

mongoDB

Bienvenidos a MONGODB

Construiremos un divertido todo list app que almacene las tareas

Prográmate
Academy

POWERED BY SIMPLON.CO By Educamás

Instalaciones Entorno de desarrollo

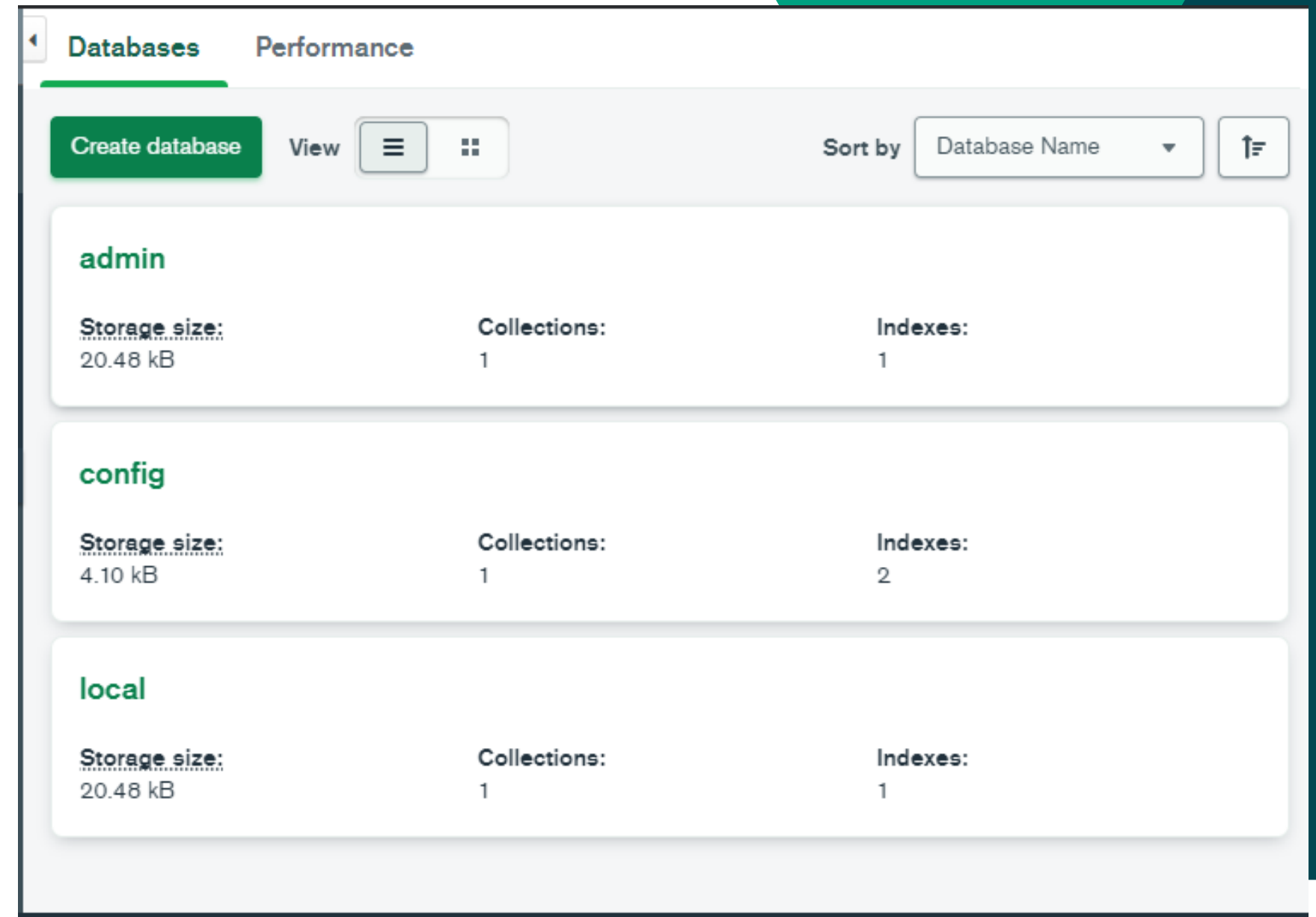
1. Node.js
2. React
3. MongoDB local
4. Express.js y Mongoose

Parte 1



Debes crear un proyecto en React llamado app
al instalar MongoDB debes configurar el
servidor local

1. crear una nueva conexión
2. con el comando `mongodb://localhost`



Parte 2

Debemos crear nuestra carpeta de backend instalar express.js y mongoose los comandos son los siguientes dentro de la carpeta backend

1. `npm init`
2. `npm install --save express`
3. `npm install --save mongoose`

```
MINGW64:/c/Users/User/Desktop/to-do-list/app/backend
Is this OK? (yes) yes
User@DESKTOP-08TC4RL MINGW64 ~/Desktop/to-do-list/app/backend
$ npm install --save express

added 50 packages, and audited 51 packages in 20s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

User@DESKTOP-08TC4RL MINGW64 ~/Desktop/to-do-list/app/backend
$ npm install --save mongoose

added 28 packages, and audited 79 packages in 6s

6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

User@DESKTOP-08TC4RL MINGW64 ~/Desktop/to-do-list/app/backend
$ |
```

REPASO

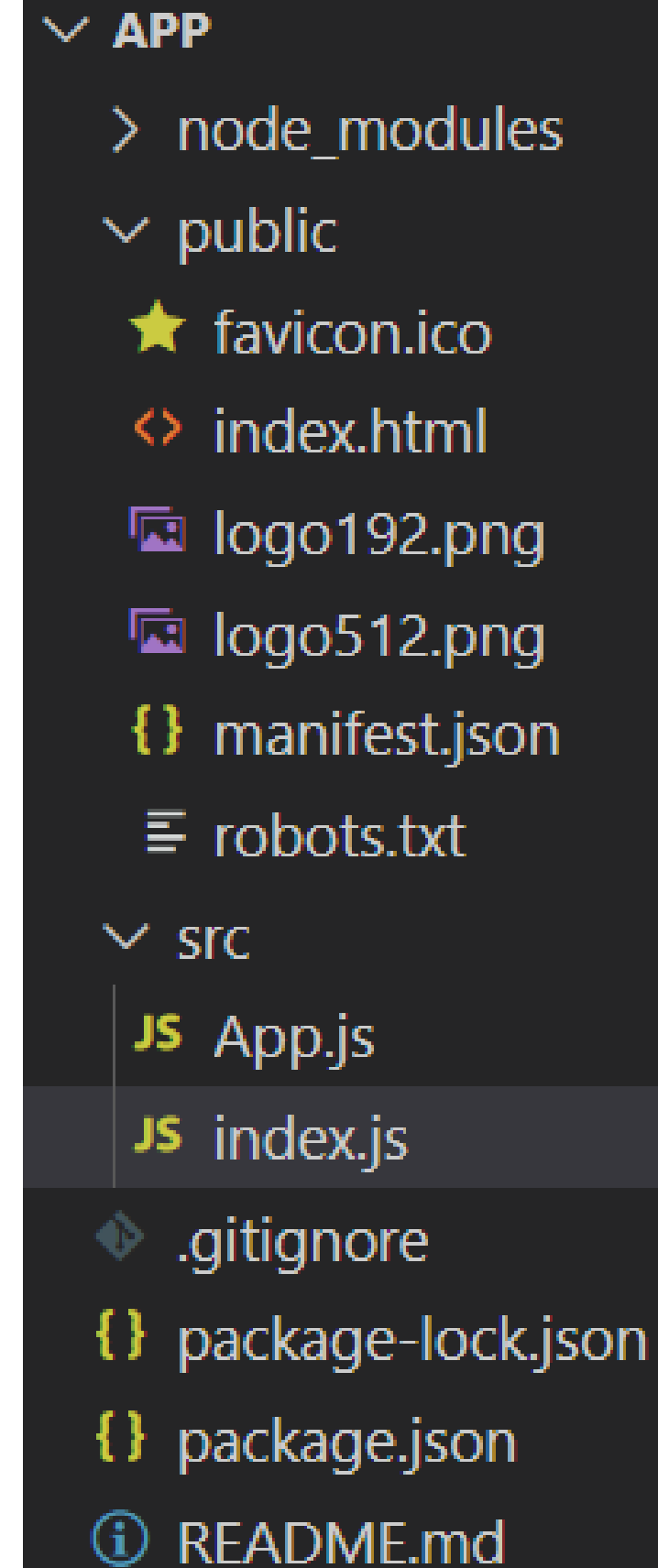


Algunos conceptos que debes
repasar o investigar

1. JSX
2. Componentes
3. Entender la estructura de nuestra APP
4. Props
5. Ciclo de vida de los métodos
6. Estados
7. Sistema de Hooks
8. UseSatate
9. UseEffect
10. The key Prop

PREPARA TU ENTORNO DE DESARROLLO

Borra todos los
archivos de SRC y en
esta ruta crea un
archivo index.js y la
convencion App.js



```
▼ APP
  > node_modules
  ▼ public
    ★ favicon.ico
    <> index.html
    🖼 logo192.png
    🖼 logo512.png
    {} manifest.json
    ≡ robots.txt
  ▼ src
    JS App.js
    JS index.js
    📄 .gitignore
    {} package-lock.json
    {} package.json
    ⓘ README.md
```

Parte 3

Creación del Front-End

DISEÑO

Pensar el Front

45°

To-Do-app

Enter something to do..

