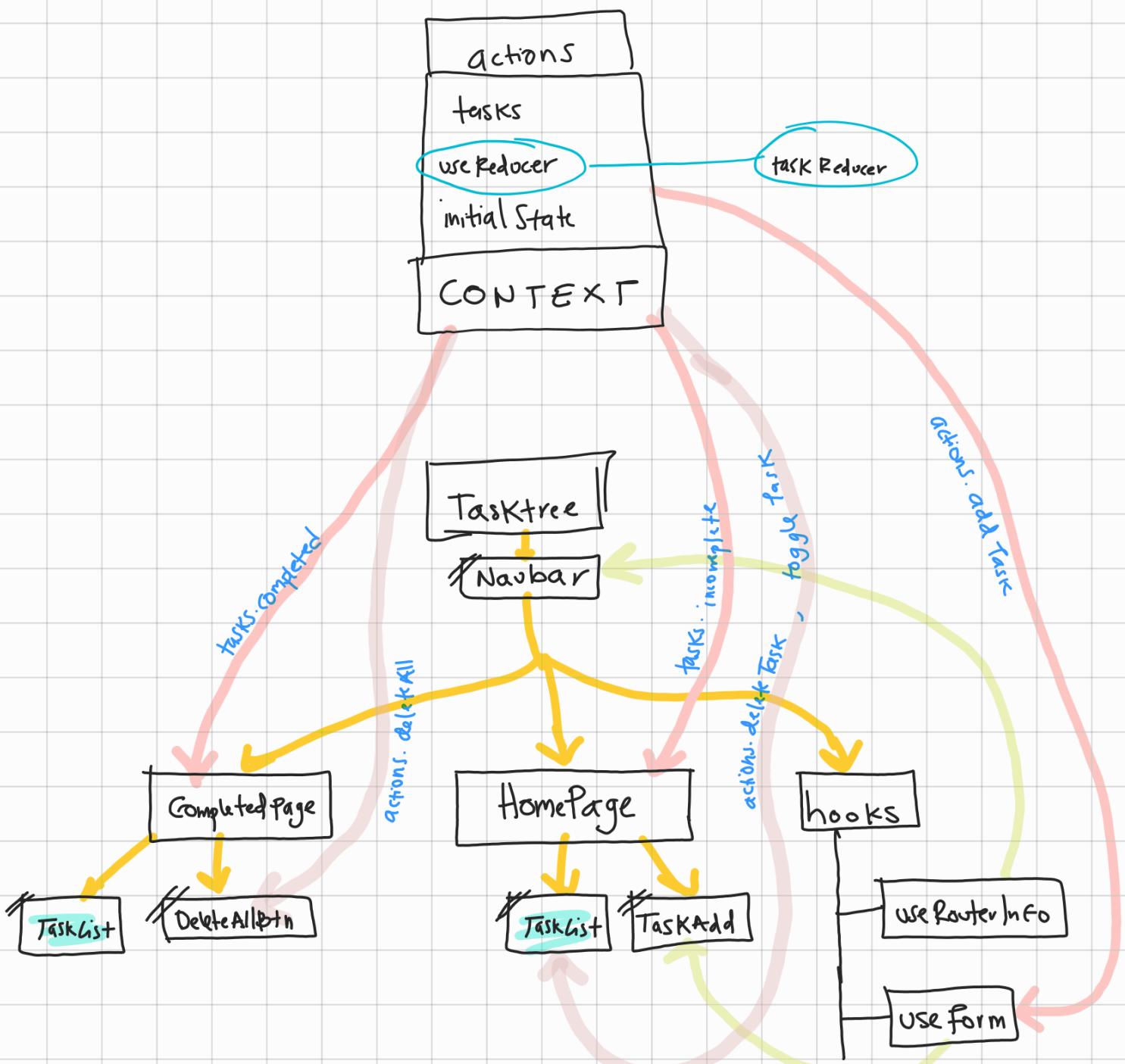


Tasktree VI
by Abril Martínez



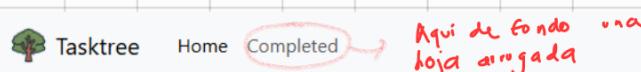
El hecho de que se comparten componentes entre los diferentes Pages, significa que hay que crearlos generalizados para usarlos libremente en varios lugares

