1. Teoria
2. Middlewares: aplicación global
3. Aplicacion:

Global.

A nivel ruta.

1. Introduccion a Express Validator
2. Exprexx validator avanzado
3. Validación completa con Express Validator
4. Notas de la clase

**Teoría:**

El req y el res son objetos, express analiza estos objetos cuando trabaja, pero como funciona?

Express divide todas las responsabilidades y objetivos. controlar si el usuario es administrador, dejar registro.

**MiddelWare** funsiones que funcionans en cadeda **entre** el pedido del cliente y la respuesta.

Las rutas que requieran middlewares —además del request y el response— deberán recibir un tercer parámetro: la función next. Next permite la ejecución en cadena de las funciones necesarias que debes ejecutarce antes de la respuesta.

Logueadomiddelware: controlaría si el usuario esta logeado por ejemplo.

Los middelware pueden cortar el proceso en el caso de que algo no este bien, agregando una capa de seguridad.

2.

**Middleeares globales,** o middlewares a nivel de la aplicación, son código que queremos que se ejecute durante toda la aplicación. Suena los recursos estáticos, son middleware globales para que las vistas puedan usarlos cuando quieran. También las configuraciones del routeo.



Las archivos de rutas tamien son middlewares.

App.use estamos diciendo que toda la app va a estar usando un middelwera, cada fsuncion recibe un req, res y next.

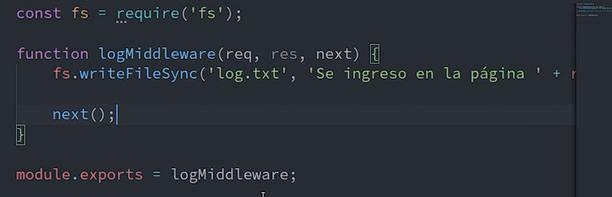
Que es next()? Es una callback qeyu se encargará de apilar todos los middlewares que apliquen a luna misma petición, ejecutará uno tras otros asta terminar. Cuando se ejecuten todos y haya salido todo bine , ejecutara el método del controlador que maneja la ruta

3.

**Global:**

Se creara en una carpeta a parte para lograr una organización, y luego se exportarán las funciones.

El archivo se llamará logMiddelware.js

res.url

Esto dejara un registro de los loging del usuario.

¿Como hacer para que el este middelwera atraviese todas las paginas? Dento de app.js requerimos el paquete.



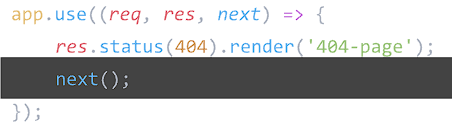
Y usamos app.use:



En el log.txt aparecerá la url donde entremos, pero en esete ejemplo la sobrescribirá. Si cambiamos por appendDileSync se soluciona

El error 404 es cuando el usuario entra a una ruta que no existe. Como lo aplicamos?

Atravez de un middlewera global, como solo se debe ejecutar si todas las rutas fallan, es importante que este simpore a lo ultimo.



**A nivel ruta:**

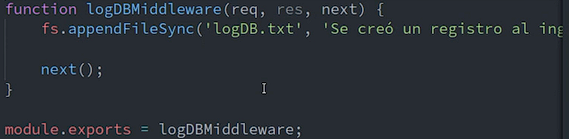
**los middlewares a nivel de ruta** son aquellos que se ejecutarán **única y exclusivamente** al ingresar a aquellas rutas que definamos previamente

en medio de la ruta pondremos el middleware:



Mejor si el middelware este en otro archivo y lo pasamos a la routa com tercer parámetros, en el medio de la ruta y la acción del controlador, ya que en algunos casos vamos a querer usarlo en varias rutas.



en la pagina” + req.url.

Dentro de el archivo de router lo requeriremos :



Metemos el let en medio de la ruta que queramos que use el middleware.



**Cuando la ruta pase por POST** se creará el archivo logDB.txt con el texto declarado.

4.

**Introducción a Express Validator:**

validacion de los datos enviados de un formulario, en la base de datos en email esta el numero de teléfono.