+

test #1



test #2



test #3



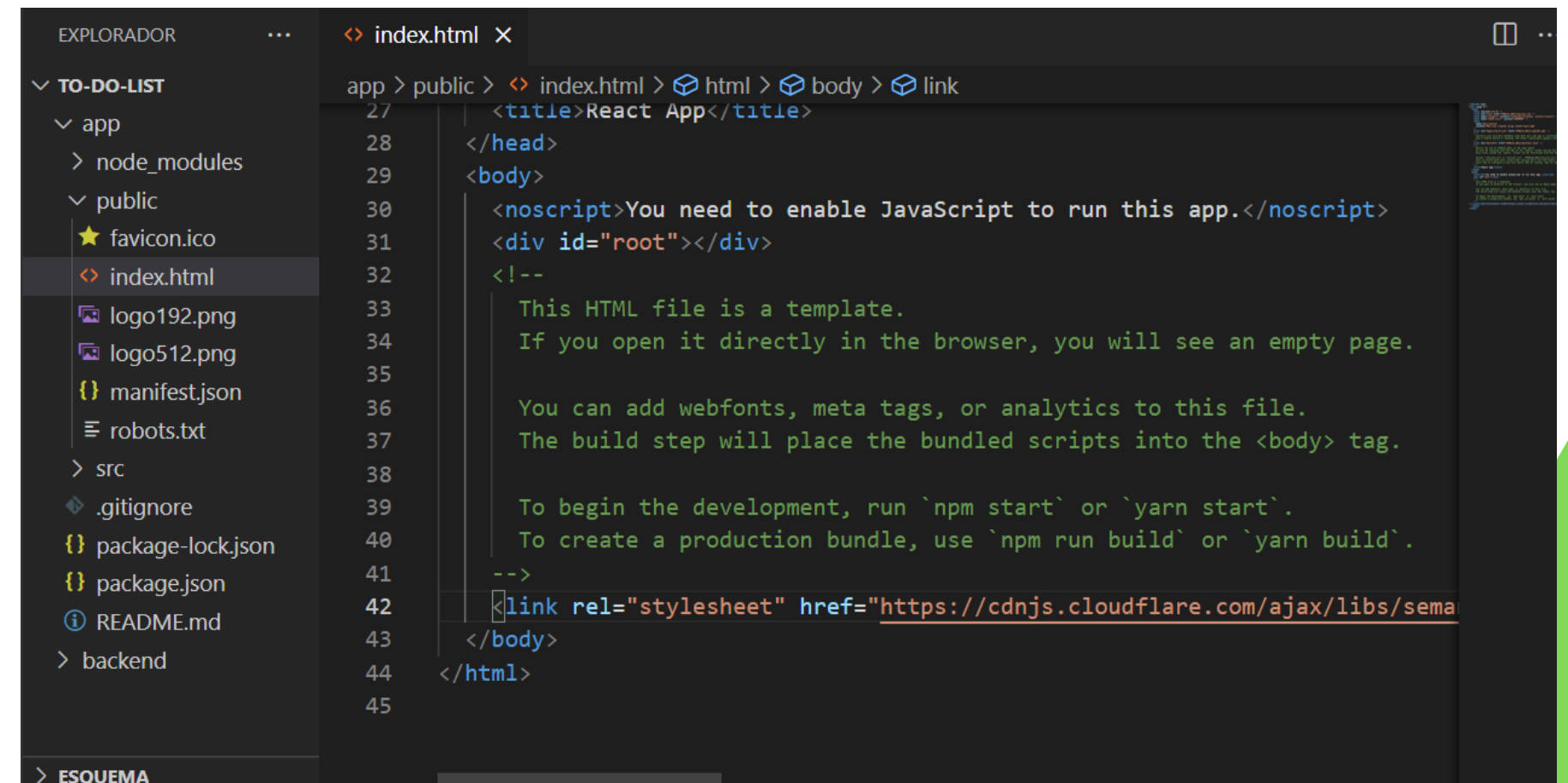
45°

Prográmate
Academy

POWERED BY SIMPLON.CO By Educamás

SEMANTIC UI

Busca en google
semantic ui cdn
y copiar el link css
dentro de public en
index.html lo pegas
antes de cerrar el body



```
EXPLORADOR  ...  <> index.html X
v TO-DO-LIST
v app
  > node_modules
  v public
    ★ favicon.ico
    <> index.html
    🖼 logo192.png
    🖼 logo512.png
    {} manifest.json
    📄 robots.txt
  > src
  💎 .gitignore
  {} package-lock.json
  {} package.json
  ⓘ README.md
  > backend
> ESQUEMA

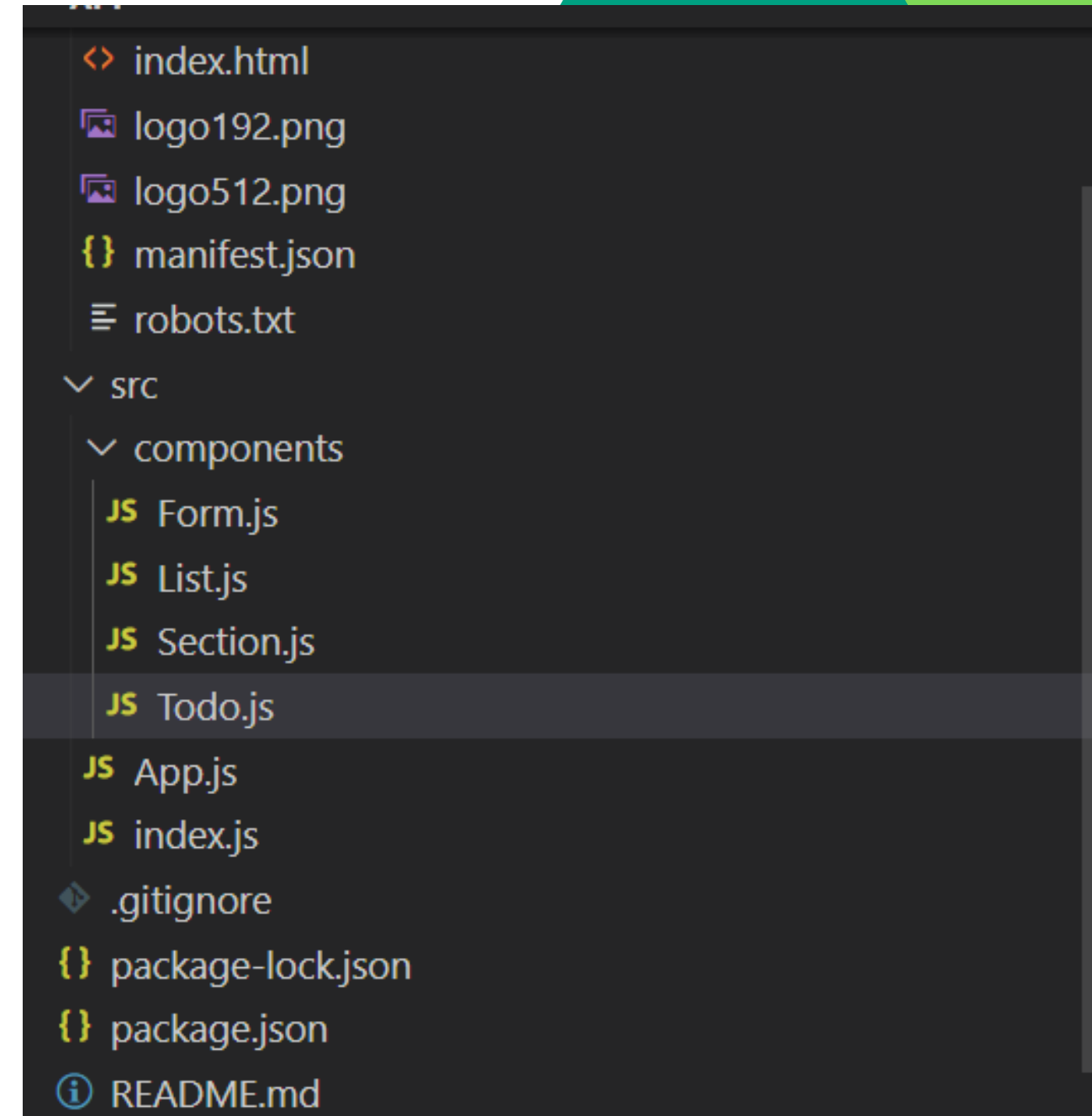
app > public > <> index.html > 📄 html > 📄 body > 📄 link
27  <title>React App</title>
28  </head>
29  <body>
30    <noscript>You need to enable JavaScript to run this app.</noscript>
31    <div id="root"></div>
32    <!--
33      This HTML file is a template.
34      If you open it directly in the browser, you will see an empty page.
35
36      You can add webfonts, meta tags, or analytics to this file.
37      The build step will place the bundled scripts into the <body> tag.
38
39      To begin the development, run `npm start` or `yarn start`.
40      To create a production bundle, use `npm run build` or `yarn build`.
41    -->
42    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.4.2/semantic.min.css">
43  </body>
44 </html>
45
```

PENSEMOS EN LOS
COMPONENTES

FORM.JS
LIST.JS
SECTION.JS
TODOS.JS

Recuerda:
en index.js llamas el componente App para que
renderize en index.html mediante root
asi:

1. import React from "react";
2. import ReactDOM from "react-dom";
3. import App from "./App";
- 4.
5. ReactDOM.render(<App />, document.querySelector("#root"));



F O R M

Vamos a crear el
cuerpo de nuestro
componente Form

```
1. import React from "react";
2.
3. const Form = () => {
4.   return (
5.     <form className="ui form" onSubmit={handleFormSubmit}>
6.       <div className="ui grid center aligned">
7.         <div className="row">
8.           <div className="column five wide">
9.             <input
10.               value={inputValue}
11.               onChange={handleInputChange}
12.               type="text"
13.               placeholder="Enter something to do..."
14.             />
15.           </div>
16.
17.           <div className="column one wide">
18.             <button type="submit" className="ui button circular icon green"><i
19.               className="white plus icon"></i></button>
20.           </div>
21.         </div>
22.       </form>
23.     );
24.   };
```

F O R M

Vamos a convertir la
entrada en un componente
para que sea el
controlador y agregar
funcionalidad
recuerda importar
useState y documenta
¿por qué?

```
1. import React, { useState } from "react";
2.
3. const Form = ({ addTodo }) => {
4.   const [inputValue, setInputValue] = useState("");
5.
6.   const handleInputChange = (e) => {
7.     setInputValue(e.target.value);
8.   };
9.
10.  const handleFormSubmit = (e) => {
11.    e.preventDefault();
12.
13.    if(inputValue.trim() === "") return;
14.
15.    addTodo({ title: inputValue, completed: false });
16.    setInputValue("");
17.  };

```

SECCIONES

Vamos a crear secciones en nuestro to-do-list para crear espaciós correctos de manera semàntica

Para ello modificaremos nuestros App.js

```
1. IMPORT REACT FROM "REACT";
2. IMPORT FORM FROM
   "./COMPONENTS/FORM";
3. IMPORT SECTION FROM
   "./COMPONENTS/SECTION";
4. CONST APPTITLE = "TO-DO-APP";
5.
6. CONST APP = () =>{
7.   RETURN <DIV CLASSNAME="UI
   CONTAINER CENTER ALIGNED">
8.     <SECTION>
9.       <H1>{APPTITLE}</H1>
10.    </SECTION>
11.
12.    <SECTION>
13.      <FORM/>
14.    </SECTION>
15.
16.  </DIV>;
17.}
18.
19. EXPORT DEFAULT APP;
20.
```

SECCIONES

En nuestro componente
Section.js

¿Por que crees que estamos
desestructurando el
componente mediante
children?

```
1. IMPORT REACT FROM
   "REACT";
2.
3. CONST SECTION = ({
   CHILDREN }) => {
4.   RETURN (
5.     <DIV STYLE={{ MARGIN:
   "50PX" }}>
6.       {CHILDREN}
7.     </DIV>
8.   );
9. };
10.
11 EXPORT DEFAULT SECTION;
```

LIST PARTE 1

Crea el componente List
y observa como vas
dándole estilos
mediante la librería
también documenta
esta sección para saber
pasa y identa el código

```
1. import React from "react";
2.
3. const List = () => {
4.   return(
5.     <div className="ui grid center aligned">
6.       <div className="row">
7.         <div className="column five wide">
8.           <h1>Test</h1>
9.         </div>
10.
11.        <div className="column two wide">
12.          <button className="ui button circular icon green"><i
            className="white check icon"></i></button>
13.        </div>
14.        <div className="column two wide">
15.          <button className="ui button circular icon red"><i
            className="white remove icon"></i></button>
16.        </div>
17.      </div>
18.    </div>
19.  );
20. };
21.
22. export default List;
23.
```

LIST PARTE 2 Y COMPONENTE TODO

Bien ahora lo que haremos es poder editar los
items de nuestra lista

para ello crearemos nuestro componente Todo

1 Observa el componente Todo

2 Observa el componente List

verifica que paso con estos dos componentes y
comenta el codigo

```
index.html JS List.js JS Form.js JS Todo.js JS App.js
app > src > components > JS Todo.js > [e] todo
1  import React from "react";
2
3  const todo = () => {
4    return (
5      <div className="row">
6        <div className="column five wide">
7          <h1>Test</h1>
8        </div>
9
10       <div className="column two wide">
11         <button className="ui button circular icon green"><i cla
12       </div>
13       <div className="column two wide">
14         <button className="ui button circular icon red"><i classl
15       </div>
16     </div>
17   );
18 };
19
20 export default todo;
```

```
index.html JS List.js JS Form.js JS Todo.js
app > src > components > JS List.js > ...
1  import React from "react";
2  import Todo from "../Todo";
3
4  const List = () => {
5    return(
6      <div className="ui grid center aligned">
7        <Todo/>
8      </div>
9    );
10 };
11
12 export default List;
```


