# PROYECTO 1: App películas

## HTML Y CSS

**Borro** el fichero **app.css**.  
Pongo los **estilos** CSS en el fichero **index.css**.

Pego el contenido del body del html dentro del return del **app.js**.  
Modifico los elementos class por className.

## COMPONENTES

Creo el componente **Listado.js** y le pego todo el contenido del ‘content’. En el content, llamo al componente Listado con su etiqueta.

Creo el componente **Buscador.js** y le pego el contenido del aside, la parte del buscador.

Creo el componente **Crear.js** y le pego el contenido del aside, la parte de crear película.

## COMPONENTE Crear.js

Creo el estado **peli** con las propiedades **id, title y description** vacíos ‘’.

Hago un **e.preventDefault()** para que no envíe el form y recargue la página.

Por otro lado, **asigno** los **datos** del form a una variable y llamo al **setPeli**.

Por último, **muestro** en un div los **datos** de la peli creada **sólo si ésta tiene datos**.

## HELPERS

Creo en src la carpeta helpers. Aquí voy a guardar funcionalidades que no sean exclusivas de un componente, como por ejemplo, guardar en localStorage.

/\*  Guardar en el localStorage. Recibe la lista y el elem a guardar. \*/

export const GuardarStorage = (lista, elem) => {

    /\* Conseguir los datos de localStorage \*/

    let elementos = JSON.parse(localStorage.getItem(lista));  
 /\* JSON.parse Convierte JSON a objeto \*/

    /\* Si elementos no es un array, lo creo. \*/

    if(!Array.isArray(elementos)){

        elementos = [];

    }

    /\* Añado al array el elemento nuevo \*/

    elementos.push(elem);

    /\* Guardo en localStorage el array modificado \*/

    localStorage.setItem(lista, JSON.stringify(elementos));

    /\* OPCIONAL: Devolver objeto \*/

    return elem;

}

## COMPONENTE Listado.js

Creo un **estado** en App.js para la **lista** de películas. Lo creo ahí porque necesito que se comunique entre los distintos componentes (Crear, Listado,…).  
A la hora de llamar a los componentes, le paso **2 props**: el **estado de lista** de películas y el **setLista** para que se pueda llamar desde dentro del componente.

Creo un **useEffect** para que **cuando cargue** el **componente**, se carguen las películas y me **actualice** el **estado** de **lista** de películas.

## Modificar COMPONENTE Crear.js

Como he puesto Listado como estado global (en App.js), ahora al crear una peli, además de actualizar el estado peli y guardar en localStorage, necesito actualizar el estado listado.

setPeli(peliNueva);

setLista( (estado\_previo) => {

    return [...estado\_previo, peliNueva];

});

GuardarStorage('pelis', peliNueva);

## BORRAR películas: modifico componente Listado.js

<button className="delete" onClick={ () => borrarPeli(peli.id) }>

Borrar </button>

const borrarPeli = (id) => {

    /\* Obtengo el listado de películas \*/

    let pelis = getLista();

    /\* Busco la que quiero eliminar. Creo una copia del array sin ella \*/

    let pelis\_res = pelis.filter( (peli) => peli.id !== parseInt(id) );

    /\* La elimino del estado listado \*/

    setLista(pelis\_res);

    /\* Actualizo localStorage \*/

    localStorage.setItem('pelis', JSON.stringify(pelis\_res));  
}

## EDITAR películas: modifico componente Listado.js

Creo un estado editar que inicialmente valga 0.

const [editar, setEditar] = useState(0);

<button className="edit" onClick={ () => setEditar(peli.id) }>

Editar </button>

Justo debajo de los botones, pongo un div condicional, que si el estado editar es igual al id de la película, me aparezca un componente que voy a crear, llamado Editar.js, que va a tener el formulario para editar la película.  
Le paso como props: la **peli** que estoy editando, **setEditar**, **setLista** y **getLista**.