Contiene:

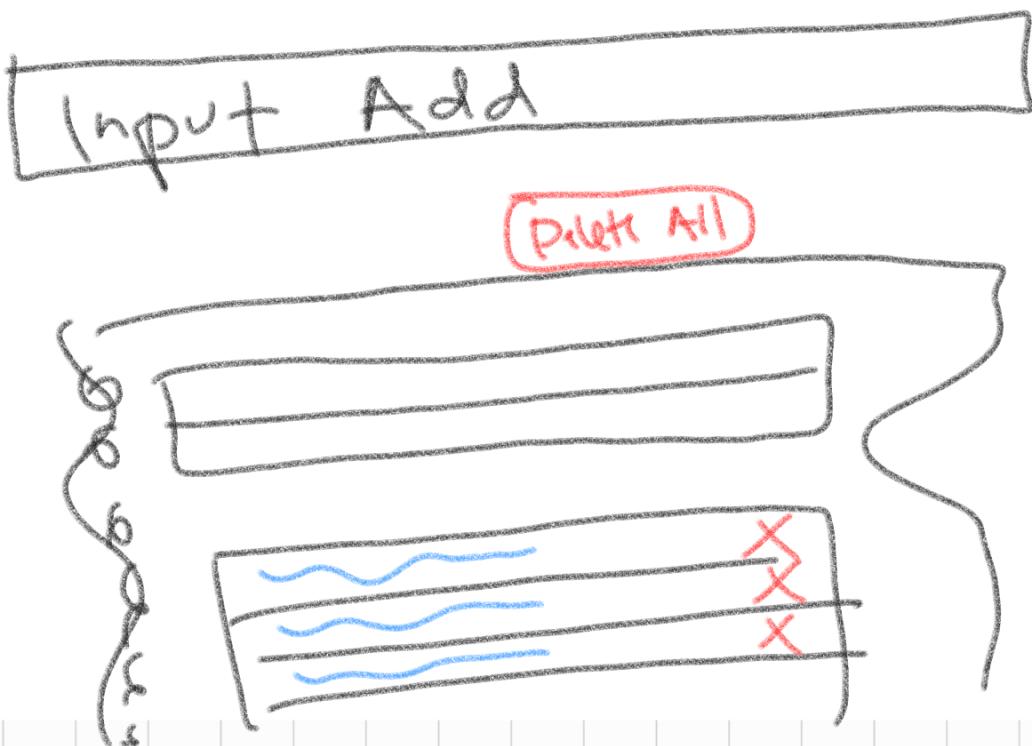
- * React Router
- * Context
- * Reducer
- * Bootstrap

HomePage

Já tenemos el navbar configurado con las rutas, ahora sigue configurar el page. Empezando por HomePage.



HomePage



Creamos los comps TaskAdd y TaskList. Tasklist lo haremos generalizadores porque lo vamos a reutilizar en otras pages.

TaskAdd {
 useForm

TaskList {

Primero creamos los esqueletos de los comps y luego vamos las funcionalidades.