FUNCIONALIDAD DE INPUT DEL TODO

De nuevo con useState
en este caso debes analizar la funcionalidad y
como se implementa en el Return

jejeje debes rescribir el código pues en
mayuscula todo no funciona

UPSI!!!!

```
1. import React, {useState} from "react";
2.
3. const TODO = () => {
4.   const [isEditing, setIsEditing] =
     useState(false);
5.
6.   const handleDivDoubleClick = () => {
7.     setIsEditing(true);
8.   };
9.   return (
10.    isEditing ?
11.    <input />:
12.    <div classname="row" onDoubleClick=
      {handleDivDoubleClick}>
13.      <div classname="column five wide">
14.        <h1>TEST</h1>
15.      </div>
16.
17.      <div classname="column two wide">
18.        <button classname="ui button
          circular icon green"><i classname="white
            check icon"></i></button>
19.      </div>
20.      <div classname="column two wide">
21.        <button classname="ui button
          circular icon red"><i classname="white remove
            icon"></i></button>
22.      </div>
23.    </div>
24.  );
25.};
26.
27. export default TODO;
```

DISEÑAR EL INPUT Y ESCUCHAR EL KEY CODER

Analiza el código y observa que esta haciendo
el evento `handleinputkeydown`
y de nuevo comenta

```
1. import React, {useState} from "react";
2.
3. const Todo = () => {
4.   const [isEditing, setIsEditing] = useState(false);
5.
6.   const handleDivDubleClick = () => {
7.     setIsEditing(true);
8.   };
9.
10.  const handleInputKeyDown = (e) => {
11.    const key = e.keycode;
12.
13.    if(key === 13 || key === 27){
14.      setIsEditing(false);
15.    }
16.
17.  };
18.
19.  return (
20.    isEditing ?
21.    <div className="row" onDoubleClick={handleDivDubleClick}>
22.      <div className="column seven wide">
23.        <div className="ui input fluid">
24.          <input onKeyDown={handleInputKeyDown}/>
25.        </div>
26.      </div>
27.    </div>:
28.    <div className="row" onDoubleClick={handleDivDubleClick}>
29.      <div className="column five wide">
30.        <h1>Test</h1>
31.      </div>
32.
33.      <div className="column two wide">
34.        <button className="ui button circular icon green"><i className="white check icon"></i>
35.      </div>
36.      <div className="column two wide">
37.        <button className="ui button circular icon red"><i className="white remove icon"></i>
38.      </div>
39.    </div>
40.  );
41.};
42.
43. export default Todo;
```

VAMOS A HACER UN CONTROLADOR PARA EL INPUT

En este paso debes recordar los Props
copia el siguiente código y revisa como se
desestructura title (revisa lo que esta en
amarillo) y recuerda llamar la propiedad en
List.js asi:

```
import React from "react";
import Todo from "../Todo";

const List = () => {
  return(
    <div className="ui grid center aligned">
      <Todo title="Test # 1"/>
    </div>
  );
};

export default List;
```

```
1. import React, {useState} from "react";
2.
3. const Todo = ({title}) => {
4.   const [Value, setvalue] =useState(title)
5.   const [isEditing, setIsEditing] = useState(false);
6.
7.   const handleDivDubleClick = () => {
8.     setIsEditing(true);
9.   };
10.
11.   const handleInputKeyDown = (e) => {
12.     const key = e.keyCode;
13.
14.     if(key === 13 || key === 27){
15.       setIsEditing(false);
16.     }
17.   };
18. };
19.
20. const handleInputOnChange =(e) => {
21.   setvalue(e.target.value);
22. };
23.
24. return (
25.   isEditing ?
26.   <div className="column seven wide">
27.     <div className="ui input fluid">
28.       <input
29.         onChange={handleInputOnChange}
30.         onKeyDown={handleInputKeyDown}
31.         autoFocus={true}
32.         value={Value}
33.       />
34.     </div>
35.   </div> :
36.   <div className="row" onDoubleClick={handleDivDubleClick}>
37.     <div className="column five wide">
38.       <h1>{Value}</h1>
39.     </div>
40.
41.     <div className="column two wide">
42.       <button className="ui button circular icon green"><i className="white check icon"></i></button>
43.     </div>
44.     <div className="column two wide">
45.       <button className="ui button circular icon red"><i className="white remove icon"></i></button>
46.     </div>
47.   </div>
48. );
49. };
50.
51. export default Todo;
52.
53. export default Todo;
```

ARREGLANDO LA TECLA ESC

En este paso debes evaluar ¿por qué?
Modificamos el condicional
de `handleInputKeyDown`
justifica la respuesta

observa los cambios de organización en el
Return

```
1. import React, { useState } from "react";
2.
3. const Todo = ({ title }) => {
4.
5.   const [isEditing, setIsEditing] = useState(false);
6.   const [Value, setValue] = useState(title)
7.   const [tempValue, setTempValue] = useState(title);
8.
9.   const handleDivDubleClick = () => {
10.     setIsEditing(true);
11.   };
12.
13.   const handleInputKeyDown = (e) => {
14.     const key = e.keyCode;
15.
16.     if (key === 13) {
17.       setValue(tempValue);
18.       setIsEditing(false);
19.     } else if (key === 27) {
20.       setTempValue(Value);
21.       setIsEditing(false);
22.     }
23.   };
24.
25.   const handleInputOnChange = (e) => {
26.     setTempValue(e.target.value);
27.   };
28.
29.   return (
30.     <div className="row" onDoubleClick={handleDivDubleClick}>
31.       {
32.         isEditing ?
33.         <div className="column seven wide">
34.           <div className="ui input fluid">
35.             <input
36.               onChange={handleInputOnChange}
37.               onKeyDown={handleInputKeyDown}
38.               autoFocus={true}
39.               value={tempValue}
40.             />
41.           </div>
42.         </div> :
43.         <>
44.
45.           <div className="column five wide">
46.             <h1>{Value}</h1>
47.           </div>
48.
49.           <div className="column two wide">
50.             <button className="ui button circular icon green"><i className="white check icon"></i></button>
51.           </div>
52.           <div className="column two wide">
53.             <button className="ui button circular icon red"><i className="white remove icon"></i></button>
54.           </div>
55.         </>
56.       }
57.     </div>
58.   );
59. };
60.
61. export default Todo;
62.
```

CHECKING TO-DO ITEMS

Para ello tenemos una nueva función

```
const [completed, setCompleted] = useState(false);
```

```
const handleInputChange = (e) => {  
  setTempValue(e.target.value);  
};
```

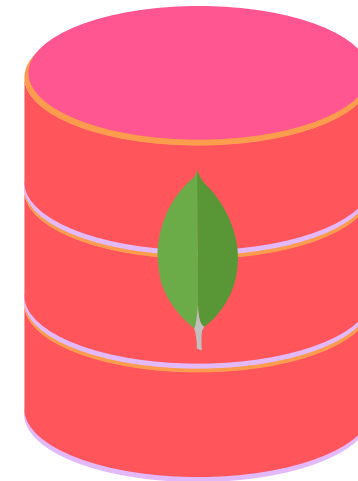
y le dimos un evento al botón verde

Inspecciona el código y verifica donde se realizaron los cambio

```
1.import React, { useState } from "react";  
2.  
3.const Todo = ({ title }) => {  
4.  
5.  const [isEditing, setIsEditing] = useState(false);  
6.  const [Value, setValue] = useState(title)  
7.  const [tempValue, setTempValue] = useState(title);  
8.  
9.  const handleDivDubleClick = () => {  
10.   setIsEditing(true);  
11.  };  
12.  
13.  const handleInputKeyDown = (e) => {  
14.   const key = e.keyCode;  
15.  
16.   if (key === 13) {  
17.    setValue(tempValue);  
18.    setIsEditing(false);  
19.   } else if (key === 27) {  
20.    setTempValue(Value);  
21.    setIsEditing(false);  
22.   }  
23.  };  
24.  
25.  const handleInputChange = (e) => {  
26.   setTempValue(e.target.value);  
27.  };  
28.  
29.  return (  
30.   <div className="row" onDoubleClick={handleDivDubleClick}>  
31.    {  
32.     isEditing ?  
33.     <div className="column seven wide">  
34.       <div className="ui input fluid">  
35.         <input  
36.           onChange={handleInputChange}  
37.           onKeyDown={handleInputKeyDown}  
38.           autoFocus={true}  
39.           value={tempValue}  
40.         />  
41.       </div>  
42.     </div> :  
43.     <>  
44.  
45.     <div className="column five wide">  
46.       <h1>{Value}</h1>  
47.     </div>  
48.  
49.     <div className="column two wide">  
50.       <button className="ui button circular icon green"><i className="white check icon"></i></button>  
51.     </div>  
52.     <div className="column two wide">  
53.       <button className="ui button circular icon red"><i className="white remove icon"></i></button>  
54.     </div>  
55.   </>  
56.   }  
57. </div>  
58. );  
59.};  
60.  
61.export default Todo;  
62.
```

¡OMG! SIMULACIÓN
DEL BACK-END

DATABASE



APP

LIST

TODO

OMG SIMULACIÓN DEL BACK-END

Has mucho zoom y
verifica estos tres
componentes e intuye
por porque se hace
esta simulación y que
debe pasar como
resultado final al dar
click en el check verde

```
JS Appjs x JS Listjs JS Todojs
app > src > JS Appjs > [0] list > [0] completed
1 import React, {useState} from "react";
2
3 import Form from "../components/Form";
4 import Section from "../components/Section";
5 import List from "../components/List";
6
7 const appTitle = "To-Do-app";
8
9 const list = [
10   { title : "test #1", completed: false },
11   { title : "test #2" },
12   { title : "test #3" }
13 ];
14 const App = () =>{
15   const [todoList, setTodoList] = useState(list);
16   return <div className="ui container center aligned">
17     <Section>
18       <h1>{appTitle}</h1>
19     </Section>
20
21     <Section>
22       <Form/>
23     </Section>
24
25     <Section>
26       <List list={todoList}/>
27     </Section>
28
29   </div>;
30 }
31
32
33 export default App;
34
35
```

```
JS Appjs JS Listjs JS Todojs
app > src > components > JS Listjs > ...
1 import React from "react";
2 import Todo from "../Todo";
3
4 const List = ({list}) => {
5   const renderedList = list.map((item) => <Todo title={item.title} completed={item.completed} key={item.title} />);
6   return(
7     <div className="ui grid center aligned">
8       {renderedList}
9     </div>
10   );
11 };
12
13 export default List; |
```

```
JS Appjs JS Listjs JS Todojs
app > src > components > JS Todojs > [0] todo > [0] handleInputKeyDown
1 import React, { useState } from "react";
2
3 const Todo = ({ title, completed }) => {
4
5   const [isEditing, setIsEditing] = useState(false);
6   const [value, setValue] = useState(title)
7   const [tempValue, setTempValue] = useState(title);
8   const [completedState, setCompleted] = useState(completed);
9
10
11   const handleDivDoubleClick = () => {
12     setIsEditing(true);
13   };
14
15   const handleInputKeyDown = (e) => {
16     const key = e.keyCode;
17
18     if (key === 13) {
19       setValue(tempValue);
20       setIsEditing(false);
21     } else if (key === 27) {
22       setTempValue(value);
23       setIsEditing(false);
24     }
25   };
26
27   const handleInputChange = (e) => {
28     setTempValue(e.target.value);
29   };
30
31   const handleButtonClick = () => {
32     setCompleted(true);
33   };
34
35   return (
36     <div className="row" onDoubleClick={handleDivDoubleClick}>
37       {
38         isEditing ?
39         <div className="column seven wide">
40           <div className="ui input fluid">
41             <input
42               onChange={handleInputChange}
43               onKeyDown={handleInputKeyDown}
44               autoFocus={true}
45               value={tempValue}
46             />
47           </div>
48         </div> :
49         <div>
50
51           <div className="column five wide">
52             <h1 className="ui header" + (completedState ? "green" : "")>{value}</h1>
53           </div>
54
55           <div className="column two wide">
56             <button
57               className="ui button circular icon green"
58               onClick={handleButtonClick}
59             >
60               <i className="white check icon"/></i></button>
61             </div>
62             <div className="column two wide">
63               <button className="ui button circular icon red"><i className="white remove icon"/></i></button>
64             </div>
65           </div>
66         </div>
67       )
68     </div>
69   );
70 };
71
```

