ProjectManager | MERN

AGREGAR TAREAS

Backend

1. Agregar la propiedad `tasks` en el modelo el `Project`

```
    tasks : [
        {
            type:mongoose.Schema.Types.ObjectId,
            ref: 'Task',
        }
    ]
```

2. Agregar el método `populate()` en la consulta que trae el proyecto por ID en el método `detail()` de `projectsController` para traer todas las tareas vinculadas a ese proyecto.

```
const project = await Project.findById(id).populate('tasks');
```

3. Requerir los modelos `Task` y `Project` en `tasksController`

```
const Task = require("../models/Task")
```

4. Editar el método `store` de `tasksController` para que quede de la siguiente forma:

```
try {

      const {name, description, priority, project : projectId} = req.body;

      if (
          [name, description, priority].includes("") ||
          !name ||
          !description ||
          !priority
        )
          throw createError(400, "Todos los campos son obligatorios");

      const project = await Project.findById(projectId);

      if (req.user._id.toString() !== project.createdBy.toString())    throw
createError(403, "No estás autorizado");
```

```
        const taskStore = await Task.create(req.body);

        project.tasks.push(taskStore._id);
        await project.save();

        return res.status(201).json({
            ok : true,
            msg :'Tarea guardada con éxito',
            task : taskStore
        })
    } catch (error) {
        console.log(error);
        return errorResponse(res, error, "STORE-TASK");
    }
```

5. Agregar el middleware `checkToken` a la ruta de Tareas

```
  .use('/api/tasks',checkToken,require('./routes/tasks'))
```

Frontend

1. Crear un nuevo *state* en el componente `<ProjectsProvider/>` para manejar las ventanas emergentes y poner a disposición en el *value* de `<ProjectsProvider/>`

```
const [showModal, setShowModal] = useState(false);
```

2. Crear la función `handleShowModal()` para manejar todo lo relacionado con la apertura/cierre del modal y ponerla disponibilidad en el *value* de `<ProjectsProvider/>`

```
const handleShowModal = () => {
    setShowModal(!showModal)
}
```

3. Instalar la dependencia *hadlessUI* para utilizar componentes compatibles con *Tailwind*, en nuestro caso el componente `Dialog`

```
npm install @headlessui/react
```

- Más info:

- Sitio Web
- Documentación

4. Crear el componente `<ModalFormTask/>` con el siguiente contenido inicial:

```
import { Dialog, Transition } from '@headlessui/react';
import { Fragment} from 'react';
import { useProjects } from '../hooks/useProjects';

export const ModalFormTask = () => {

    const {showModal, handleShowModal} = useProjects();

    const handleClosed = () => {
        handleShowModal()
    }
    const handleSubmit = (e) => {
        e.preventDefault()
    }

    return (
        <Transition.Root show={ showModal } as={Fragment}>
            <Dialog as="div" className="fixed z-10 inset-0 overflow-y-auto"
onClose={handleClosed} static>
                <div className="flex items-end justify-center min-h-screen pt-4
px-4 pb-20 text-center sm:block sm:p-0">
                    <Transition.Child
                        as={Fragment}
                        enter="ease-out duration-300"
                        enterFrom="opacity-0"
                        enterTo="opacity-100"
                        leave="ease-in duration-200"
                        leaveFrom="opacity-100"
                        leaveTo="opacity-0"
                    >
                        <Dialog.Overlay
                            className="fixed inset-0 bg-gray-500 bg-opacity-75
transition-opacity"
                        />
                    </Transition.Child>

                    <span className="hidden sm:inline-block sm:align-middle sm:h-
screen" aria-hidden="true">
                        &#8203;
                    </span>

                    <Transition.Child
                        as={Fragment}
                        enter="ease-out duration-300"
                        enterFrom="opacity-0 translate-y-4 sm:translate-y-0
sm:scale-95"
                        enterTo="opacity-100 translate-y-0 sm:scale-100"
```

```
                        leave="ease-in duration-200"
                        leaveFrom="opacity-100 translate-y-0 sm:scale-100"
                        leaveTo="opacity-0 translate-y-4 sm:translate-y-0
sm:scale-95"
                    >
                            <div className="inline-block align-bottom bg-white
rounded-lg px-4 pt-5 pb-4 text-left overflow-hidden shadow-xl transform
transition-all sm:my-8 sm:align-middle sm:max-w-lg sm:w-full sm:p-6">


                            <div className="hidden sm:block absolute top-0 right-0
pt-4 pr-4">
                                <button
                                    type="button"
                                    className="bg-white rounded-md text-gray-400
hover:text-gray-500 focus:outline-none focus:ring-2 focus:ring-offset-2
focus:ring-indigo-500"
                                    onClick={ handleClosed }
                                >
                                <span className="sr-only">Cerrar</span>
                                    <svg xmlns="http://www.w3.org/2000/svg"
className="h-6 w-6" viewBox="0 0 20 20" fill="currentColor">
                                        <path fillRule="evenodd" d="M10 18a8 8 0
100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0
101.414 1.414L10 11.414l1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1
0 00-1.414-1.414L10 8.586 8.707 7.293z" clipRule="evenodd" />
                                    </svg>
                                </button>
                            </div>


                            <div className="sm:flex sm:items-start">
                                <div className="mt-3 text-center sm:mt-0 sm:ml-4
sm:text-left w-full">
                                    <Dialog.Title as="h3" className="text-lg
leading-6 font-bold text-gray-900">
                                       Nueva Tarea
                                    </Dialog.Title>

                                    <form
                                         className='my-10'
                                         onSubmit={handleSubmit}
                                         >
                                        <div className="mb-5">
                                            <label htmlFor="name"
className='uppercase text-gray-500 font-bold text-sm'>Nombre</label>
                                            <input
                                            type="text"
                                            placeholder='Nombre de la tarea'
                                            className='border w-full p-2 mt-2
placeholder-grey-400 rounded-md'
                                            name="name"
                                            />
                                        </div>
```

```
                                        <div className="mb-5">
                                            <label htmlFor="description"
  className='uppercase text-gray-500 font-bold text-sm'>Descripción</label>
                                            <textarea
                                            type="text"
                                            placeholder='Descripción de la tarea'
                                            className='border w-full p-2 mt-2
  placeholder-grey-400 rounded-md'
                                            style={{resize:"none"}}
                                            name="description"
                                            />
                                        </div>
                                        <div className="mb-5">
                                            <label htmlFor="dateExpire"
  className='uppercase text-gray-500 font-bold text-sm'>Fecha de Entrega</label>
                                            <input
                                            type="date"
                                            className='border w-full p-2 mt-2
  placeholder-grey-400 rounded-md'
                                            name='dateExpire'
                                            />
                                        </div>
                                        <div className="mb-5">
                                            <label htmlFor="priority"
  className='uppercase text-gray-500 font-bold text-sm'>Prioridad</label>
                                            <select
                                            className='border w-full p-2 mt-2
  placeholder-grey-400 rounded-md'
                                            >
                                                <option value="" hidden
  defaultValue={true} key="">Seleccione...</option>
                                                {
                                                    ['Baja', 'Media',
  'Alta'].map(prioridad => (
                                                        <option value={prioridad}
  key={prioridad}>{prioridad}</option>
                                                    ))
                                                }
                                            </select>
                                        </div>
                                        <button type="submit"
                                        className=' bg-sky-600 hover:bg-sky-700 w-
  full p-3 text-white transition-colors cursor-pointer rounded'
                                        >
                                            Guardar Tarea
                                        </button>
                                    </form>

                                </div>
                            </div>
                        </div>
                    </Transition.Child>
                </div>
            </Dialog>
```

```
        </Transition.Root>
    )
}
```