Task Add

+ useForm

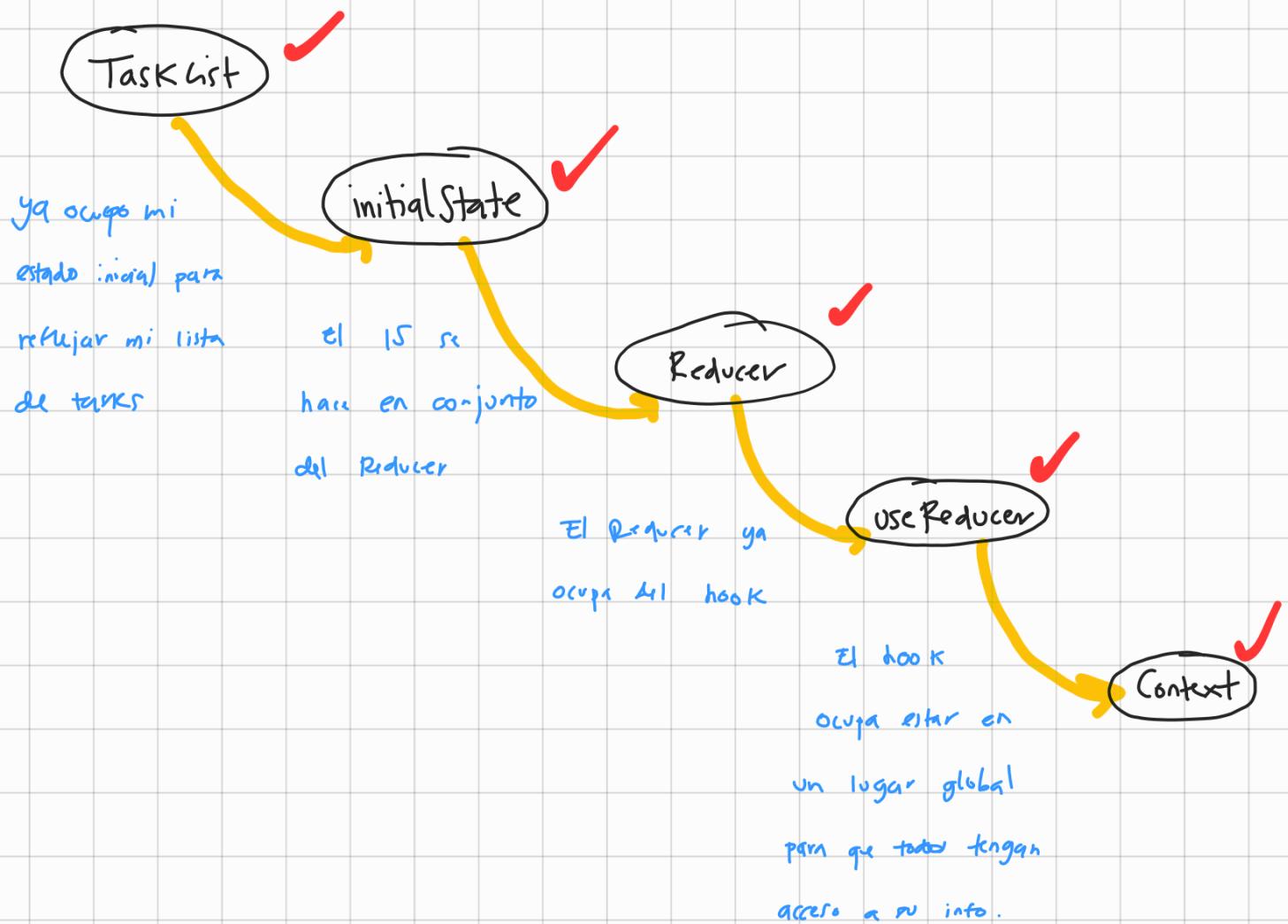
- onChange

- onSubmit

- inputValue

Task list

Teniendo el cascarón (super básico) nos topamos con que ya es necesario crear el Reducer para reflejar nuestro initial state.



Como poner 2 elementos alineados horizontalmente?



Tasktree

Home

Completed

What needs to be done? 📝



2 elementos

Código actual :

```
1 import { useContext } from "react";
2 import { TaskContext } from "../context/TaskContext";
3
4 export const TaskList = () => {
5
6   const { tasks } = useContext( TaskContext );
7
8
9   return (
10     <ul className="list-group">
11       { tasks.map( task => (
12         <li
13           className="list-group-item"
14           key={ task.id }>
15           { task.description }
16           <div className="">
17             <button className="btn btn-danger">
18               X
19             </button>
20           </div>
21         </li>
22       )) }
23     </ul>
24   )
25 }
26 }
```

Entonces organizamos establecer
esa regla en el padre.