To-Do-app

Enter something to



test #1



test #2



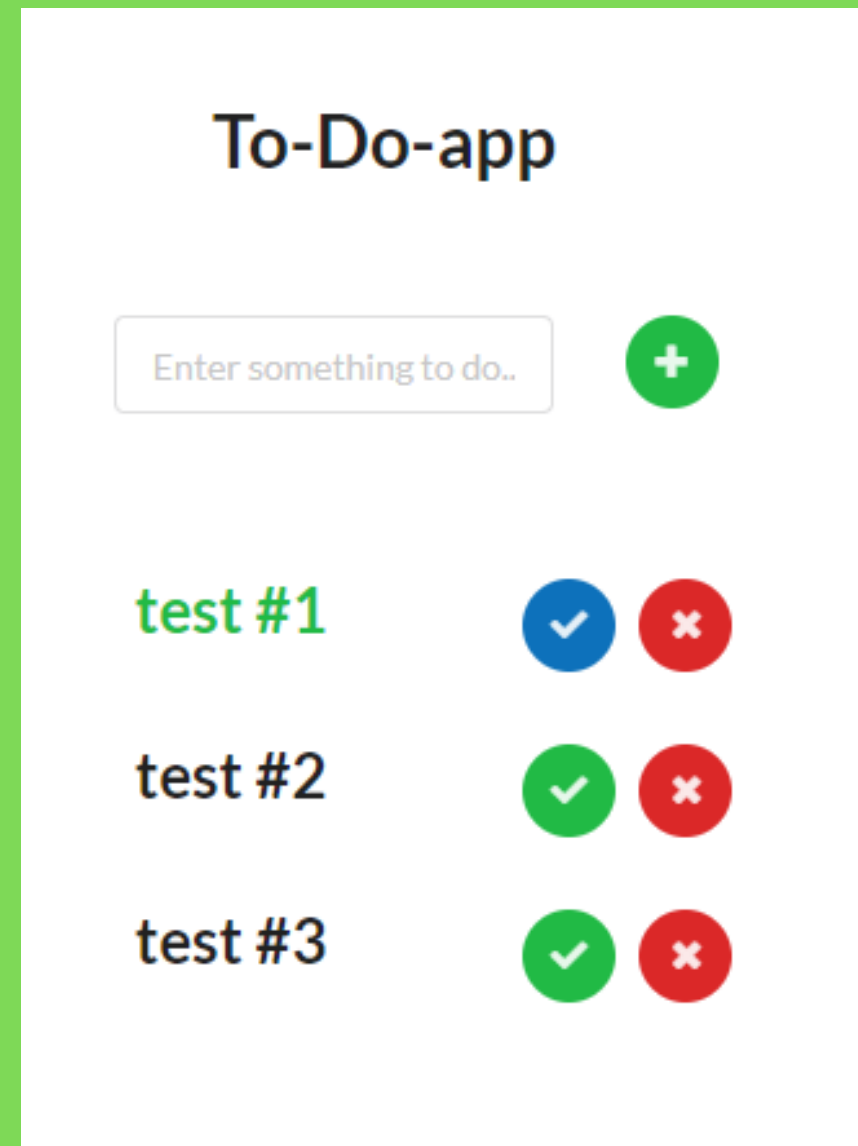
test #3



DETALLES E INTERACCIONES

Asi debe verse para ello revisa este código y has los cambios...

jejeje de nuevo dale formato pues en mayúscula todo no funcionara es una manera divertida de revisar (pd no odien a sus profes)



```
1. IMPORT REACT, { USESTATE } FROM "REACT";
2.
3. CONST TODO = ({ TITLE, COMPLETED }) => {
4.
5.   CONST [ISEDITING, SETISEDITING] = USESTATE(FALSE);
6.   CONST [VALUE, SETVALUE] = USESTATE(TITLE)
7.   CONST [TEMPVALUE, SETTEMPVALUE] = USESTATE(TITLE);
8.   CONST [COMPLETEDSTATE, SETCOMPLETED] = USESTATE(COMPLETED);
9.
10.
11.   CONST HANDLEDIVDOUBLECLICK = () => {
12.     SETISEDITING(TRUE);
13.   };
14.
15.   CONST HANDLEINPUTKEYDOWN = (E) => {
16.     CONST KEY = E.KEYCODE;
17.
18.     IF (KEY === 13) {
19.       SETVALUE(TEMPVALUE);
20.       SETISEDITING(FALSE);
21.     } ELSE IF (KEY === 27) {
22.       SETTEMPVALUE(VALUE);
23.       SETISEDITING(FALSE);
24.     }
25.   };
26.
27.   CONST HANDLEINPUTONCHANGE = (E) => {
28.     SETTEMPVALUE(E.TARGET.VALUE);
29.   };
30.
31.   CONST HANDLEBUTTONCLICK = () => {
32.     SETCOMPLETED((OLDCOMPLETED) => !OLDCOMPLETED);
33.   };
34.
35.   RETURN (
36.     <DIV CLASSNAME="ROW">
37.       {
38.         ISEDITING ?
39.         <DIV CLASSNAME="COLUMN SEVEN WIDE">
40.           <DIV CLASSNAME="UI INPUT FLUID">
41.             <INPUT
42.               ONCHANGE={HANDLEINPUTONCHANGE}
43.               ONKEYDOWN={HANDLEINPUTKEYDOWN}
44.               AUTOFOCUS={TRUE}
45.               VALUE={TEMPVALUE}
46.             />
47.           </DIV>
48.         </DIV> :
49.         <>
50.
51.         <DIV CLASSNAME="COLUMN FIVE WIDE" ONDOUBLECLICK={HANDLEDIVDOUBLECLICK}>
52.           <H2 CLASSNAME={"UI HEADER" + (COMPLETEDSTATE ? " GREEN" : "")}>{VALUE}</H2>
53.         </DIV>
54.
55.         <DIV CLASSNAME="COLUMN ONE WIDE">
56.           <BUTTON
57.             CLASSNAME={"UI BUTTON CIRCULAR ICON" + (COMPLETEDSTATE ? " BLUE" : " GREEN")}
58.             ONCLICK={HANDLEBUTTONCLICK}
59.           >
60.             <I CLASSNAME="WHITE CHECK ICON"></I></BUTTON>
61.           </DIV>
62.           <DIV CLASSNAME="COLUMN TWO WIDE">
63.             <BUTTON CLASSNAME="UI BUTTON CIRCULAR ICON RED"><I CLASSNAME="WHITE REMOVE ICON"></I></BUTTON>
64.           </DIV>
65.         </>
66.       }
67.     </DIV>
68.   );
69. };
70.
71. EXPORT DEFAULT TODO;
```


AGREGAR ITEMS TO-DO VACIAR LA ENTRADA Y VALIDACIÓN DEL FORMULARIO.

Serán los ítemes que almacene inicialmente nuestra base de datos y se renderizarán en la lista

1. crear una función en App.js que traerá los props de Form.js

2. En el componente Form.js verifica el código a continuación y recuerda para que sirve

```
const handleFormSubmit = (e) => {
```

```
  e.preventDefault();
```

3. revisa el metodo .trim para que servira?

```
1. import React, {useState} from "react";
2.
3. import Form from "../components/Form";
4. import Section from "../components/Section";
5. import List from "../components/List";
6.
7. const appTitle = "To-Do-app";
8.
9. const list = [
10.  { title : "test #1", completed: false },
11.  { title : "test #2" },
12.  { title : "test #3" }
13.];
14.
15.
16.
17. const App = () =>{
18.  const [todoList, setTodoList] = useState(list);
19.
20.  const addTodo = (item) => {
21.    setTodoList((oldlist) => [...oldlist, item]);
22.  };
23.
24.  return <div className="ui container center aligned">
25.    <Section>
26.      <h1>{appTitle}</h1>
27.    </Section>
28.
29.    <Section>
30.      <Form addTodo={addTodo}/>
31.    </Section>
32.
33.    <Section>
34.      <List list={todoList}/>
35.    </Section>
36.  </div>;
37.
38.
39. </div>;
40.}
41.
42. export default App;
```

```
1. import React, { useState } from "react";
2.
3. const Form = ({ addTodo }) => {
4.  const [inputValue, setInputValue] = useState("");
5.
6.  const handleInputChange = (e) => {
7.    setInputValue(e.target.value);
8.  };
9.
10. const handleFormSubmit = (e) => {
11.  e.preventDefault();
12.
13.  if(inputValue.trim() === "") return;
14.
15.  addTodo({ title: inputValue, completed: false });
16.  setInputValue("");
17. };
18.
19. return (
20.  <form className="ui form" onSubmit={handleFormSubmit}>
21.    <div className="ui grid center aligned">
22.      <div className="row">
23.        <div className="column five wide">
24.          <input
25.            value={inputValue}
26.            onChange={handleInputChange}
27.            type="text"
28.            placeholder="Enter something to do..."
29.          />
30.        </div>
31.
32.        <div className="column one wide">
33.          <button type="submit" className="ui button circular icon green"><i className="white plus icon"></i>
34.        </div>
35.      </div>
36.    </div>
37.  </form>
38. );
39.};
40.
41. export default Form;
```

VAMOS A REMOVER LOS ITEMS DEL TO-DO

PARTE 1

Locura de props ...
debes poner mucha atención en
App.js, List.js y Todo.js
analiza a :

