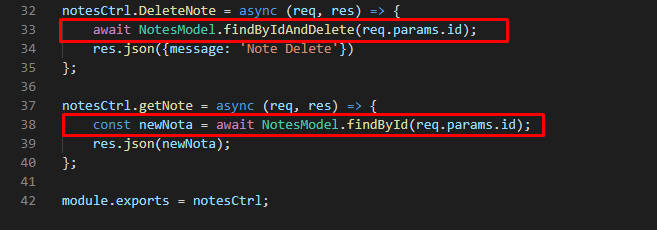
* Modelo de **notes.js**, que es un modelo exactamente es donde ponemos los datos o creamos los datos que va a tener la tabla que en este caso se va a llamar **NotaSchema** que seria la que estamos exportando y futuro lo vamos a utilizar, que tenemos en este modelo **5** datos que serían **título**, **descripción** también le estamos poniendo a ese campo que es **required** ósea que es un campo obligatorio, **autor** que le ponemos **unique** ósea que un dato que no se puede repetir, **date** y **timestaps** eso una función de **mongodb** para que cuando se crea un nuevo registro nos de la fecha y también cuando se actualiza un campo de la fila se de la fecha que se actualizo.



* Modelo de **users** este solo tiene un solo dato que **username** tiene casi los mismos opciones o configuraciones que están en el modelo anterior de la imagen la única diferencia es que tiene **trim** que es para que cuando en el imput puede eliminar espacios no necesarios.



* Ahora le ponemos toda la lógica para cada uno de nuestros controllers este es el notes.controllers.
  + **getNotes:** acá utilizamos nuestro modelo que esta **requerido** en la línea **6** con nuestro modelo utilizamos la función **find** que es una consulta para que nos traiga todos los datos de la tabla y su resulta la guardamos en una constante y por medio de **res.json** le enviamos la consulta que esta guardado en la constante **Notas.**
  + **CreateNote:** **req.body** nos trae los datos que nos esta mandando el cliente y los guardamos cada uno de los datos en una constante todos en general, creamos un nuevo modelo que es **new NotesModel** y lo guardamos en una constante que sería **NewNotas** y después con la nueva constante que tenemos utilizamos la función **save** que es para que se guarde los nuevos datos a la tabla **notes**  y con un res.json le enviamos un **message** indicando que se guardó su nueva nota**.**
  + **UpdateNote:** hacemos los mismo como en **createnote** que es guardar los datos del cliente que manda en una constante y que lo obtenemos por medio del **req.body**, utilizamos la función **findOneAndUpdate** que pide dos cosas el id de la fila que se va a actualizar y los datos nuevos que vamos a actualizar ya sea su título o descripción o autor y al finalizar le enviamos al cliente un **res.json** con su mensaje de nota actualizada.



* + **DeleteNote:** utilizamos la función **finfByAndDelete** que pide el id de la fila que se va a eliminar y con una res.json indicando nota eliminada.
  + **getNote:** este es el mismo que getNotes la diferencia es que en ves de **find** es **findById** que lo único que pide es el id de la fila para mostrar una sola fila.
* **Importante:** otra cosa muy importante es que a toda consulta que hacemos en necesario ponerle su **async – await**  ya que la consulta demora en traer los datos que le estamos pidiendo en la consulta.
* **OBTAVA CLASE:**
* creamos una aplicación ya por parte del **fronted** con **npx create-react-app fronted** con el nombre **fronted**

**NOTA:** error al ejecutar por primera vez el **npm start** la solución es bajar el react-scripts solución que he sacado de esta dirección <https://github.com/facebook/create-react-app/issues/6985>

* corremos el **npm start** y nos sale error una de las soluciones es correr el comando que esta en siguiente guion



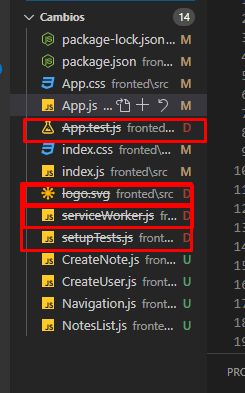
* Al terminar todo el instalador corremos el **npm install** [**react-scripts@2.1.8**](mailto:react-scripts@2.1.8) que es para que nos pueda correr el npm start

**MONGOD…**  es para ejecutar el servidor

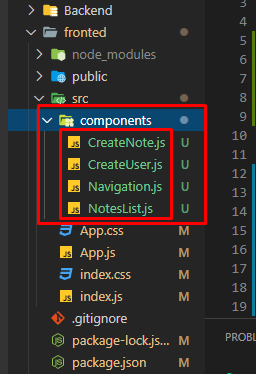
**MONGO…** es para poder interactuar con mongodb



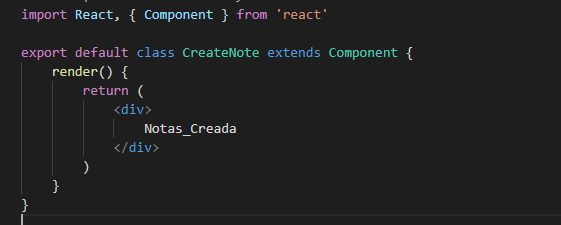
* después de crear el proyecto descargamos react-router-dom que es para poder navegar por los diferentes componentes



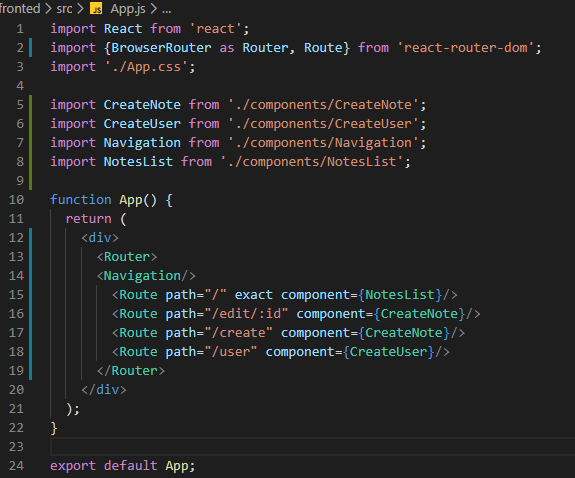
* eliminamos estos archivos en especifico ya que no son necesarios



* quedaría así el proyecto después de eliminar lo que comentamos anterior mente después creamos la carpeta **components** y creamos sus estos cuatros archivos que son los cuatro componentes principales para nuestro sitio web



* estructura básica para cada uno de los componentes



* importamos el react-router-dom que descargamos anterior mente y utilizamos su componente Router y también **Route** donde le ponemos su **ruta path** y que componente debe retornar.
* **NOVENA CLASE:**