Padre

Hermanos



Tasktree

Home Completed

What needs to be done? 🖌

Crear Tasktree



1 <li

2 className="list-group-item d-flex justify-content-between align-items-center"
3 key={ task.id }>espacio en
medio

proporcional

ajuste en medio
horizontalmente

Y ya tenemos el cascarón del HomePage completo.



Tasktree Home Completed

What needs to be done? 🖌

Crear Tasktree



Ir a correr



También se aplicó estilo para pantallas pequeñas.



Tasktree Home Completed

What needs to be done? 🖌

Crear Tasktree



Ir a correr



El TaskAdd sólo imprime el inputValue.

Ahora hay que hacer la conexión entre el Reducer y el onsubmit para que se refleje en TaskList.

Pero recordemos que hay que separar el initialState en 2 listas, los task done true y los tasks done false.

¿Cómo le hacemos para eso?

```
● ● ●  
1 const initialState = {  
2   incomplete: [  
3     {  
4       id: new Date().getDate() * 3,  
5       description: 'Crear Tasktree',  
6       done: false,  
7     },  
8     {  
9       id: new Date().getDate() * 2,  
10      description: 'Ir a correr',  
11      done: false,  
12    }  
13  ],  
14   completed: [  
15     {  
16       id: new Date().getDate() * 5,  
17       description: 'Lavar mi Miata',  
18       done: true,  
19     },  
20     {  
21       id: new Date().getDate() * 4,  
22       description: 'Comprar los boletos de avión a Brazil',  
23       done: true  
24     }  
25   ]  
26 }  
27 }
```

Cambiamos el 1.5 por un objeto con 2 props.

- incomplete
- completed

Dentro de ellos

Creamos un array con los objetos (tasks)

Desactualizado

```
1 export const HomePage = () => {
2
3   const { tasks } = useContext( TaskContext );
4
5   return (
6     <>
7       <TaskAdd/>
8       <TaskList tasks={ tasks.incomplete } />
9     </>
10   )
11 }
```

y para consumirlos usamos el `useContext()` en donde lo requerimos y luego le pasamos el prop incomplete

Ahora sí tenemos nuestro cascarón completo del HomePage. Sigamos con el cascarón del CompletedPage.