removeTodoListProp
removeTodoItemProp

```
1.import React, {useState} from "react";
2.
3.import Form from "../components/Form";
4.import Section from "../components/Section";
5.import List from "../components/List";
6.
7.const appTitle = "To-Do-app";
8.
9.const list = [
10. { id: 1, title : "test #1", completed: false },
11. { id: 2, title : "test #2" },
12. { id: 3, title : "test #3" }
13.];
14.
15.const App = () =>{
16.  const [todoList, setTodoList] = useState(list);
17.
18.  const addTodo = (item) => {
19.    setTodoList((oldlist) => [...oldlist, item]);
20.  };
21.
22.  const removeTodo = (id) => {
23.
24.  };
25.
26.  return <div className="ui container center aligned">
27.    <Section>
28.      <h1>{appTitle}</h1>
29.    </Section>
30.
31.    <Section>
32.      <Form addTodo={addTodo}/>
33.    </Section>
34.
35.    <Section>
36.      <List removeTodoListProp={removeTodo} list={todoList}/>
37.    </Section>
38.
39.
40.  </div>;
41.}
42.
43.export default App;
44.
45.
46.
```

```
1.import React from "react";
2.import Todo from "../Todo";
3.
4.const List = ({list, removeTodoListProp }) => {
5.  const renderedList = list.map((item) => <Todo title={item.title} completed={item.completed}
    removeTodoItemProp={((e) => removeTodoListProp(item.id)) key={item.title} />);
6.  return(
7.    <div className="ui grid center aligned">
8.      {renderedList}
9.    </div>
10. );
11.};
12.
13.export default List;
14.
15.
16.
17.
```

```
1.import React, { useState } from "react";
2.
3.const Todo = ({ title, completed, removeTodoItemProp }) => {
4.
5.  const [isEditing, setIsEditing] = useState(false);
6.  const [Value, setValue] = useState(title);
7.  const [tempValue, setTempValue] = useState(title);
8.  const [completedState, setCompleted] = useState(completed);
9.
10.
11.  const handleDivDoubleClick = () => {
12.    setIsEditing(true);
13.  };
14.
15.  const handleInputKeyDown = (e) => {
16.    const key = e.keyCode;
17.
18.    if (key === 13) {
19.      setTempValue(tempValue);
20.      setIsEditing(false);
21.    } else if (key === 27) {
22.      setTempValue(tempValue);
23.      setIsEditing(false);
24.    }
25.  };
26.
27.  const handleInputOnChange = (e) => {
28.    setTempValue(e.target.value);
29.  };
30.
31.  const handleButtonClick = () => {
32.    setCompleted((oldCompleted) => !oldCompleted);
33.  };
34.
35.  return (
36.    <div className="row">
37.      {
38.        isEditing ?
39.        <div className="column seven wide">
40.          <div className="ui input fluid">
41.            <input
42.              onChange={handleInputOnChange}
43.              onKeyDown={handleInputKeyDown}
44.              autoFocus={true}
45.              value={tempValue}
46.            />
47.          </div>
48.        </div> :
49.        <>
50.
51.        <div className="column five wide" onDoubleClick={handleDivDoubleClick}>
52.          <h2 className="ui header" + (completedState ? " green" : "")>{Value}</h2>
53.        </div>
54.
55.        <div className="column one wide">
56.          <button
57.            className="ui button circular icon" + (completedState ? " blue" : " green")
58.            onClick={handleButtonClick}
59.          >
60.            <div className="white check icon"></div></button>
61.          </div>
62.        <div className="column two wide">
63.          <button
64.            oneClick={removeTodoItemProp}
65.            className="ui button circular icon red"
66.          >
67.            <div className="white remove icon"></div></button>
68.          </div>
69.        </>
70.      }
71.    </div>
72.  );
73.};
74.
75.export default Todo;
76.import React, { useState } from "react";
77.
78.const Todo = ({ title, completed, removeTodoItemProp }) => {
79.
80.  const [isEditing, setIsEditing] = useState(false);
81.  const [Value, setValue] = useState(title);
82.  const [tempValue, setTempValue] = useState(title);
83.  const [completedState, setCompleted] = useState(completed);
84.
85.
86.  const handleDivDoubleClick = () => {
87.    setIsEditing(true);
88.  };
89.
90.  const handleInputKeyDown = (e) => {
91.    const key = e.keyCode;
92.
93.    if (key === 13) {
94.      setValue(tempValue);
95.      setIsEditing(false);
96.    } else if (key === 27) {
97.      setTempValue(tempValue);
98.    }
99.    setIsEditing(false);
100.  };
101.
102.  const handleInputOnChange = (e) => {
103.    setTempValue(e.target.value);
104.  };
105.
106.  const handleButtonClick = () => {
107.    setCompleted((oldCompleted) => !oldCompleted);
108.  };
109.
110.  return (
111.    <div className="row">
112.      {
113.        isEditing ?
114.        <div className="column seven wide">
115.          <div className="ui input fluid">
116.            <input
117.              onChange={handleInputOnChange}
118.              onKeyDown={handleInputKeyDown}
119.              autoFocus={true}
120.              value={tempValue}
121.            />
122.          </div>
123.        </div> :
124.        <>
125.
126.        <div className="column five wide" onDoubleClick={handleDivDoubleClick}>
127.          <h2 className="ui header" + (completedState ? " green" : "")>{Value}</h2>
128.        </div>
129.
130.        <div className="column one wide">
131.          <button
132.            className="ui button circular icon" + (completedState ? " blue" : " green")
133.            onClick={handleButtonClick}
134.          >
135.            <div className="white check icon"></div></button>
136.          </div>
137.        <div className="column two wide">
138.          <button
139.            oneClick={removeTodoItemProp}
140.            className="ui button circular icon red"
141.          >
142.            <div className="white remove icon"></div></button>
143.          </div>
144.        </>
145.      }
146.    </div>
147.  );
148.
149.
150.export default Todo;
151.
```

FUNCIONALIDAD DE REMOVER ITEMS DEL TO-DO

List.filter es la pista!
reto te dejaremos fragmentos de
codigo para que analices en que
componentes deben ir recuerda
los 3 componentes que
trabajamos en el anterior slide

```
CONST REMOVETODO = (ID) => {  
  
  SETTODOLIST((OLDLIST)=>OLDLIST.FI  
LTER((ITEM) => ITEM.ID !== ID));
```

```
<SECTION>  
  <LIST REMOVETODOLISTPROP=  
{REMOVETODO} LIST={TODOLIST}/>  
</SECTION>
```

Parte

4

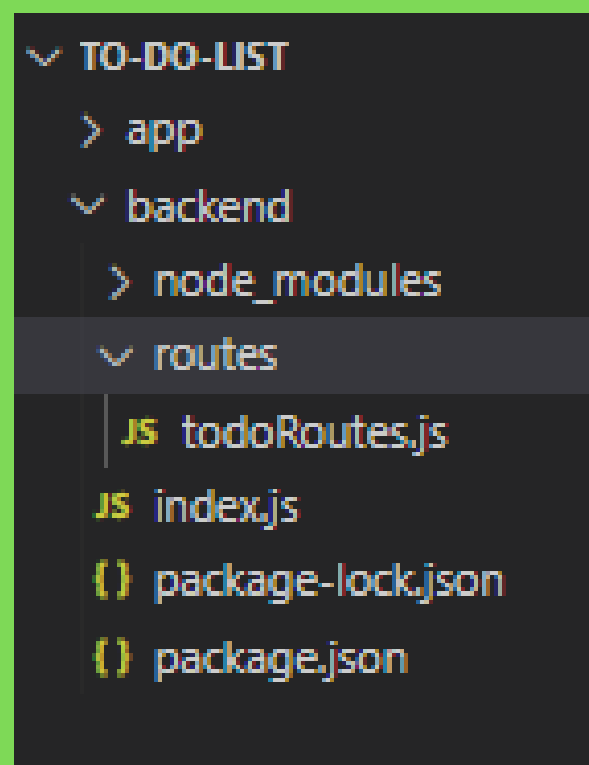
Creación del Back-End

CONSTRUCCIÓN DE LA ESTRUCTURA BACK-END



En la carpeta backend debemos crear

1. carpeta llamada routes
2. dentro de routes debemos crear el archivo todoRoutes.js
3. y al mismo nivel de la carpeta routes crea un archivo llamado index.js
4. y agregaremos el código a continuación



```
JS index.js X JS todoRoutes.js
backend > JS index.js > ...
1  const express = require("express");
2
3
4  const PORT = 3030;
5  const app = express();
6
7  const todoRoutes = require("../routes/todoRoutes");
8
9  app.use("/todos", todoRoutes);
10
11 app.listen(PORT, () => {
12   |   console.log("The server is listening on port " + PORT);
13   });
```

```
JS index.js JS todoRoutes.js X
backend > routes > JS todoRoutes.js > ...
1  const router = require("express").Router();
2
3  router.get("/", (req, res) => {
4   |   console.log("you're in the index page")
5   });
6
7  module.exports = router;
```

VERIFICAREMOS SI EL SERVIDOR DE NODE ESTA ESCUCHANDO



En la consola verificando estar en nuestra ruta
back-end vas a escribir

node index (debe salir el mensaje de consola
que tenemos en index.js

```
User@DESKTOP-08TC4RL MINGW64 ~/Desktop/to-do-list/backend  
$ node index  
The server is listening on port 3030
```

CONEXIÓN A NUESTRA BASE DE DATOS Y CREACIÓN DEL MODELO

1. Crearemos la carpeta models al mismo nivel que routes, crearemos el archivo todo.js dentro de models. Luego usaremos mongoose para la conexión y crearemos nuestro esquema (si no entiendes este término te invitamos a investigarlo y a profundizarlo un poco)
2. En index.js debe quedar como la segunda imagen pregúntate por qué se está generando esta conexión (tiene que ver con los primeros pasos)

```
const mongoose = require("mongoose");

const TodoSchema = new mongoose.Schema({
  title: String,
  completed: Boolean
});

module.exports = mongoose.model("Todo", TodoSchema);
```

```
backend > JS index.js > [Ⓢ] app
1  const express = require("express");
2  const mongoose = require("mongoose");
3
4
5
6  const PORT = 3030;
7  const app = express();
8
9  const todoRoutes = require("../routes/todoRoutes");
10
11  mongoose.connect("mongodb://localhost/todolist")
12    .then(() => console.log("Connected successfully"))
13    .catch((err) => console.error(err));
14
15  app.use("/todos", todoRoutes);
16
17  app.listen(PORT, () => {
18    console.log("The server is listening on port " + PORT);
19  });
```

FETCHING DATA DE NUESTRA BASE DE DATOS

1. Bien ahora modificaremos nuestro archivo `todorouters` como se muestra a continuación
2. luego iremos a nuestro navegador pondremos `localhost:3030/todo/` (no tiene por qué renderizar algo)
3. y en consola testaremos detenemos el servidor lo volvemos arrancar y nos debe salir así:

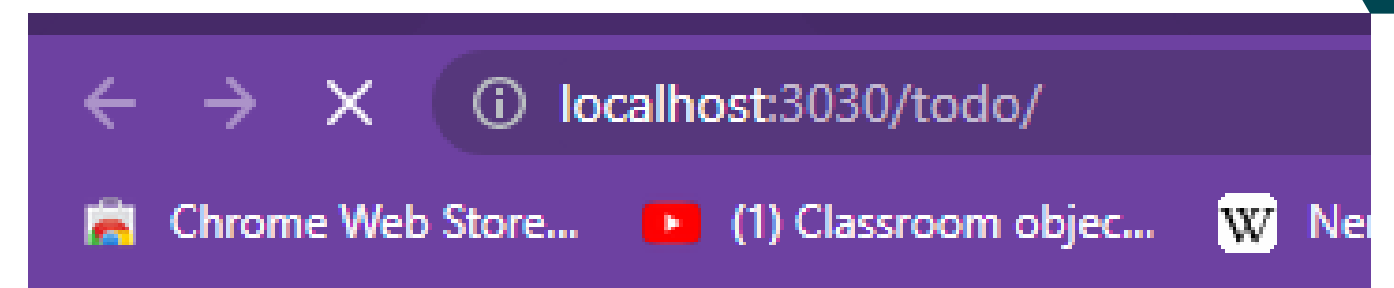
The server is listening on port 3030

Connected successfully

[]

aL tener este array vacío quiere decir que nuestra base está casi lista para recibir entradas

