|  |
| --- |
| Todo React |
|  |

Técnica para usar o module.css

Criar arquivo: Global.d.ts

declare module “\*.module.css”

**Iniciando ações de form**

**Task.ts**

export interface ITask {//nome da interface sempre tem um I no começo

  id: number;//ajudará na remoção e edição

  title: string;

  difficulty: number;

}

**TaskForm.tsx**

const [id, setId] = useState<number>(0); //tipo:number - começa como '0'

  const [title, setTitle] = useState<string>(""); //tipo:string - começa vazio

  const [difficulty, setDifficulty] = useState<number>(0); //dificuldade da tarefa

  const addTaskHandler = () => {}; //realiza ação de inclusão de tarefas

  const handleChange = (e: ChangeEvent<HTMLInputElement>) => {

    //tipando event como elemento html

    if (e.target.name === "title") {

      setTitle(e.target.value);

    } else {

      setDifficulty(parseInt(e.target.value)); //transformando string em number

    }

    //console.log(title);

    //console.log(difficulty);

  };

<form onSubmit={addTaskHandler} className={styles.form}>

      {/\*quando formulário for enviado, função addTaskHandler é executada\*/}

      <div className={styles.input\_container}>

        <label htmlFor="title">Título:</label>

        <input

          type="text"

          name="title"

          placeholder="Título da tarefa"

          onChange={handleChange} //quando há mudança no elemento, evento é disparado

        />

      </div>

      <div className={styles.input\_container}>

        <label htmlFor="difficulty">Dificuldade:</label>

        <input

          type="text"

          name="difficulty"

          placeholder="Dificuldade da tarefa"

        />

      </div>

      <input type="submit" value={btnText} onChange={handleChange} />

    </form>

**Importação(App.tsx, TaskForm.tsx)**

//interface

import { ITask } from "./interfaces/Task";

**Adicionando tarefas**

**App.tsx**

import React, {useState} from "react";

const [taskList, setTaskList] = useState<ITask[]>([]); //é um array de ITask e começará com um array vazio

<TaskForm

            btnText="Criar tarefa"

            taskList={taskList}

            setTaskList={setTaskList}

          />

**TaskForm.tsx**

type Props = {

  taskList: ITask[]; //chamando lista

  setTaskList? : React.Dispatch<React.SetStateAction<ITask[]>>; //dispachando um evento(setState) que trabalha com ITask//?-pode ou não vir(opcional)

};

const TaskForm = ({ btnText, taskList, setTaskList }: Props) => {

  const [id, setId] = useState<number>(0); //tipo:number - começa como '0'

  const [title, setTitle] = useState<string>(""); //tipo:string - começa vazio

  const [difficulty, setDifficulty] = useState<number>(0); //dificuldade da tarefa

  const addTaskHandler = (e: FormEvent<HTMLFormElement>) => {

    //realiza ação de inclusão de tarefas - envio de submissão do formulário

    e.preventDefault();

    const id = Math.floor(Math.random() \* 1000); //id aleatório para ser possível se referir facilmente a tarefa a ser incluída ou deletada

    const newTask: ITask = { id, title, difficulty }; //nova tarefa a ser adicionada

    setTaskList!([...taskList, newTask]); //adiciona tudo que tem na taskList e as novas tasks e junta tudo em um array

    //limpando formulário após envio dos dados da tarefa

    setTitle("");

    setDifficulty(0);

    //console.log(taskList)

  };

const addTaskHandler = (e: FormEvent<HTMLFormElement>) => {

    //realiza ação de inclusão de tarefas - envio de submissão do formulário

    e.preventDefault();

    const id = Math.floor(Math.random() \* 1000); //id aleatório para ser possível se referir facilmente a tarefa a ser incluída ou deletada

    const newTask: ITask = { id, title, difficulty }; //nova tarefa a ser adicionada

    setTaskList!([...taskList, newTask]); //adiciona tudo que tem na taskList e as novas tasks e junta tudo em um array

    //limpando formulário após envio dos dados da tarefa

    setTitle("");

    setDifficulty(0);