## EDITAR películas: componente Editar.js

Tiene un formulario, el cual al darle a submit nos lleva a una función:

const editarPeli = (e, id) => {

    e.preventDefault();

    /\* Consigo la lista de películas \*/

    const pelis = getLista();

    /\* Conseguir los datos del formulario para actualizar la peli \*/

    let peliNueva = {

        id,

        title: e.target.title.value,

        description: e.target.description.value

    }

    /\* Busco la peli a modificar \*/

    const indice = pelis.findIndex(peli => peli.id === id);

    /\* Modifico la película en el listado de pelis. \*/

    pelis[indice] = peliNueva;

    /\* Actualizo estado lista \*/

    setLista(pelis);

    /\* Actualizo estado editar \*/

    setEditar(0);

    /\* Guardo en localStorage \*/

    localStorage.setItem('pelis', JSON.stringify(pelis));

}

## COMPONENTE Buscador.js

Recibe las props **lista** y **setLista**.  
Creo el **estado búsqueda**, **estado encontrado**, y un **useEffect** que llama a la función buscarPeli.

const buscarPeli = () => {

    /\* Filtro para encontrar pelis \*/  
    let resultado = lista.filter( (peli) => {  
        return peli.title.toLowerCase().includes(busqueda.toLowerCase())  
    })  
  
    /\* Si no encuentra o no ha escrito nada, muestra localStorage. \*/  
    if(resultado <= 0 || busqueda.length <=0){  
        resultado = JSON.parse(localStorage.getItem('pelis'));  
        setEncontrado(false);  
    }  
    else{  
      setEncontrado(true);  
    }  
      
    setLista(resultado);  
}

onChange={(e) => setBusqueda(e.target.value)}

# PROYECTO 2: Portafolio

Creo la carpeta components.  
Creo componentes: Contacto, Curriculum, Error, Inicio, Portafolio, Servicios.

Creo la carpeta router. Dentro creo el componente Rutas.

Creo la carpeta components/layout. Dentro creo los componentes: Header, Nav y Footer.

Creo la carpeta data. Dentro creo trabajos.js que contiene un array de objetos con los proyectos que he realizado.

Creo la carpeta public/images donde voy a meter una imagen para cada proyecto.

Creo el componente Proyecto dentro de components.

Creo el componente ListadoTrabajos dentro de components.

# PROYECTO 3: Blog

1. **Instalaciones previas**

* **NodeJS**: Descargar e instalar NodeJS LTS.
* **MondoDB (+ Servicio y Compass)**: Descargo de su web el Community Server.
* **Postman**: Descargar e instalar de su web.

1. **Crear proyecto en node**

* Creo una carpeta en mi equipo.
* Abro la carpeta en VSCode.
* En la consola ejecuto **npm init**

1. **Dependencias**

* **npm i express --save** Framework para manejar peticiones HTTP (GET, POST , PUT).
* **npm i mongoose --save**  Librería de MongoDB que nos aporta muchas funcionalidades.
* **npm i multer --save** Para subir archivos.
* **npm i validator --save** Para validar datos.
* **npm i cors --save** Para hacer peticiones entre dominios cruzados.
* **npm i nodemon --save-dev** Watcher de cambios, para no parar y arrancar el server.

--save guardo la dependencia dentro del proyecto.  
--save-dev guardo la dependencia dentro del proyecto como devDependencia.

1. **Configuración previa**

En el package.json en la parte de scripts voy a crear un nuevo comando para ejecutar en nodemon el archivo principal de la aplicación.



1. **Arrancar el proyecto**

* Para arrancar en nodemon: **npm start**
* Para arrancar sin nodemon: **node index.js**

1. **Conectar BBDD con el proyecto Node**

* En la raíz del proyecto, creo la carpeta database
* Dentro creo connection.js

const mongoose = require('mongoose')

mongoose.set('strictQuery',true)

const connection = async (url, bbdd) => {

    try{

        await mongoose.connect(`mongodb://${url}/${bbdd}`)

        console.log('Conectado a BBDD')

    }

    catch(error){

        console.log(error)

        throw new Error('Error al conectar a la BBDD')

    }

}

module.exports = { connection }

* Importo la conexión a la BBDD en el index.js

const { connection } = require('./database/connection')

//Inicializar app

console.log('App de node inicializada.')

//Conectar a BBDD

const url = '127.0.0.1:27017'

const bbdd = 'blog'

connection(url, bbdd)

1. **Crear servidor Node**

A falta de crear las rutas, me tiene que quedar así el index.js

const { connection } = require('./database/connection')

const express = require('express')

const cors = require('cors')

//Inicializar app

console.log('App de node inicializada.')