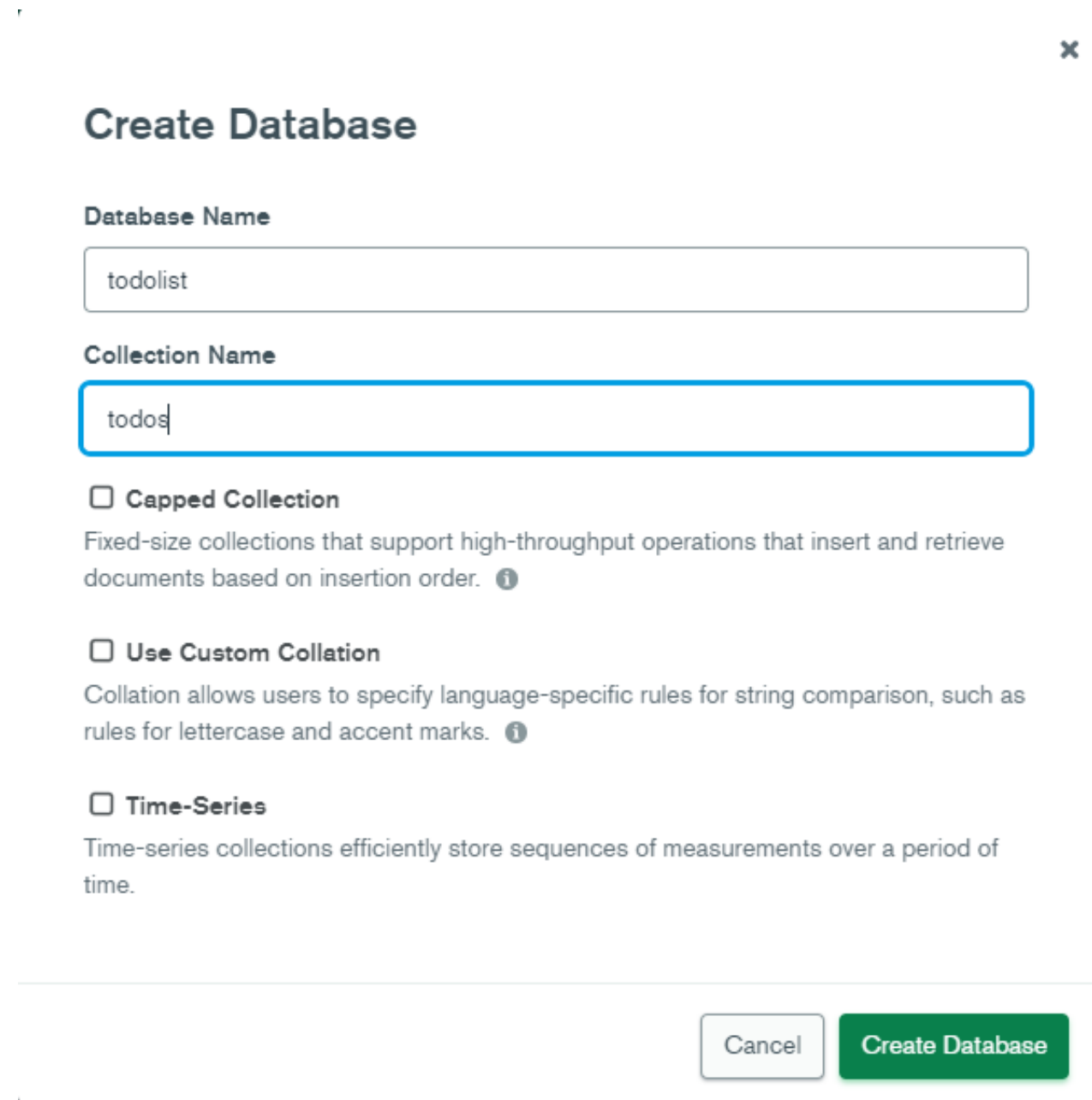
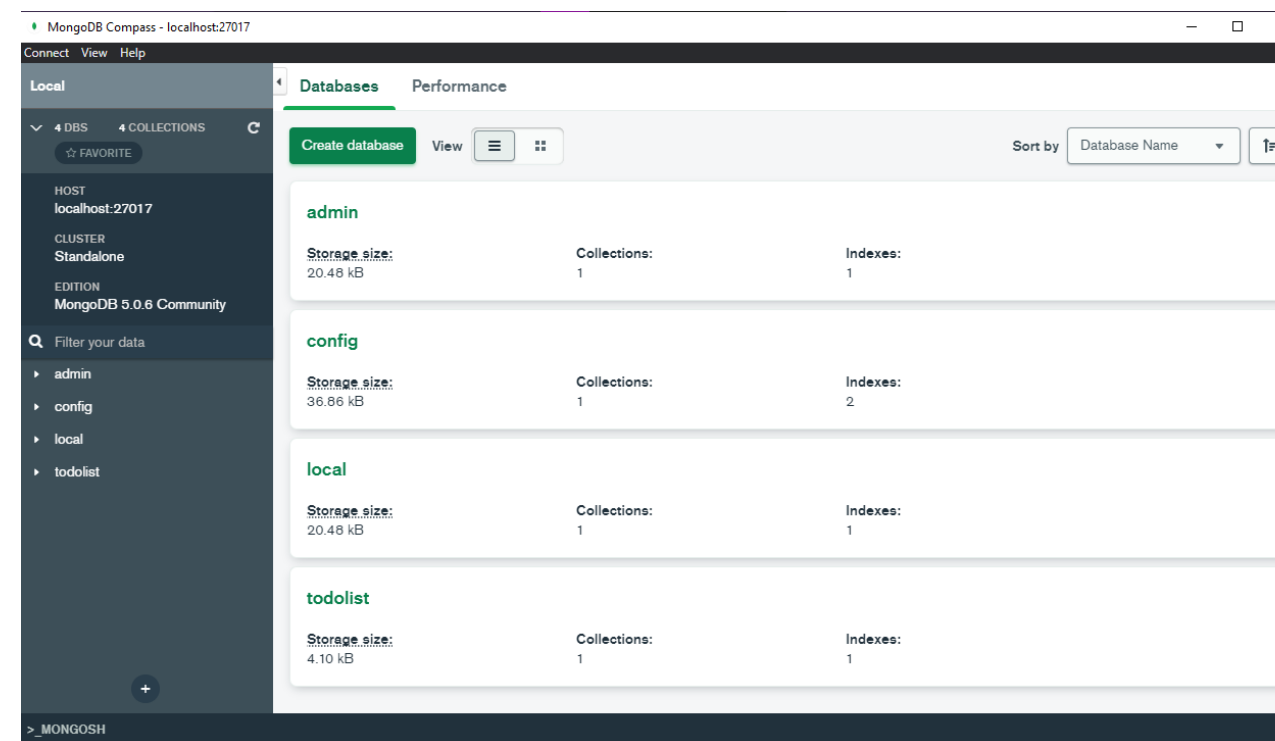
```
1. const router = require("express").Router();
2. const Todo = require("../models/Todo");
3.
4.
5. router.get("/", (req, res) => {
6.   Todo.find((err, result) => {
7.     if(err) throw new Error(err);
8.     console.log(result);
9.   });
10.});
11.
12.
13. module.exports = router;
14.
```



```
PS C:\Users\User\Desktop\to-do-list\backend> node index
The server is listening on port 3030
Connected successfully
[]
[]
```


FETCHING DATA DE NUESTRA BASE DE DATOS

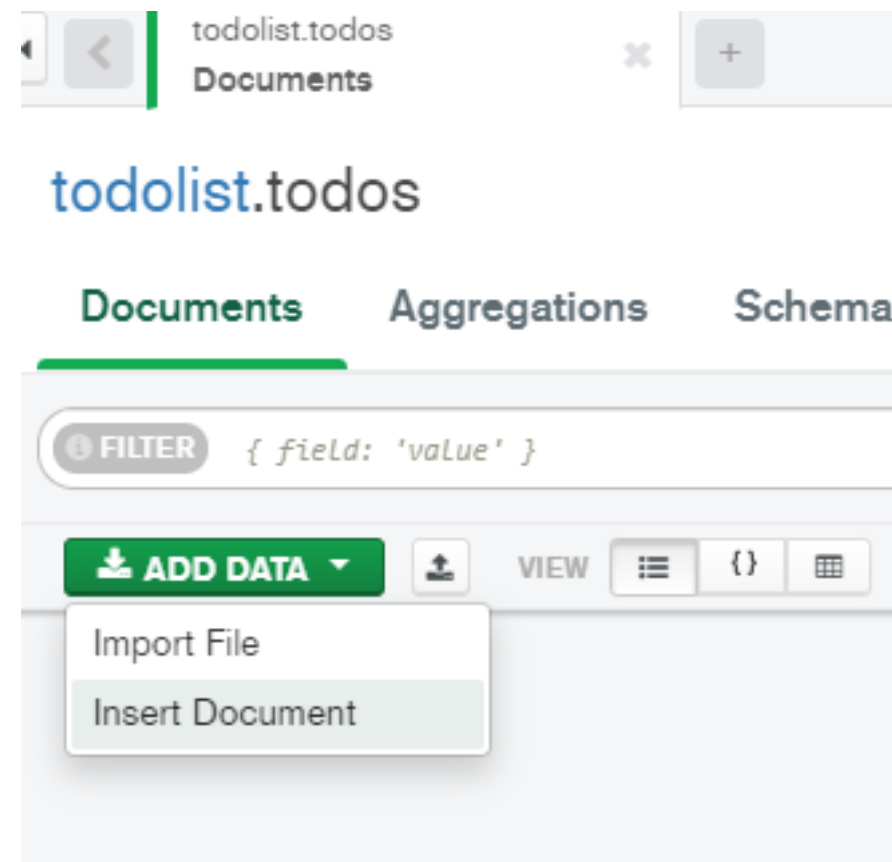
1. En nuestro archivo `todoRoutes.js` llamamos ruta `models`
`const Todo = require("../models/Todo");`
2. Abriremos nuestro MongoDB Compass y verificaremos en nuestra última conexión si en lista está nuestra base `todolist` ... Si no no pasa nada debes crearla
3. para crearla le das en `create database` y en el form le damos el nombre de `todolist` y en la colección le ponemos `todos`, ya que es el nombre que tiene nuestro modelo



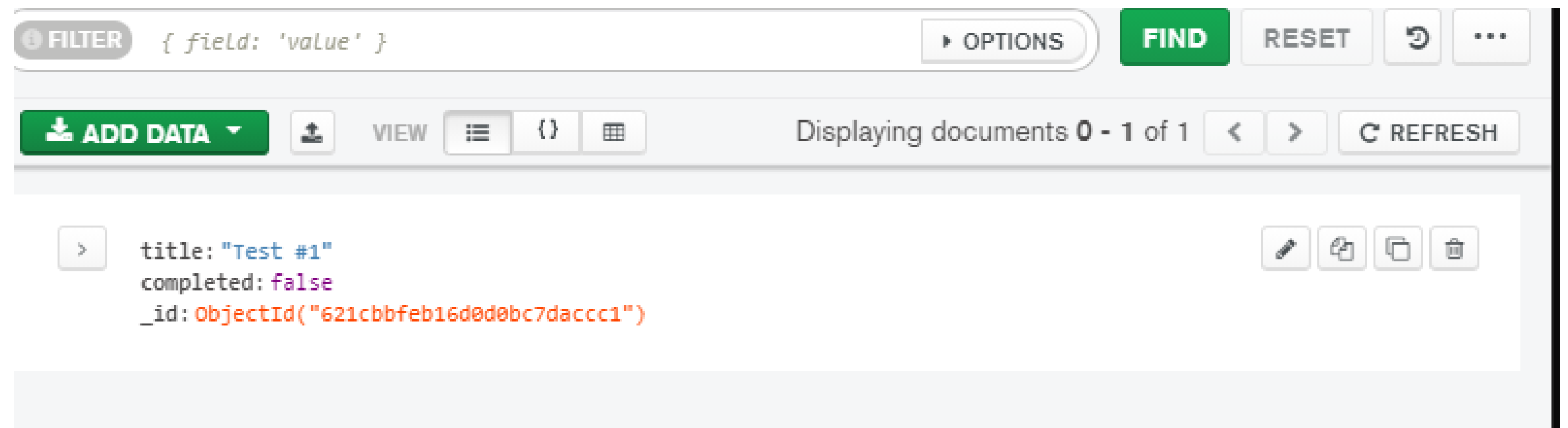
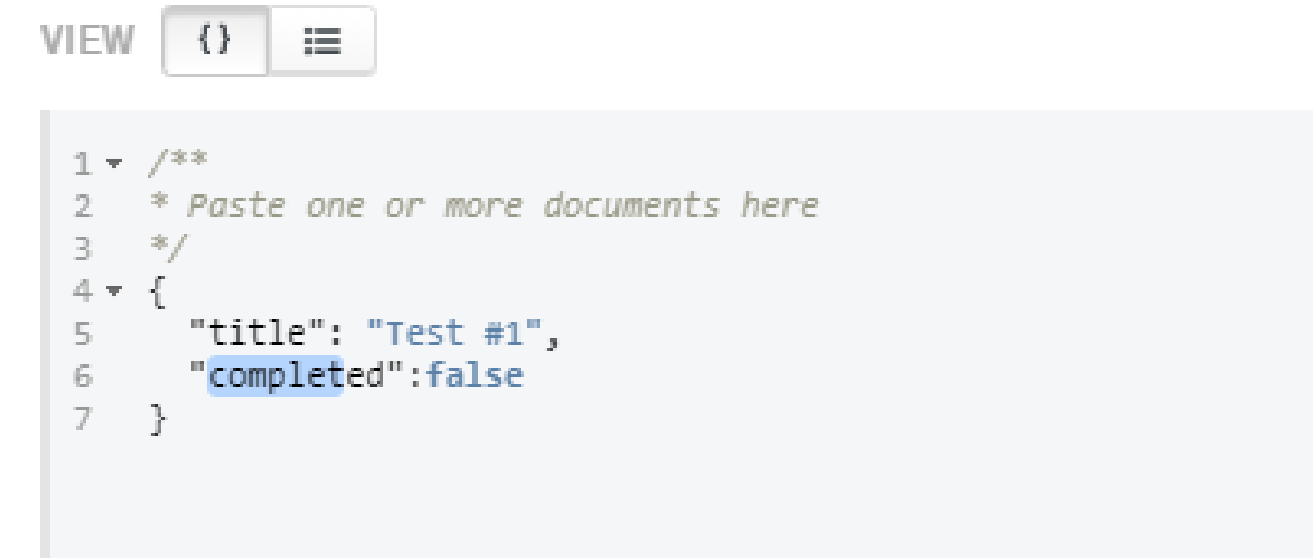
FETCHING DATA DE NUESTRA BASE DE DATOS