    //console.log(taskList)

  };

**Exibindo lista de tarefas**

**App.tsx**

<div>

          <h2>Suas tarefas:</h2>

          <TaskList taskList={taskList} />

        </div>

type Props = {

  taskList: ITask[];//a taskList é um array de ITask

};

const TaskList = ({ taskList }: Props) => {//desestruturação da taskList

  return (

    <>

      {taskList.length > 0 ? (//checando se existem itens

        taskList.map((task) => (//cada item é nomeado com 'task'

          <div key={task.id}>{/\*exibe título do item(chamado com a key do identificador único\*/}

            <p>{task.title}</p>

          </div>

        ))

      ) : (

        <p>Não há tarefas cadastradas</p>

      )}

    </>

  );

};

**HTML – itens da lista**

<div key={task.id}>{/\*key com identificador único\*/}

            <div>

              <h4>{task.title}</h4>{/\*título\*/}

              <p>Dificuldade: {task.difficulty}</p>

            </div>

            <div>

            <i className="bi bi-pencil"></i>

            <i className="bi bi-trash"></i>

</div>{/\*ícones de edição e exclusão vindos do bootstrap\*/}

**Remoção de item da lista**

const deleteTask = (id: number) => {

    //recebe id do elemento

    setTaskList(

      //alterar task

      taskList.filter((task) => {

        //fltra lista, onde cada item é nomeado como task

        return task.id !== id; //retorna elementos que tem id diferente ao que será removido

      }) //sendo assim ele retorna a lista toda, menos o item que eu quero remover

    );

  };

 <TaskList taskList={taskList} handleDelete={deleteTask} />

<i

                  className="bi bi-trash"

                  onClick={() => {

                    handleDelete(task.id);

                  }}

                ></i>{/\*função não é passada direto para não ser executada assim que a página for carregada, por isso, precisa ser passada por uma arrowfucntion \*/}

**Criando modal**

Criado componentes:

***Modal*.tsx** e **Modal.*module*.css**.

Importação no **App.tsx**

import Modal from "./components/Modal";

  return (

    <div>

      <Modal />

      <Header />

**Estilizando modal**

**App.tsx**

<Modal

        children={<TaskForm btnText="Editar Tarefa" taskList={taskList} />}

      />

      {/\*component é enviado como propriedade e impresso no Modal de edição\*/}

**Modal.tsx**

type Props = {

  children: React.ReactNode; //basicamente dizendo que JSX será usado dentro desse children

};

const Modal = ({ children }) => {

  return (

    <div id="modal">

      <div className={styles.fade}></div>

      <div className={styles.modal}>

        <h2>Texto modal</h2>

        {children}

**Update**

task?: ITask | null; //opcional e derivada de ITask - valor pode vir preenchido ou vazio

  handleUpdate?(id: number, title: string, difficulty: number): void;

}; //? - opcional

  const addTaskHandler = (e: FormEvent<HTMLFormElement>) => {

    //realiza ação de inclusão de tarefas - envio de submissão do formulário

    e.preventDefault();

    if (handleUpdate) {

      handleUpdate(id, title, difficulty);

    } else {

      const id = Math.floor(Math.random() \* 1000); //id aleatório para ser possível se referir facilmente a tarefa a ser incluída ou deletada

      const newTask: ITask = { id, title, difficulty }; //nova tarefa a ser adicionada

      setTaskList!([...taskList, newTask]); //adiciona tudo que tem na taskList e as novas tasks e junta tudo em um array

      //limpando formulário após envio dos dados da tarefa

      setTitle("");

      setDifficulty(0);

    }

**Fechar modal**

***todo\_react/src/App.tsx***

const hideOrShowModal = (display: boolean) => {

  //será feito uma condição de true e false(display:boolean) para ativar e desativar modal

  const modal = document.querySelector("#modal");

  if (display) {

    //se display for true, é para exibir modal, logo, hide é removida

    modal!.classList.remove("hide");

  } else {

    modal!.classList.add("hide");