Desactualizado

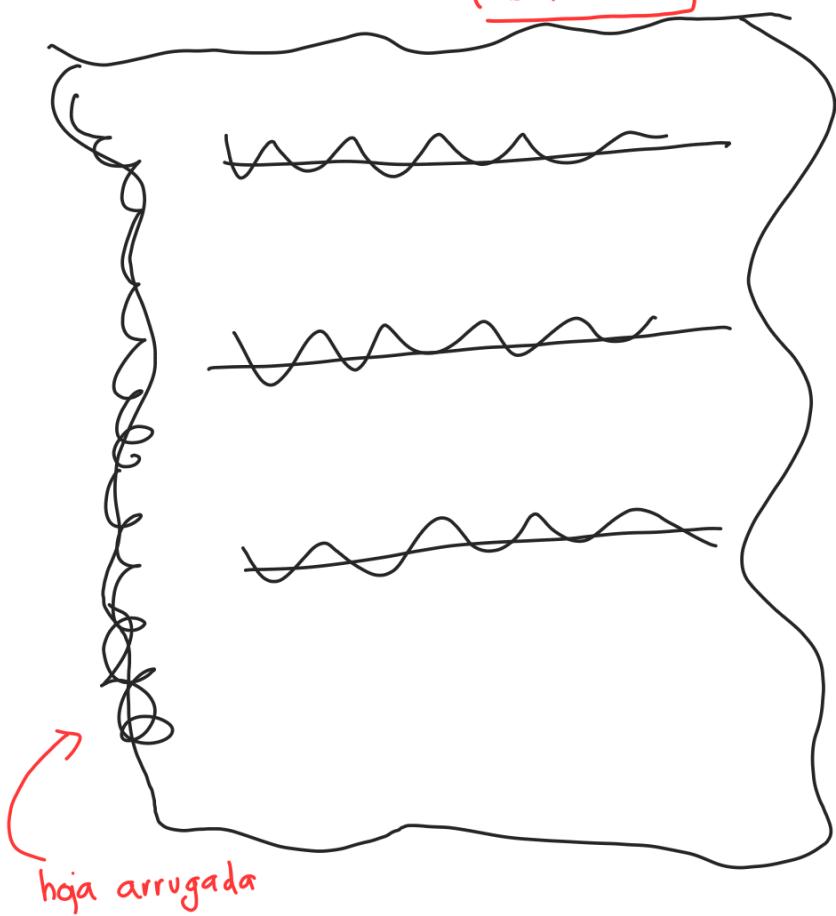
Completed Page



Tasktree

Home Completed

Delete all



hoja arrugada

[Delete all](#)

Lavar mi Miata

Comprar los boletos de avión a Brazil

Ya tenemos el esqueleto del CompletedPage.

Me encontré con 2 problemas:

- Agregar al task.description de /completed'
- Quitar el X btn en el mismo path.



What needs to be done? 📝

Crear Tasktree

X

Ir a correr

X

[Delete all](#)

Lavar mi Miata

Comprar los boletos de avión a Brazil

Para hacer esto es muy fácil.



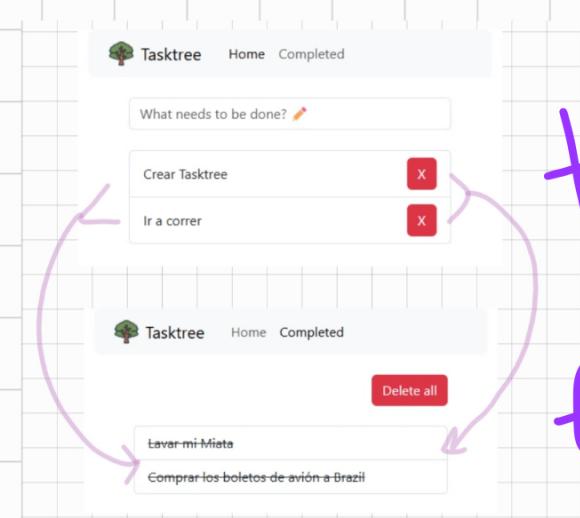
tasktree - TaskList.jsx

```
1 import { useLocation } from "react-router-dom";
2
3 export const TaskList = ({ tasks, img }) => {
4   const location = useLocation();
5
6   return (
7     <ul className="list-group">
8       { tasks.map( task => (
9         <li
10           className="list-group-item d-flex justify-content-between align-items-center"
11           key={ task.id }>
12             {( location.pathname === '/' )
13               ? [
14                 1. { task.description },
15                 2. <button className="btn btn-danger">X</button>
16               ]
17               : [
18                 <del> {task.description} </del>
19               ]
20             }
21           </li>
22         ) )
23       </ul>
24     )
25   )
26 }
27
28 }
```

Aquí dentro están los elementos

Usamos el `useLocation()` para crear el if teniendo dependiendo del pathname.

No se puede usar `if()` en un `{ } de return()`



Recap

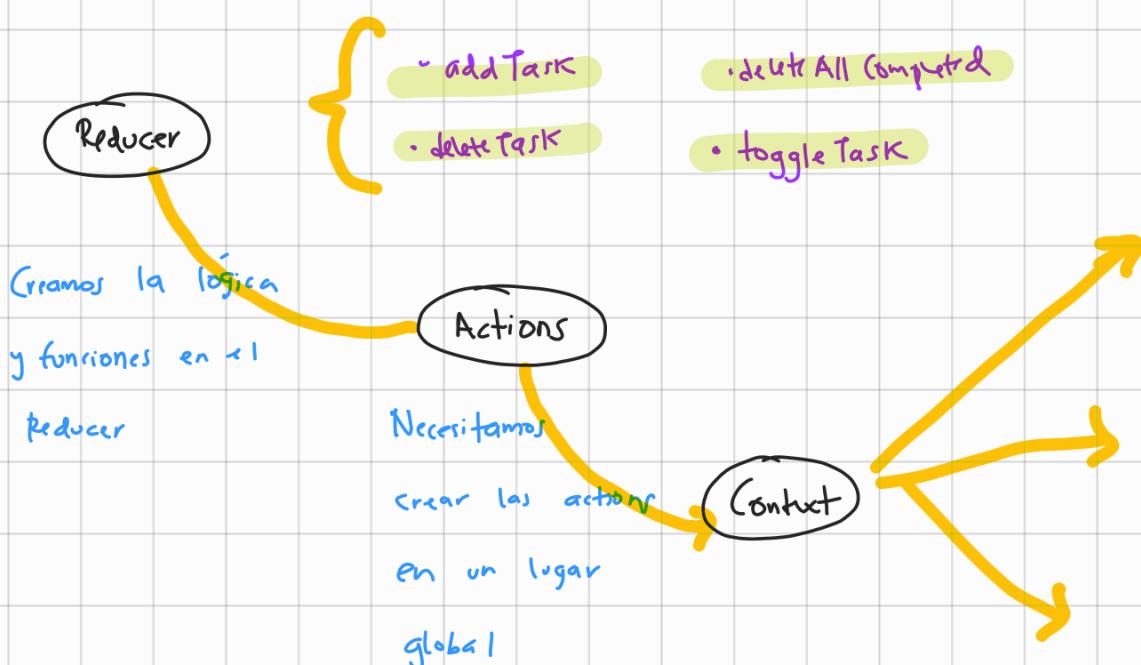
Tenemos creadas las estructuras de ambas Pages.

Falta:

- crear lógica de Reducer
- completar diseño de Pages

Dejari para el final el diseño final.

Actions - Reducer



Cómo hacerle para usar el 'add task' en un initialState que es un objeto con 2 arrays?

initialState {

 incomplete: [],
 completed: []

}

action
addTask

global
Context

TaskAdd
onSubmit

Se crea el action con el Payload

Se pone global

Lo consume TaskAdd

taskReducer Add task

case 'Add todo':

```
return [ ...state.incomplete, action.payload ];
```



ma)

Recordemos que `initialState` es un objeto, no un array.

Y cuando queremos devolver un nuevo estado que es un array,

se debe devolver todas sus propiedades.

```
tasktree - taskReducer.js
```

```
1 export const taskReducer = ( state, action ) => {
2
3   switch( action.type ){
4     case 'Add task':
5       return {
6         ...state,
7         incomplete: [ ...state.incomplete, action.payload ]
8       }
9     }
10    }
11  }
```

Devuelve el objeto que es I.S., con el array NO modificado
y el modificado.

Y de ahí crea un nuevo estado completo.

Delete task



tasktree - taskReducer.js

```
1 case 'Delete task':  
2   return {  
3     ... state,  
4     incomplete: state.incomplete.filter( task => task.id !== action.payload )  
5   }
```

Aquí no se usa incomplete: [...] como en 'Add task' porque filter ya devuelve un array.

Entonces incomplete = []

Toggle task

En el reducer tengo que recibir el id del task que quiero cambiar para comparar el id con todos mis states y sacar el del mismo id para hacer el cambio.

* ¿Va a devolver un estado nuevo?

R = un Reducer, siempre.

Desactualizados

Tenemos el initialState con sus 2 arrays.

initialState {

 incomplete: [{}, {}]

 completed: [{}, {}]

}

Podemos juntar los 2 arrays y crear uno solo.

CAMBIO de estructura

Para mejorar la lógica del ToggleTask he decidido unir el LS en un solo array.

¿Dónde se muestra el LS?

En el HomePage, CompletedPage y TaskList.

¿A qué le pega un cambio de estructura del LS?

A la lógica del Reducer. Add task, Delete task y Delete completed.



Como primero queremos que se vean las listas sin usar las funciones del Reducer, editamos primero las vistas de las listas.

Podemos quitar el flujo de contexto desde cada page y hacerlo directamente en TaskList.

En TaskList podemos separar el LS en dos arrays para que se creen dos variables que tengan la condición que sólo contenga el mismo estado done true o false.



Resultado

1. Quitamos el flujo de tasks de context desde las Pages.
2. Modificamos la lógica de Tasklist.



tasktree - TaskList.jsx

```

1  export const TaskList = ({ img }) => {
2
3    const { tasks, deleteTask } = useContext( TaskContext );
4    const location = useLocation();
5
6
7    const tasksCompleted = tasks.filter( task => task.done === true );
8    const tasksPending = tasks.filter( task => task.done === false );
9
10
11   const onBtnClick = ( task ) => {
12     deleteTask( task.id );
13   };
14
15   Para saber que array usar
16   en map, si hace este if ternario
17
18   return (
19     <ul className="list-group">
20       {(( location.pathname === '/' ) ? tasksPending : tasksCompleted ).map( task => (
21         <li
22           className="list-group-item d-flex justify-content-between align-items-center"
23             key={ task.id }>
24
25           {(
26             location.pathname === '/'
27             ?
28               <>
29                 { task.description }
30             <button
31               onClick={ () => onBtnClick( task ) }
32               className="btn btn-danger">
33                 X
34               </button>
35             : <del> { task.description } </del>
36           )
37         }
38       )})
39     </ul>
40   )
41 }
42 }
```

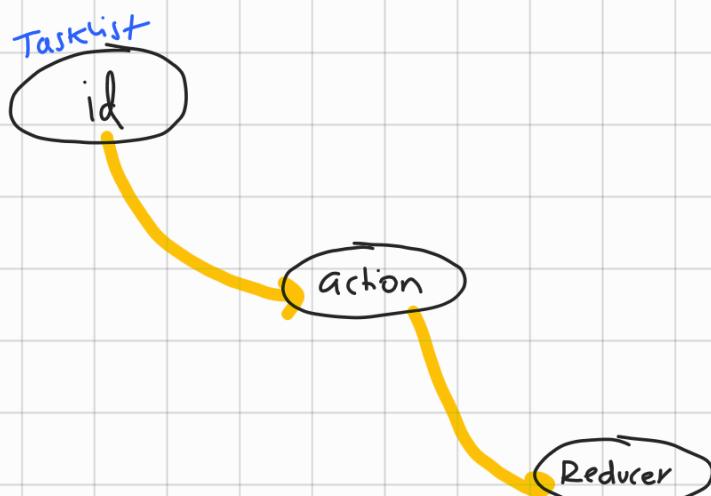
Se consume el contexto para usar los tasks

Se parten los tasks en 2 array dependiendo su estado.

Para saber que array usar en map, si hace este if ternario

esto se queda igual.

Volvemos a Toggle task.



Aquí ocupa el id.

En el Reducer se tiene que recorrer todos los state (tasks) existentes para ver cual coincide con el id y trabajar sobre ese elemento.

1. Recorrer el state

2. Que el id (payload) coincida con un elemento del State.



tasktree - taskReducer.js

```
1 case 'Toggle task':  
2   return state.map( task =>  
3     ( task.id === action.payload )  
4       ? { ...task, done: !task.done }  
5       : task  
6   )
```

LocalStorage



tasktree - TaskProvider.jsx

```
1 const init = () => {
2   return JSON.parse( localStorage.getItem('tasks') ) || initialState;
3 }
4
5
6 export const TaskProvider = ({ children }) => {
7
8   const [ tasks, dispatch ] = useReducer( taskReducer, initialState, init );
9
10  useEffect(() => {
11    localStorage.setItem('tasks', JSON.stringify( tasks ));
12  }, [tasks]);
```

Cosas que faltan

* Agregarle el diseño completo.

Cosas que quiero agregar

* Agregar Snackbar al borrar un item en HomePage ✓

* Agregar otro también al borrar todos en CompletedPage ✓

* Otro también al completar un items ✓

* Agregar un texto cuando no hay tasks en cada Page. ✓

Toast Notification

Utilicé Sonner.

Pendiente:

- * Agregar 'Add something' cuando la lista está vacía. ✓
- * Actualizar estas notas
- * Subir a Github
- * Actualizar mi perfil de Github

<https://sonner.emilkowal.ski/> Sonner (emilkowal.ski)