4. bien ahora crearemos un nuevo record en ADD DATA / Inster Document
5. Creamos un JSON para insertarlo así:

```
{  "title": "Test #1",  "completed": false}
```
6. verifica en consola que el servidor te traiga el ID



Insert to Collection todolist.todos



FETCHING DATA DE NUESTRA BASE DE DATOS

7. Ahora recargamos nuestro navegador y nuestra terminal debe mostrar el objeto que creamos en MongoDB así:

```
PS C:\Users\User\Desktop\to-do-list\backend> node index
The server is listening on port 3030
Connected successfully
[]
[]
[
  {
    _id: new ObjectId("621cef89a2e15eea65d85cf5"),
    title: 'Test #1',
    completed: false
  }
]
```

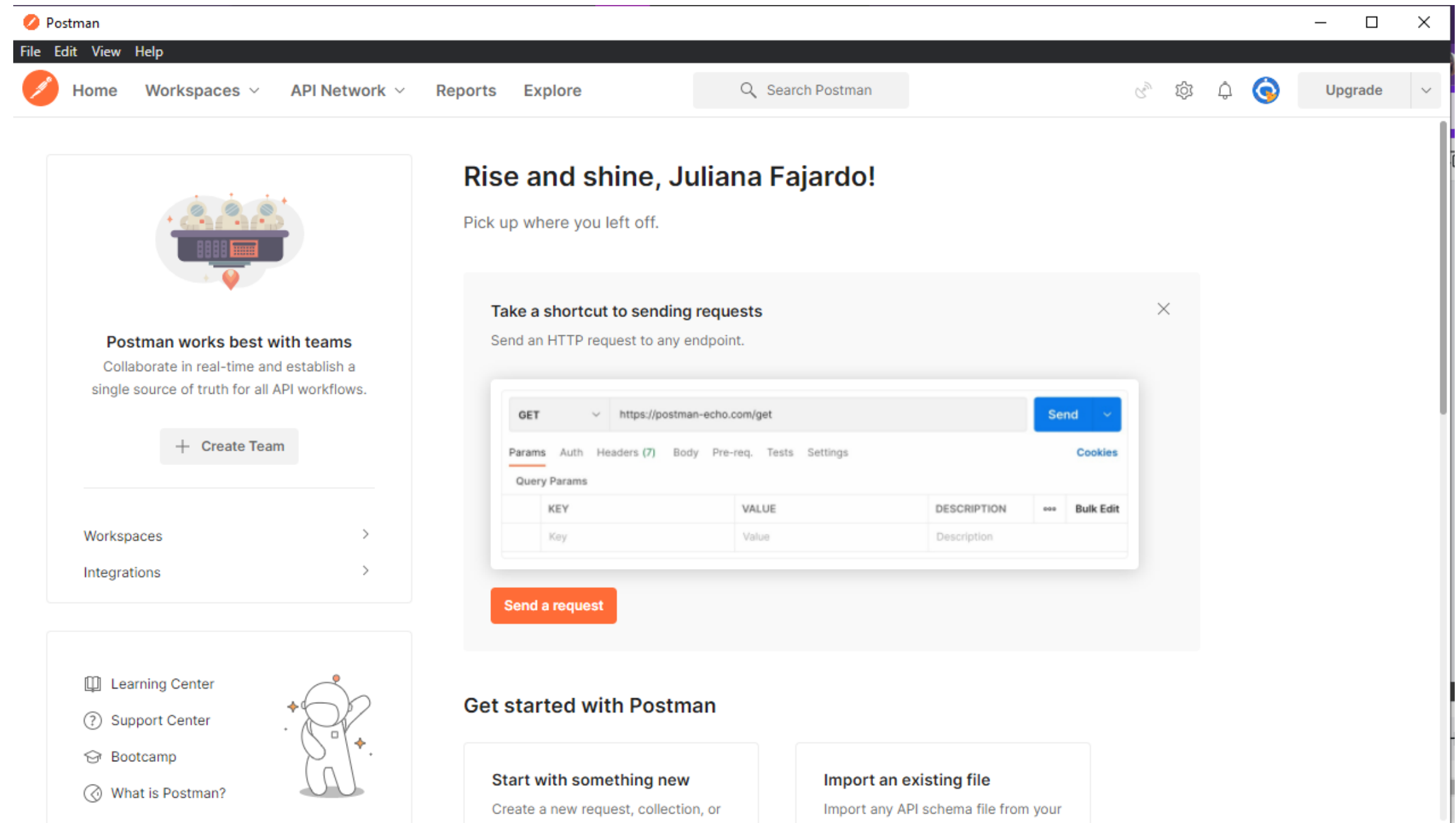
MAS RUTAS

Para ello necesitamos modificar el archivo Todorouts.js

```
1. CONST ROUTER =  
  REQUIRE("EXPRESS").ROUTER;  
2. CONST TODO = REQUIRE("../MODEL/TODO");  
3.  
4.  
5. ROUTER.GET("/", (REQ, RES) => {  
6.   TODO.FIND((ERR, RESULT) =>  
7.     IF(ERR) THROW NEW ERROR("Error al encontrar TODO");  
8.     CONSOLE.LOG(RESULT);  
9.   });  
10});  
11.  
12ROUTER.POST("/NEW" , (REQ, RES) => {  
13.  CONSOLE.LOG(REQ.BODY);  
14.  });  
15.  
16MODULE.EXPORTS = ROUTER;
```

POSTMAN

Crea una cuenta en postman
y descarga postman agent
vincula

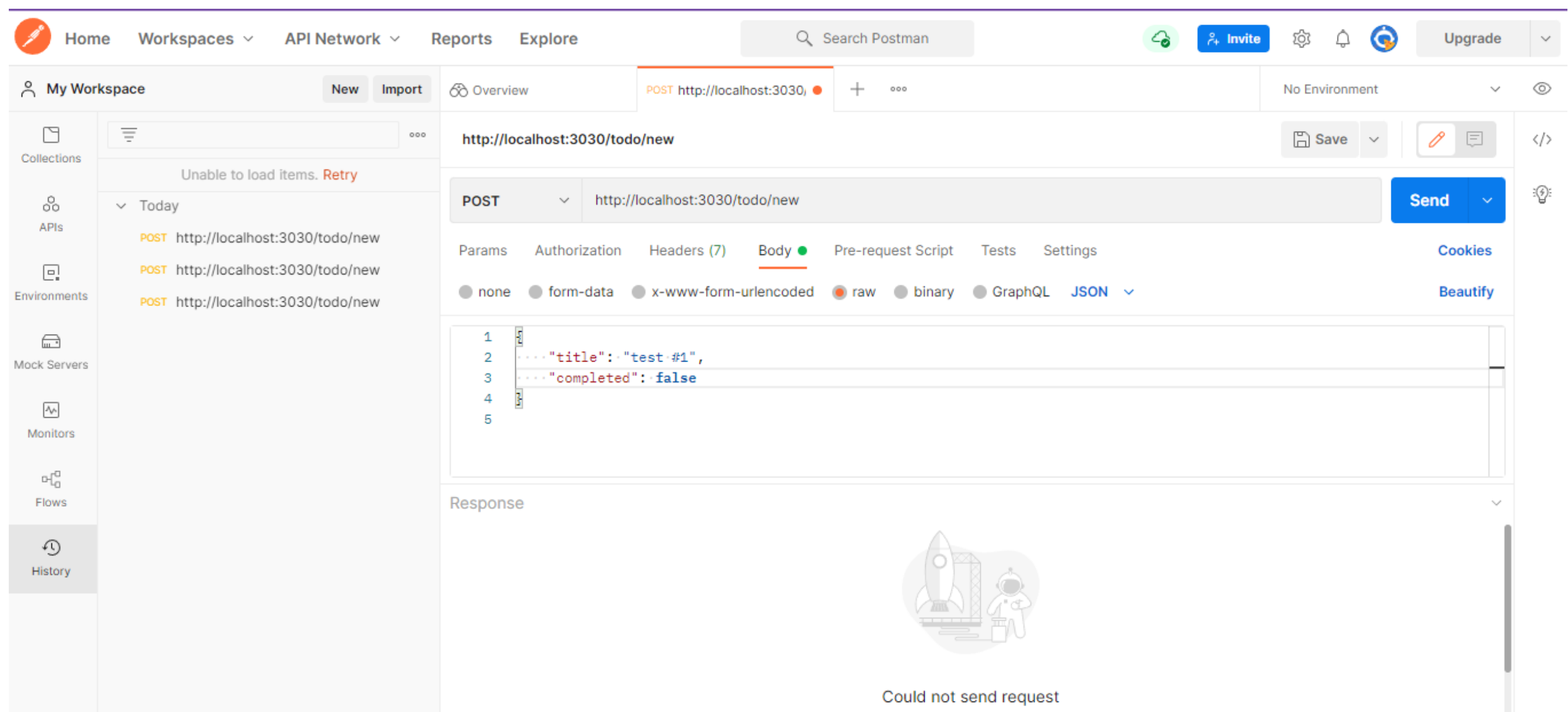
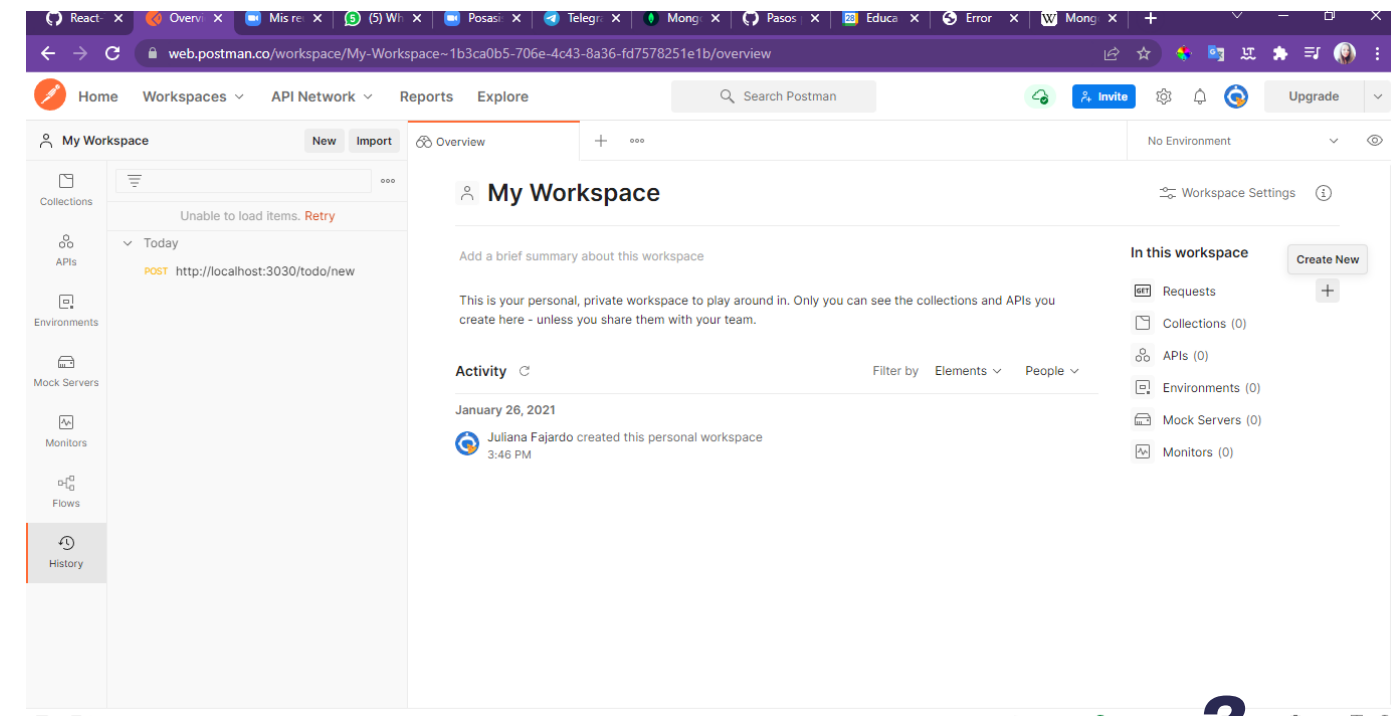
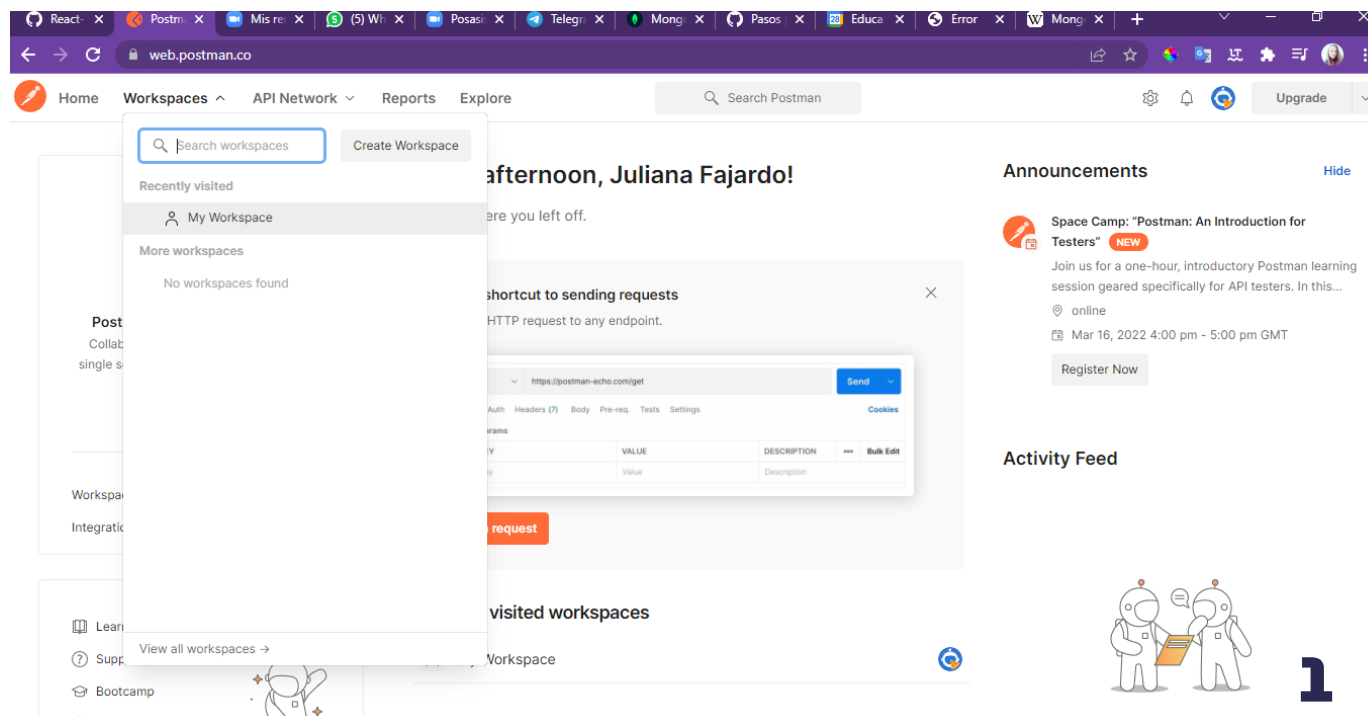