5. Utilizar el hook `useForm` en el componente `<ModalFormTask/>`

```
const {formValues, handleInputChange, reset} = useForm({
    name : "",
    description : "",
    dateExpire : "",
    priority : ""
});
const {name, description, dateExpire, priority} = formValues;
```

6. Agregar los atributos en los campos del formulario. Ejemplo:

```
value={name}
onChange={handleInputChange}
```

7. Crear un nuevo *state* en el componente `<ProjectsProvider/>` para manejar los errores de los modales y su correspondiente función. Tanto el estado como la función deber quedar disponibles en el *value*

```
  const [alertModal, setAlertModal] = useState({});
  const showAlertModal = (msg, time = true) => {
    setAlertModal({
      msg,
    });

    if (time) {
      setTimeout(() => {
        setAlertModal({});
      }, 3000);
    }
  };
```

8. Aplicar las validaciones correspondientes

```
if([name, description, priority, dateExpire].includes("")){
    showAlert("Todos los campos son obligatorios");

    return null
}
```

9. Crear la función `storetask()` en el componente `<ProjectsProvider/>` y hacerla disponible en el *value*

```
  const storeTask = async (task) => {
    try {
      const token = sessionStorage.getItem("token");
      if (!token) return null;

      const config = {
        headers: {
          "Content-Type": "application/json",
          Authorization: token,
        },
      };
      task.project = project._id;
      const { data } = await clientAxios.post("/tasks", task, config);
      project.tasks = [...project.tasks, data.task];
      setProject(project);

      setShowModal(false)

      Toast.fire({
        icon: "success",
        title: data.msg,
      });
      setAlert({});

    } catch (error) {
        console.log(error);
        showAlertModal(error.response ? error.response.data.msg : "Upss, hubo un
error", false);
      }

  };
```

10. Utilizando la función `storeTask()` en la función `handleSubmit()` del componente `<ModalFormTask/>` enviar los datos del formulario, previa validación

```
    const handleSubmit = (e) => {
        e.preventDefault()

        if([name, description, priority, dateExpire].includes("")){
            showAlertModal("Todos los campos son obligatorios");

            return null
        }
```

```
        storeTask({
            name,
            description,
            dateExpire,
            priority
        })

        reset()
    }
```

11. Completar la función `handleClosed()` con el siguiente código:

```
    const handleClosed = () => {
        handleShowModal()
        showAlertModal("")
    }
```