//Conectar a BBDD

const url = '127.0.0.1:27017'

const bbdd = 'blog'

connection(url, bbdd)

//Crear servidor Node

const app = express()

//Configurar CORS

app.use(cors())

//Convertir body a JS

app.use(express.json())

//Crear rutas

// Escuchar peticiones HTTP

const puerto = 3900

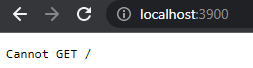
app.listen(puerto, () => {

    console.log(`Servidor corriendo en el puerto ${puerto}.`)

})

1. **Visualizar el proyecto**

Voy a la web y escribo localhost:3900



Veo esto porque no tengo rutas asociadas aún

1. **Crear rutas**

Voy a crear 2 rutas: la ruta ‘/’ y la ruta ‘/probando’.

//Crear rutas

app.get('/', (req,res) => {

    return res.status(200).send( '<h1>Bienvenido a la API REST</h1>' )

})

app.get('/probando', (req,res) => {

    console.log('Se ha ejecutado el endpoint PROBANDO.')

    return res.status(200).json(

        [

            { nombre: 'Kevin', apellidos: 'Arcas Monteiro', edad: 29 },

            { nombre: 'Alejandra', apellidos: 'Ramos Fdez', edad: 27 }

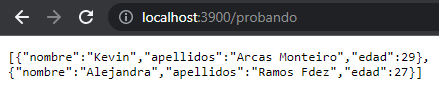
        ]

    )

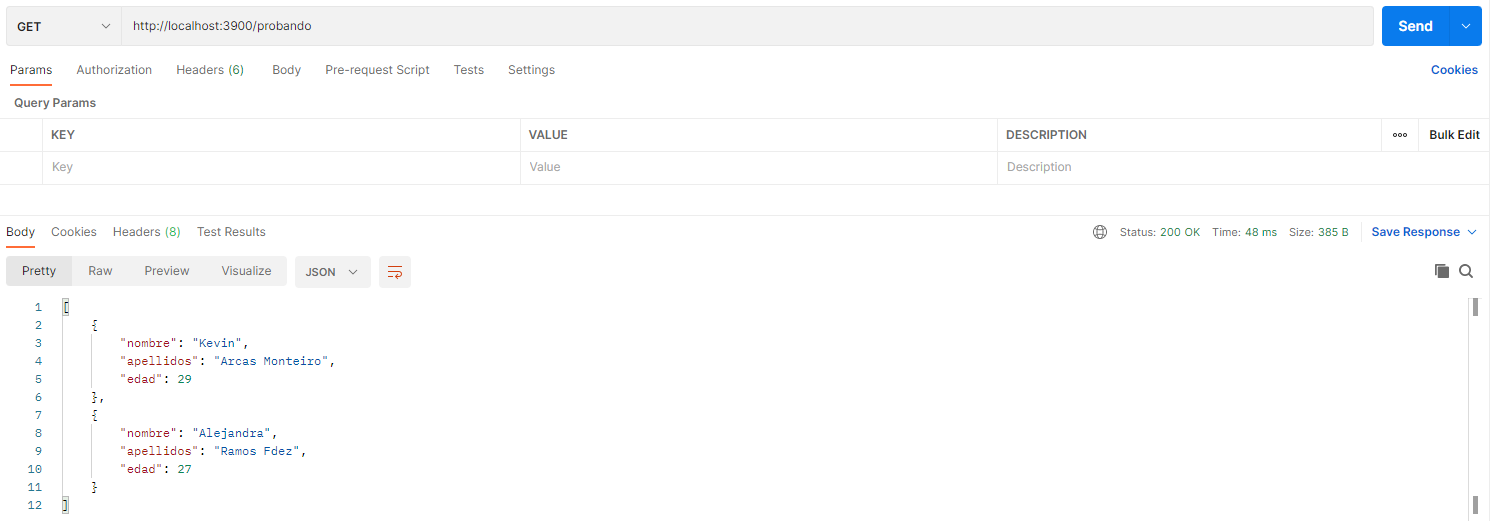
})

La ruta ‘/’ manda html, para ello pongo **send**.

  
  
La ruta ‘/probando’ manda json, por eso en lugar de send utilizo **json**.



1. **Probando la api con Postman**



Muestra los datos que devuelve la api.

# PROYECTO 4: Red social

Asasassasa

# FUNCIONES

class Toggle extends React.Component {  
 constructor(props) {  
 super(props);  
 this.state = {isToggleOn: true};  
 }  
}



this.setState(prevState => ({  
 isToggleOn: !prevState.isToggleOn

}));



cambioInput = event => {  
 this.setState({temperatura: event.target.value});  
}

…

<input type="text" value={this.state.temperatura} onChange={this.cambioInput} />