Express validator es una biblioteca de express que funciona como midellwware que se monta en una ruta y nos servirá para validar los campos del formulario y tambioen nos dara siertas alertas.

Primero: npm install express-validator

Segundo**: implementación en el archivo de rutas**

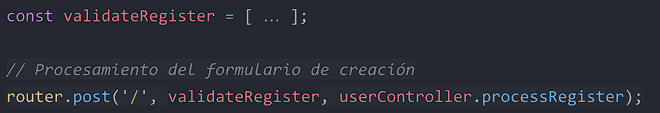
Primero vamos a la ruta donde usaremos express-validator, por ejemnplo en esta ruta de POST

Aquí requerimos express-valudator, con la destructurtacion de objetos solo llamaos a la función body.

Con body vamos a verificar si el campo este vacío o si el tipo de dato es el correcto.

Express nos pide que las validadicones estén dentro de una array de la siguiente forma

La fuincion body toma el nombre del campo que queremos valuidar. Y noEmpty() indicamos que el campo no debe estar vasio

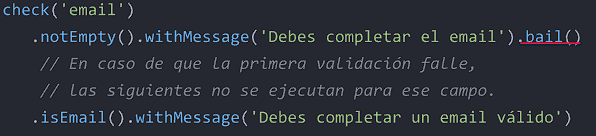
Por ultimo pasamos esta array en la ruta que queremos que la utilice, justo entre la ruta y el controlador.



Depsues del notEmpty() puedes poner una alerta para el usuario.



**Bail()**: en el caso de que queramos cortar la validación , poruqe hay un campo vasio y no tiene sentido validarlo usaremos bail():

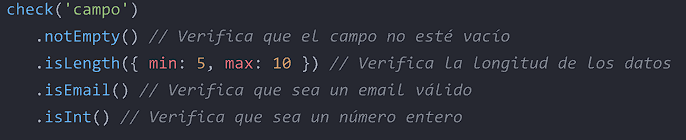
****

de no ser asi el usuario recivira todos las alertas juntas de error de validación.

url para ver los ejemplos.

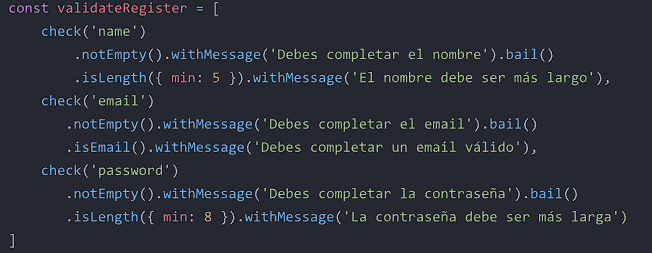
<https://github.com/codelando/dh-node-express-validator-introduccion>

hay muchas más validaciones: las mas comunes



El resto: <https://github.com/validatorjs/validator.js#validators>

Asi quedaría una validación ejemplo:

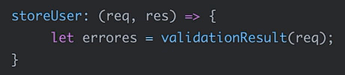


**LOS METODOS BODY Y CHECK DE EXPRESS VALIDATOR FUNSIONAN MUY IGUAL, ABMAS PUEDEN USARCE.**

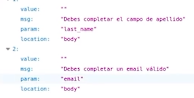
5.

**Express Validator avanzado:**

**Implementar el validador directamente en el controlador** de las rutas:

Lo requerimos, { validationResult }, iremos al método especifico y lo guardaremos en una variable local, asi:  lo importante, a la función le pasamos el req para que al info sea procesada.

Los errores se verán algo asi en el caso de que existan errores, de no ser asi será una array vacía:



msg es el string que le pusimos en el withMessage en el router



Volviendo a el controler, aremos un if para verificar si error esta vasio, si es asi cargara el usuario



Si existen errores, me renderisara la vista del formulario



Ahora falta darle alguna indicacion al ususario para que corrija los erorre.

A render le pasaremos un abjeto literal con el metodo llamado errors (puede ser cualquier combre) y como valor el error que ya tenemos, y del error vamos a pedirle el metodo array() que me va permitir convertir los errore en una array para poder enviarlos a la vista

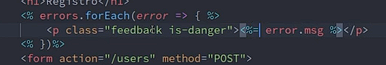


Tambien podemos usar el metodo mapped() que nos permite enviar los errores a la vista como un objeto. y el old : req.body lo explico mas abjo.



Asi ya los tenemos en la vista, vamos a vista.

Dentor de un blque de ejs hacemos un porEach mostrando todos los errores:

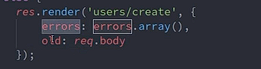


Pero si entro al fomrulario sin enviar nada al formulario dara error porque la variable errors no existe, entoces debo considerar si errors no existe. Meteremos todo el forEach dentro de un if que conprovará si exsite:

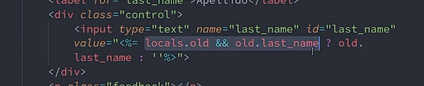


**LOCALS ES UNA FUNCION QUE TIENE EXPRESS QUE TIENE CUALQUIER COSA QUE ENVIEMOS A LA VISTA DENTRO DE UN OBJETO.**

Por ultimo como ahcemos para que queden los campos completados en el caso de que haya un error. Toda la info queda guardada en req.body, esto se lo enviaremos al render, ademas mandaremos el otra propiedad (old) con el req.body para otro uso (para ponerlo dentro de los campos del imput):



En la vista vamos al imput en cuastion y agremaos el value= dentro ponemos un ejs, dentro preguntramos si locals.old existe y si old last\_name tambioen tiene un valor, si es asi nos dara old.last-name y si no existe nos dará un string vacio “”



En el input de select es parecido:



Codigo usado de ejemplo: <https://github.com/codelando/dh-node-express-validator-introduccion>

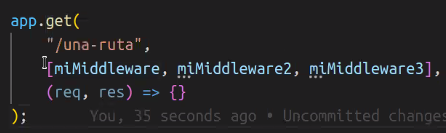
6.

7.

App.use global

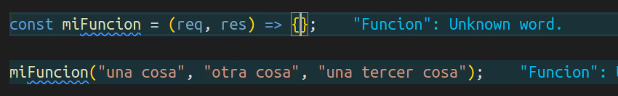
App.get por ejemplo es por ruta.

Con express validator se puede poner todos los middeleware que se quieran uno al lado del otro o todos dentro de una array

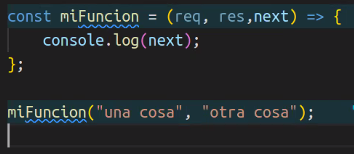
 ambos casos funsionan

La utilidad de pasarle muchas cosas al app.get es la organización del app.js

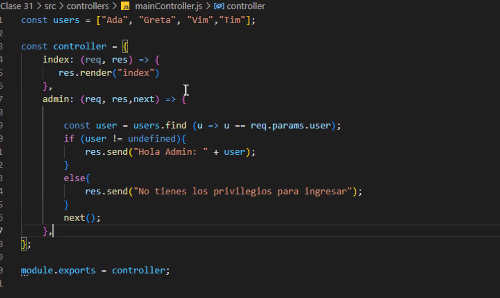
Si a una funcion le pasamaos mas parametros que los que le indicamos, no vamosa ponder usar el tercer parametro dentro de la funcion.

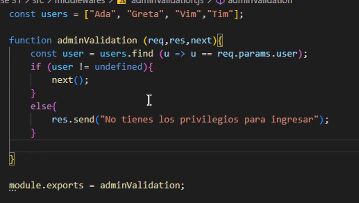


Y al reves, como en el siguienten caso, next va a ser undefine



npx degit [https://github.com/1022FSNSANCN02ARED/clases-resueltas/clase-31 clase-31](https://github.com/1022FSNSANCN02ARED/clases-resueltas/clase-31%20clase-31)





Qeudó el res.send

Video de una hora, mostrando cmo se hace un formulario:

