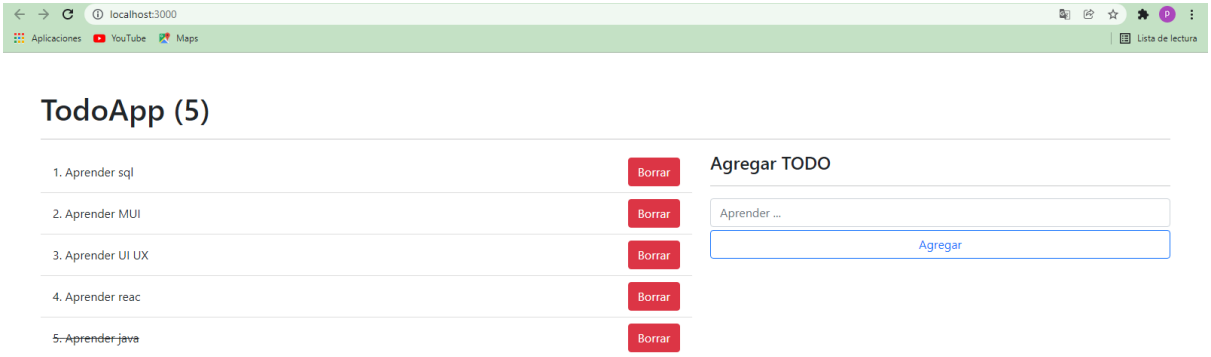
Resumen

La app TodoApp escrita en react,consiste en poder agregar/borrar tareas a una lista, y marcarlas como realizadas.

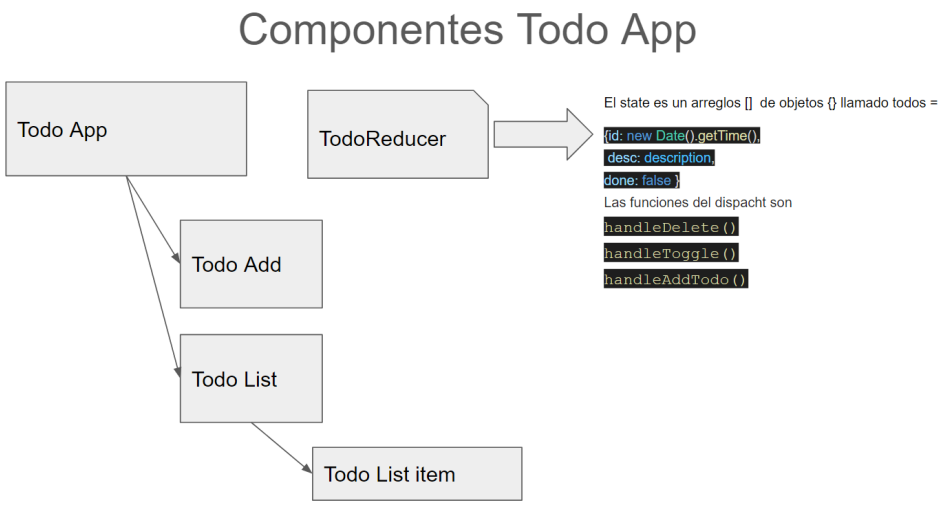


Objetivo

Documentar todo los conocimientos que se consideren relevantes al momento de escribir una app del tipo Todo App

Introducción

La aplicación tine cuatro componentes



Desarrollo

Componente Todo-App

El componente principal TodoApp posee un hook reducer “TodoReducer” encargado de mantener el estado de la información de la lista “todos” y tres funciones que disparan distintos dispatch

El state es un arreglos [] de objetos {} llamado todos =

{id: new Date().getTime(),   // ID generado con date
 desc: description, // Tarea
done: false } // Estado de tarea realizada

Las funciones del dispatch son
handleDelete()
handleToggle()
handleAddTodo()

El state se lee del localStorage al iniciar la app

```
const init = () => {

    // Si no hay todos devuelve un arreglo vacío y si no el arreglo.
    // El jSon parse convierte el string del localStorage en un Json
    // El localStorage almacena string

    return JSON.parse(localStorage.getItem('todos')) || [];
}
```

Se utiliza el hook useEffect para monitorear el cambio de los “todos” y si cambian guardarlos en el localStorage

```
// El localStorage guarda string por eso la conversión JSON.stringfy
    useEffect(() => {
        localStorage.setItem('todos',JSON.stringify(todos));

    }, [todos])// Cada vez que cambia todos guarda en el localStorage
```

EL Todo Reducer.js es el encargado de ejecutar las acción que se pasan en el dispatch

```
export const todoReducer = (state = [] , action )=>{

 switch(action.type){

    case 'add':
        return [ ...state, action.payload ];

        // El .filter devuelve un nuevo arreglo ,con todos los
        // elemento del arreglo que cumpla la condición . Ver mozilla
Doc
        // tambien puedo poner action.payload.id, ya va a comparar con
el .id
        // Como estoy mandado payload = todo.id dejo solo payload

    case 'delete':
        return state.filter( todo => todo.id !== action.payload);

    // Mismo resultado que toggle-old con solución ternario con return
implícito

   case 'toggle':
       return state.map ( todo=>
        (todo.id === action.payload )
            ? {...todo,done: !todo.done}
            : todo
            );
   /*
     Recorre todo el state=todos
    cuando encuentra el id toggle el valor de done,
    importante : si no encuentra nada devuelve el todo,
    para no devolver undefined
   */

    case 'toggle-old':
        return state.map ( todo => {
            if (todo.id === action.payload ){
                return {
                    ...todo,
                    done: !todo.done
                }
            }else {
```

```
                return todo;
            }
        })
    default: //Se llama al inicializar
        return state;

 }

}
```

Componente Todo-List

Este componente se encarga de iterar los “todos” con la función map e imprime el componentes Todo-List-Item en esta iteración .

```
export const TodoList = ({ todos, handleDelete, handleToggle }) => {
    return (
        <ul className="list-group list-group-flush">
        {
            todos.map( (todo, i) => (
                <TodoListItem
                    key={ todo.id }
                    todo={ todo }
                    index={ i }
                    handleDelete={ handleDelete }
                    handleToggle={ handleToggle }
                />
            ))
        }
    </ul>
    )
}
```

Componente Todo-List-Item

Este componente se encarga de generar cada ítem de la lista,
El ítem posee un clase de css en style.css

```
.complete {
    text-decoration: line-through;
}
```

que lo marca como tachado si todo.done === true

El Ítem posee un botón de borrar, que elimina el ítem de la lista.

Estas dos acciones la ejecuta mediante las funciones pasadas como props ,handle Toggle y handle Delete ,

```
import React from 'react';

export const TodoListItem = ({ todo, index, handleDelete, handleToggle
}) => {



    return (
        <li
            key={ todo.id }
            className="list-group-item"
        >
            <p
                className={ `${ todo.done && 'complete' }` }
                onClick={ () => handleToggle( todo.id ) }
            >
                { index + 1}. { todo.desc }
            </p>
            <button
                className="btn btn-danger"
                onClick={ () => handleDelete( todo.id ) }
            >
                Borrar
            </button>
        </li>
    )
}
```

Componente Todo-Add

Este componente da soporte al formulario que agrega tareas ,tiene un custom hook UseForm,js que mantiene estados de formularios de manera genérica

```
import { useState } from "react"


export const useForm = ( initialState = {} ) => {

    const [values, setValues] = useState(initialState);

    const reset = () => {
        setValues (initialState);
    }

    const handleInputChange = ({ target }) => {

        setValues({
            ...values,
            [ target.name ]: target.value
        });

    }

    return [ values, handleInputChange,reset ];

}
```

Todo-Add.js

```
import React from 'react'
import { useForm } from '../../hooks/useForm';

export const TodoAdd = ({ handleAddTodo }) => {

    const [ { description }, handleInputChange, reset ] = useForm({
        description: ''
    });

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

        if ( description.trim().length <= 1 ) {
            return;
        }

        const newTodo = {
            id: new Date().getTime(), // Id generado con el date
            desc: description,        // Tarea
            done: false               // Estado de tarea realizada.
        };

        handleAddTodo( newTodo );
        reset();

    }


    return (
        <>
            <h4>Agregar TODO</h4>
            <hr />

            <form onSubmit={ handleSubmit }>

                <input
                    type="text"
                    name="description"
                    className="form-control"
                    placeholder="Aprender ..."
                    autoComplete="off"
                    value={ description }
```

```
                    onChange={ handleInputChange }
                />

                <button
                    type="submit"
                    className="btn btn-outline-primary mt-1 btn-block"
                >
                    Agregar
                </button>


            </form>

        </>
    )
}
```