**MERN**

**React (app) (codigo de la bibliote y codigo de la interfaz)🡪 se transforma en codigo standard y se coloca en public**

**Creacion del proyecto**

**Carpetas y archivos**

1. **Proyecto/webpack.config.js**
2. **Proyecto/Src/database.js**
3. **Proyecto/Src/index.js**
4. **/src/models/task.js**
5. **/src/public/index.html**
6. **/src/routes/task.routes.js**
7. **/src/app(react)/index.js**
8. **/src/app(react)/app.js**
9. **/src/.babelrc**

**Instalacion de modulos**

**Para instalar nuevos modulos reiniciar el servidor**

1. **npm init –yes**
2. **npm i express**
3. **npm I nodemon –D**
4. **npm I morgan**
5. **npm I mongoose**
6. **npm I webpack –D**
7. **npm I webpack-cli**
8. **npm I react react-dom –D**
9. npm install --save-dev @babel/core
10. npm i @babel/preset-react -D
11. npm i @babel/preset-env -D
12. npm i babel-loader

**Es una direccion de donde esta el archive actual**

console.log(\_\_dirname);

**En linux**

**Sudo service mongod start**

1. **El Schema es exportado a las rutas, este objeto que exportamos es en lo que se basan los**
2. **Por lo visto al requererir mongoose desde database.js al requerirlo ya lo ejecutamos**

**No olvidar que estamos conectados a la base de datos a través del controlador en index.js 🡪 node en estos aspectos contiene todo el proyecto en su ejecución 🡪 todo esto sucede a través de express**

**En un punto esto solo levanta el servidor con la base de datos y conecta a los request del navegador, pero una vez conectado el servidor podemos dejarlo andando.**

**Babel traduce codigo JSX de REACT para que WEBPACK lo entienda**

**Webpack 🡪 traduce react a simple JS, CSS, HTML y no es necesario cuando se trabaje en un entorno real**

1. **En package .json 🡪 <scripts> 🡪** "webpack": "webpack --mode development " 🡪 **npm run webpack**
2. Nos crea un archive bundle.js dentro de public

**Webpack y REACT**

**Desde /app/index.js envio codigo de React 🡪 webpack lo convierte en bundle.js y a través de public/index.html Despliega bundle.js en el navegador**

**Por tanto leera bundle.js en lugar de app/index.js**

**Webpack puede complementarse con otras herramientas para distintas funcionalidades**

**-Tener un servidor en vivo,convertir codigo de zaz, convertir codigo de un motor de plantillas**

**-Luego de instalar babel 🡪 npm run webpack 🡪 tendremos instalado React dentro de bundle.js**

**(.babelrc y webpack.config.js lleva codigodentro de ellos que los conecta)**

**-Reiniciamos el servidor 🡪 npm run dev**

**ya el codigo JSX de app/index.js se ejecutara traducido en el navegador**

**(APP/INDEX.JS (REACT) ENVIA EL HOLA MUNDO🡪 BABEL (CONVIERTE JSX) 🡪 WEBPACK (COMPRIME CODIGO EN EL BUNDLE Y SE LO PASA A 🡪 PUBLIC/INDEX.HTML**

  "scripts": {

    "dev": "nodemon src/index.js",

    "webpack": "webpack --mode development --watch" //para que el proceso siga atento a cada cambio (como un nodemon para front-end)

  },

**REST API**

<https://www.youtube.com/watch?v=204v5QXHlmc> 18:41

Repositorio del proyecto 🡪 <https://github.com/FaztWeb/mern-stack-tasks>

**Materialize** 🡪 Framework de CSS

Se puede instalar en npm pero tomaría más tiempo configurarla en webpack

Igual npm nos permitiría trabajar la aplicación desde de los componentes

<https://materializecss.com/> 🡪 Get Started 🡪 Importamos el <link>a css y el <script>

SUBMIT 🡪

Cuando enviamos un formulario tenemos por defecto un evento que capturar (e)

class App1 extends Component {

    constructor(){

        super();

        this.state = {

            tittle:'',

     Description:''

        };

        this.addTask = this.addTask.bind(this); //Para que addTask lea el contenido de state

    }

    addTask(e){

        console.log(this.state);

        e.preventDefault();

    };

1. Creamos un constructor que contenga el **state** inicial de la aplicación
2. <form onSubmit={this.addTask}> 🡪 Al hacer submit del formulario acciona función
3. addTask 🡪 nos muestra por consola el estado del objeto actual (vacio ya que no recibe ningún parámetro)

Si alguna vez vemos mensajes por consola de navegador que desaparecen rápido cuando se recarga la pagina podemos ir a configuraciones 🡪 preserve log

**Capturar datos a través del estado de la aplicación**

    //Captura todos los cambios dentro de un input

        handleChange(e){

//extraigo el valor del cambio y nombre del objeto que sufrio el cambio.

            const { name, value  } = e.target;

//setState cambia el estado de una app de react

            this.setState({

[name] 🡪 tittle o description

[value] 🡪 el contenido que reciben en el change asignado al nuevo estado

               [name]: value              })

**EN HTML**

//en el componente, registra todos los cambios dentro de este campo

<input name='tittle' onChange={this.handleChange} type="text" placeholder="Task Tittle "/>

Ahora que estamos capturando los datos a través del estado de la aplicación

45:57

**FETCH** (Evento propio del navegador) 🡪 **Para enviar petición HTTP al servidor**

Ya que probamos anteriormente con postman que las rutas funcionas le enviamos los datos de esta manera

    addTask(e){

        //le doy la ruta donde envio los datos

        fetch('api/tasks',{//no hace falta agregar localhost:3000/api/tasks ya que react esta servida por el mismo servidor

            //con que metodo

            method: 'POST',

            //el contenido que envio, en este caso mi estado actual (convertido a JSON)

            body: JSON.stringify(this.state),

            headers: { //Aclara tipo de contenido y pide que lo acepten

                'Accept':'application/json',

                'Content-Type' : 'application/json'

            }

        })

           .then(res => console.log(res))

           //nos devuelve la respuesta de a que direccion hemos enviado esto y en teoria deberia haber guardado este dato en la base y coleccion correspondiente

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

Mas adelante tenemos que buscar la forma de que no cualquiera consulte las rutas con datos de mi app (**Authentication, json web tokens**

**TOAST 🡪 funcionalidad de materialize <script> que muestra un mensaje temporal en pantalla**

Mostrar tareas en una tabla 🡪

Para que apenas app sea montada la aplicación obtenga los datos

**METODO CONFIRM no funciono y esta muy bueno**

Ver bien el EDIT, PROBAR TODO, ESTUDIAR BIEN EL CODIGO, HACER GRAFICOS DE PEQUEÑOS PROCESOS

JSON WEB TOKENS 🡪 PROTECCION DE LOS DATOS EN RUTAS SECRETAS

SE USAN TOKENS DE VALIDACION O USUARIOS QUE SE LOGEAN

GRAPH QL

<https://www.youtube.com/watch?v=204v5QXHlmc> 01:12:00