POSTMAN

Antes de iniciar postman en tu equipo debes hacer los siguientes pasos en el navegador una vez estes logiado ...

En la terminal te debe salir undefined

observa bien cada una de las imagenes y deja tu postman identico



POSTMAN

Modifica el archivo index.js
como a continuación

luego de agregar
`app.use(express.json());`
preguntate por que este dato es
importante
vuelve a postman y envia de
nuevo el post

en la terminal debe salir el
objeto asi:

```
{ title: 'test 1', completed: false }
```

```
1.const express = require("express");
2.const mongoose = require("mongoose");
3.
4.
5.const PORT = 3030;
6.const app = express();
7.
8.const todoRoutes = require("./routes/todoRoutes");
9.const connectionOptions = { useUnifiedTopology: true,
  useUrlParser: true, useFindAndModify: false };
10.
11.app.use(express.json());
12.
13.mongoose.connect("mongodb://localhost/todolist",
  connectionOptions)
14. .then(() => console.log("Connected successfully"))
15. .catch((err) => console.error(err));
16.
17.app.use("/todo", todoRoutes);
18.
19.app.listen(PORT, () => {
20.  console.log("The server is listening on port " + PORT);
21.});
```

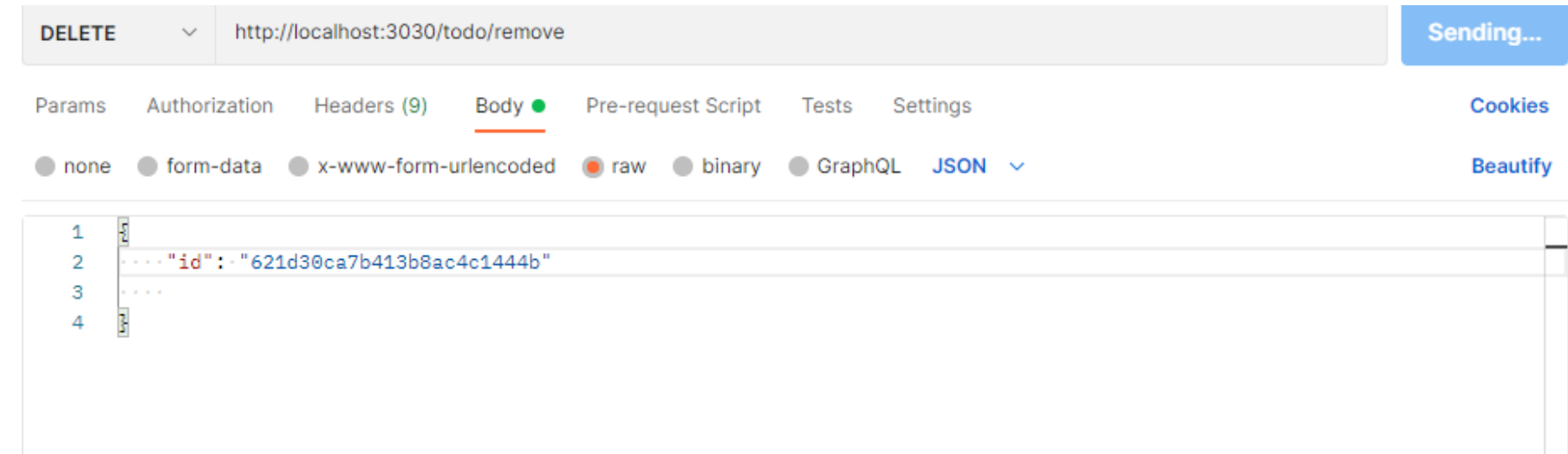
INSERTAR Y BORRAR DATOS

Modifica el archivo
todoRoutes.js como a
continuación

Modificar el Postman para
hacer el test del Delete

En la terminal debe quedar
como la tercera imagen

```
1. const router = require("express").Router();
2. const Todo = require("../models/Todo");
3.
4.
5. router.get("/", (req, res) => {
6.   Todo.find((err, result) => {
7.     if(err) throw new Error(err);
8.     console.log(result);
9.   });
10.});
11.
12. router.post("/new", (req, res) => {
13.   Todo.create(req.body, (err, result) => {
14.     if(err) throw new Error(err);
15.     console.log(result);
16.
17.   });
18.});
19.
20. router.delete("/:id", (req, res) => {
21.   Todo.findOneAndRemove({ _id: req.body.id }, (err, result) => {
22.     if(err) throw new Error(err);
23.     console.log(result);
24.
25.   });
26.});
27. module.exports = router;
28.
```



```
{
  title: 'test #1',
  completed: false,
  _id: new ObjectId("621d30ca7b413b8ac4c1444b"),
  __v: 0
}
{
  _id: new ObjectId("621d30ca7b413b8ac4c1444b"),
  title: 'test #1',
  completed: false,
  __v: 0
}
```


SIGUIENDO RESTFUL CONVENCIONES Y CREACIÓN DEL MÉTODO PUT

Modifica el archivo
todoRoutes.js como a
continuación

Observa el código y
coméntalo para saber que
pasa

busca como ir haciendo las
pruebas en postman que se
reflejen en mongodb y de
resultados en consola
mediante el servidor Node







Finalmente revisa y valida
las respuesputas de cada
metodo y verifica en
postman

```
1.const router = require("express").Router();
2.const Todo = require("../models/Todo");
3.
4.router.get("/", (req, res) => {
5.  Todo.find((err, result) => {
6.    if(err) throw new Error(err);
7.    res.json(result);
8.  });
9.});
10.
11.router.post("/", (req, res) => {
12.  Todo.create(req.body, (err, result) => {
13.    if(err) throw new Error(err);
14.    res.json(result);
15.  });
16.});
17.
18.router.put("/:id", (req, res) => {
19.  Todo.findOneAndUpdate({ _id: req.params.id }, req.body, { new: true }, (err, result) => {
20.    if(err) throw new Error(err);
21.    res.json(result);
22.  });
23.});
24.
25.router.delete("/:id", (req, res) => {
26.  Todo.findOneAndRemove({ _id: req.params.id }, (err, result) => {
27.    if(err) throw new Error(err);
28.    res.end();
29.  });
30.});
31.
32.module.exports = router;
33.
```

Parte 5

Conectar el front-backend

RECURSOS

TITLE	LAST MODIFIED
 .gitignore	2/22/21
 app	2/28/22
 backend	2/28/22
 Bienvenidos a MONGODB.pdf	3/1/22
 LICENSE	2/22/21
 README.md	2/22/21

RETO CONEXIÓN BACK FRONT Y DESPLIEGUE

Aquí se presenta el reto

debes crear la conexión tu sol@ para luego realizar el despliegue tanto del backend como del front

Te dejamos el link del código completo en drive pues deberás crear tu propio repositorio.

Si observas bien el código algunas pistas podrás encontrar de como realizarlo como: API, axios, useEffect, async await y sobre todo no dejes de testear en Postman, mongoBD (revisa conexiones y colecciones) y testear en consola que estás llamando y almacenando.

Recuerda personalizar los estilos de este proyecto para que sea único.

¡ÁNIMO!