  } //adiciona class hide(modal oculto)

};

const editTask = (): void => {

  hideOrShowModal(true);

};

<TaskList

          taskList={taskList}

          handleDelete={deleteTask}

          handleEdit={editTask}

        />

***todo\_react/src/components/Modal.tsx***

const Modal = ({ children }: Props) => {

  const closeModal = (e: React.MouseEvent): void => {

    //método para fechamento do modal//MouseEvent - pois será um click no componente de react

    const modal = document.querySelector("#modal");

    modal!.classList.add("hide"); //adicionando hide a lista de classes(quando closeModal é ativado, hide está sendo enviado)

  }; //exclamação! indica que com certeza item vai ser enviado

  return (

    <div id="modal" className="hide">{/\*hide faz com que item fique oculto\*/}

      <div className={styles.fade} onClick={closeModal}></div>

***todo\_react/src/components/TaskList.tsx***

type Props = {

  handleEdit(): void;

};

const TaskList = ({ taskList, handleDelete, handleEdit }: Props) => {

  <i

                  className="bi bi-pencil"

                  onClick={() => {

                    handleEdit();

                  }}

                ></i>

}

**Preenchendo modal com dados**

***todo\_react/src/App.tsx***

const [taskToUpdate, setTaskToUpdate] = useState<ITask | null>(null);

const editTask = (task: ITask): void => {

  setTaskToUpdate(task); //será alterada baseada em quando clicamos em editar task

};

return (

  <Modal

    children={

      <TaskForm

        btnText="Editar Tarefa"

        taskList={taskList}

        task={taskToUpdate} //a partir disso modal pode ser preenchido

      />

    }

  />

);

***todo\_react/src/components/TaskForm.tsx***

task?: ITask | null//opcional e derivada de ITask - valor pode vir preenchido ou vazio

}; //? - opcional

const TaskForm = ({ btnText, taskList, setTaskList, task }: Props) => {

  useEffect(()=>{//mapeia valor de array de dependência, com isso tem o controle de quando pode ser executado, não tendo um loop infinito

    if(task){

      setId(task.id)

      setTitle(task.title)

      setDifficulty(task.difficulty)

    }//quando vier uma task, retorna dados da tarefa

  }, [task])//quando houver uma alteração useEffect é rodado

}

***todo\_react/src/components/TaskList.tsx***

type Props = {

  handleEdit(task: ITask): void;

}

const TaskList = ({ taskList, handleDelete, handleEdit }: Props) => {

                    handleEdit(task ); //funcionalidade de edição

                <i

**Finalizando Update**

**TaskForm.tsx**

handleUpdate?(id: number, title: string, difficulty: number): void;

const TaskForm = ({

  btnText,

  taskList,

  setTaskList,

  task,

  handleUpdate,

const addTaskHandler = (e: FormEvent<HTMLFormElement>) => {

if (handleUpdate) {

      handleUpdate(id, title, difficulty);

    } else {

      const id = Math.floor(Math.random() \* 1000); //id aleatório para ser possível se referir facilmente a tarefa a ser incluída ou deletada

      const newTask: ITask = { id, title, difficulty }; //nova tarefa a ser adicionada

      setTaskList!([...taskList, newTask]); //adiciona tudo que tem na taskList e as novas tasks e junta tudo em um array

      //limpando formulário após envio dos dados da tarefa

      setTitle("");

      setDifficulty(0);

    }

**App.tsx**

const updateTask = (id: number, title: string, difficulty: number) => {

    //parâmetros necessários para atualizar uma tarefa

    const updateTask: ITask = { id, title, difficulty };

    const updateItems = taskList.map((task) => {

      //loop pela taskList - nomeando cada task como 'task'

      return task.id === updateTask.id ? updateTask : task; //se o id da task atual do loop for igual a do updateTask, se for atualiza a tarefa, se não, continua igual

    });

    setTaskList(updateItems); //atualizando listas do componente

    hideOrShowModal(false);

  };

<TaskForm

            handleUpdate={updateTask}

          />