# Iniciando

Creo la Estructura:

Index.js para el código del servidor

Database.js para la conexión a la BD

SRC/

App: todo el código react

Public: React exporta un bundle.js ira aquí

Routes: Para las rutas del servidor

Models: Para el schema o modelo a guardar en la BD

Inicio un proyecto NODEJS

$npm init –yes

Ya puedo instalar modulos:

Inicio con express y trabajo en crear el servidor en index.js

Instalo los Middlewares:

Morgan: es un modulo q permite registrar o ver por consola las peticiones que están llegando desde el navegador, recibe un parametro

<https://www.npmjs.com/package/morgan>

Cuando establezca la conexión con React habrá intercambio de datos(json) y necesito entenderlo.

Para ello uso un middleware de express.json()

Creo las rutas usando un método de express.Router()

En public estará el index.html que llamará a mi app REACT, para ello debo configurar la ruta PUBLIC

# Base de datos

Instalo mongoose, este modulo nos permite conectarnos a la BD y al mismo tiempo definir como se verán los datos en ella.

Establezco la conexión en database.js y lo requiero en index.js

Ahora que la BD esta conectada puedo definir mi modelo/task.js

Teniendo mi schema ahora lo puedo usar en routes y a través de ese modelo puedo hacer consultas a la BD.

# FrontEnd – React

## Configuracion basica

El frontEnd lo guardare en App/

Primero instalo webpack quien traduce el código React y genera un bundle para producción:

$npm i -D webpack webpack-CLI

Luego de esto al mismo nivel de package.json creo el webpack.config.js:

module.exports={

entry:'./src/app/index.js',

output:{

path:\_\_dirname+'/src/public',

filename:'bundle.js'

}

}

Y un script para ejecutarlo:

"webpack":"webpack --mode development"

Ahora instalo React:

$npm i react react-dom

Creamos el primer componente en index.js pero esto da error al ejecutar el script webpack debido a que webpack solo traduce JS básico y React usa JSX(JS con HTML) para ello ocupamos BABEL:

$npm install -D babel-loader @babel/core @babel/preset-env @babel/preset-react

Creamos el archive de configuracion de BABEL, el internamente lo buscara .babelrc

{

"presets": ["env","react"]

}

Pero ahora tengo que decirle a webpack que lo use:

module.exports={

entry:'./src/app/index.js',

output:{

path:\_\_dirname+'/src/public',

filename:'bundle.js'

},

module:{

rules:[{

use:'babel-loader',

test:/\.js$/,

exclude:/node\_modules/

}

]

}

}

## FrontEnd

Trabajaremos con materializer para los estilos

<https://materializecss.com/>

Se puede agregar a webpack y usar directo en mis componentes pero para mas rápido se hará con CDN en el index.html

Tambien usare sus iconos

<https://materializecss.com/icons.html>

Trabajando en app/App.js usando el framework materilize

Luego de crear la parte visual debo capturar el evento que lanza el formulario en un método propio.

Capturo cada cambio del input:

<input onChange={this.handleChange} type="text" name="title" placeholder="Task Title" value={this.state.title} />

//cuando el usuario escriba algo lo capturamos

handleChange(event){

//obtengo name, value del input que este cambiando

const {name, value}=event.target

this.setState({

[name]:value

})

}

Antes tuve que definer un state:

this.state={

title:'',

description:''

}

Cuando se presione el botón, el form enviara al método add Task

<form onSubmit={this.addTask}>

Este metodo enviara la petición al servidor para agregar la tarea

// capturo el evento

addTask(event){

console.log(this.state)

//ENVIO la peticion al servidor

fetch('/api/task',{

method:'POST',

body:JSON.stringify(this.state),

headers:{

'Accept':'aplication/json',

'Content-Type':'application/json'

}

}).then(res => res.json())

.then( data => {

console.log(data)

//alert de materialize

M.toast({html:'Task Saved'})

this.setState({title:'',description:''})

})

.catch(err => console.error(err));

//previene que se recargue la pagina

event.preventDefault()

}

Ahora quiero listar todas las tareas:

//Consulto las tareas en la BD

fetchTask(){

//por defecto fech hace peticion GET

fetch('/api/task')

.then(res=> res.json())

.then(data=>{

this.setState({tasks:data})

console.log(this.state.tasks)

})

}

Cuando se monte el componente llamo a este método:

componentDidMount(){

this.fetchTask()

}

Ahora quiero eliminar una tarea y necesito pasar el parámetro, la forma de hacerlo es pasar como arrow function:

<button onClick={()=>this.deleteTask(task.\_id)}

//ELIMINO una tarea

deleteTask(id){

if(confirm('Are you sure you want to delete it?')){

fetch(`/api/task/${id}`,{

method:'DELETE',

headers:{

'Accept':'aplication/json',

'Content-Type':'application/json'

}

})

.then(res=>res.json())

.then(data=>{

console.log(data)

M.toast({html:'Task Deleted'})

this.fetchTask()

})

.catch(err=>console.error(err))

}

}

Ahora editar, para ello utilizare el mismo formulario y el mismo método de addTask con un if y agregando un id al state: