Projeto final -Módulo 22

Código fonte

package.json

```
{
  "name": "appl-notas",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.14.1",
    "@testing-library/react": "^13.0.0",
    "@testing-library/user-event": "^13.2.1",
    "@types/jest": "^27.0.1",
    "@types/node": "^16.7.13",
    "@types/react": "^18.0.0",
    "@types/react-dom": "^18.0.0",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-icons": "^5.2.1",
    "react-scripts": "5.0.1",
    "typescript": "^4.4.2",
    "web-vitals": "^2.1.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "@babel/preset-env": "^7.25.3",
    "@babel/preset-react": "^7.24.7",
```

```
    "babel-jest": "^29.7.0",
    "jest": "^29.7.0",
    "react-test-renderer": "^18.3.1"
  }
}
```

@types

Notas.d.tsx

```
        export interface INotas {
    id: number;
    titulo: string;
    conteudo: string;
    data: Date;
    prioridade: string;
  }
  export type NotasContextType = {
    notas: INotas[];
    nota: INotas;
    notasFiltradas: INotas[],
    setNotasFiltradas: (notas: INotas[]) => void;
    criandoNota: (nota: INotas) => void;
    salvarNota: (nota: INotas) => void;
    editandoNota: (id: number) => void;
    atualizarNota: (id: number, notaAtualizada:INotas) => void;
    deletarNota: (id: number) => void;
  };
```

Context

Notas.context.tsx

```
import { useState, useEffect, createContext } from "react";
import { INotas, NotasContextType } from "../@types/notas.d";

export const NotasContext = createContext<NotasContextType | null>(null);

const NotasProvider = ({children}: {children: React.ReactNode}) => {
  const [notas, setNotas] = useState<INotas[]>([
    {
      id: 1,
      titulo: "Nota 1",
      conteudo: "Conteudo 1",
      data: new Date('2024-06-03T18:13:26Z'),
      prioridade: "baixa",
    },
    {
      id: 2,
      titulo: "Nota 2",
      conteudo: "Conteudo 2",
      data: new Date('2024-06-01T18:13:26Z'),
```

```
    prioridade: "media",
  },
  {
    id: 3,
    titulo: "Nota 3",
    conteudo: "Conteudo 3",
    data: new Date('2024-06-02T18:13:26Z'),
    prioridade: "alta",
  },
]);

const [nota, setNota] = useState<INotas>({
  id: notas.length + 1,
  titulo: "",
  conteudo: "",
  data: new Date(),
  prioridade: "baixa",
});


const[notasFiltradas, setNotasFiltradas] = useState<INotas[]>([]);



useEffect(() => {
  const notasOrdenadasPorData = notas.sort((a: INotas, b: INotas) => {
    return new Date(b.data).getTime() - new Date(a.data).getTime();
  });
  setNotas(notasOrdenadasPorData);
  setNotasFiltradas(notasOrdenadasPorData);

}, [notas]);


const salvarNota = (todo: INotas) => {
  console.log(todo);

  const novaNota: INotas = {
    id: notas.length + 1,
    titulo: todo.titulo,
    conteudo: todo.conteudo,
    data: new Date(),
    prioridade: todo.prioridade,
  };
  setNotas([...notas, novaNota]);
  setNota({
    id: notas.length + 1,
    titulo: "",
    conteudo: "",
    data: new Date(),
    prioridade: "baixa",
  });
```

```
};

const editandoNota = (id: number) => {
  notas.filter((nota: INotas) => {
    if (nota.id === id) {
      setNota(nota);
    }
  });
};

const criandoNota = (nota: INotas) => {
  setNota(nota);
};

const atualizarNota = (id: number, notaAtualizada: INotas) => {
  const n = notas.find((nota: INotas) => nota.id === id);
  const notasDsatualizadas = notas.filter((nota: INotas) => nota.id !== id);
  setNotas(notasDsatualizadas);

  if (n) {
    n.titulo = notaAtualizada.titulo;
    n.conteudo = notaAtualizada.conteudo;
    n.prioridade = notaAtualizada.prioridade;

    setNotas([...notasDsatualizadas, n]);
  }

  setNota({
    id: notas.length + 1,
    titulo: "",
    conteudo: "",
    data: new Date(),
    prioridade: "baixa",
  });
};

const deletarNota = (id: number) => {
  setNotas(notas.filter((nota: INotas) => nota.id !== id));
};

return (
  <NotasContext.Provider
    value={{
      nota,
      notas,
      criandoNota,
      salvarNota,
      editandoNota,
      atualizarNota,
      deletarNota,
      notasFiltradas,
      setNotasFiltradas
```

```
    }}
   >
    {children}
   </NotasContext.Provider>
  );
};

export default NotasProvider;
```

Componentes

AdicionarNota.tsx

```
import * as React from "react";
import { NotasContext } from "../context/notas.context";
import { NotasContextType } from "../@types/notas.d";

import { FaPlus } from "react-icons/fa6";

const AdicionarNota = () => {
  const { nota, criandoNota, atualizarNota, salvarNota } = React.useContext(
    NotasContext
  ) as NotasContextType;

  const handleInputChange = (
    e: React.ChangeEvent<
      HTMLInputElement | HTMLTextAreaElement | HTMLSelectElement
    >
  ) => {
    criandoNota({
      ...nota,
      [e.target.name]: e.target.value,
    });
  };

  const handleSubmit = (e: React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();
    if (nota.id !== 0) {
      atualizarNota(nota.id, nota);
    } else {
      salvarNota(nota);
    }
  };

  return (

        <form onSubmit={handleSubmit}>
          <input
            type="text"
            required
            name="titulo"
            placeholder="Titulo"
```

```
          value={nota.titulo}
          onChange={handleInputChange}
        />

        <textarea
          name="conteudo"
          rows={5}
          required
          placeholder="Conteudo"
          value={nota.conteudo}
          onChange={handleInputChange}
        ></textarea>
          <label>Prioridade


        <select id="prioridade"
          name="prioridade"
          value={nota.prioridade}
          onChange={handleInputChange}
          >
          <option value="baixa">Baixa</option>
          <option value="media">Media</option>
          <option value="alta">Alta</option>
        </select>
          </label>
        <button title="Adicionar nota" type="submit">
          <FaPlus />
        </button>
      </form>

  );
};

export default AdicionarNota;
```

Cabecalho.tsx

```
function Cabecalho() {
  const logo = (
    <img
      src="https://www.gstatic.com/images/branding/product/1x/keep_2020q4_48dp.png"
      alt="logo"
    />
  );
  return (
    <div className="header">
      {logo}
      <h1>Keep</h1>
    </div>
  );
}
```

```
export default Cabecalho;
```

Notas.tsx

```
import * as React from "react";
import { NotasContext } from "../context/notas.context";
import { INotas, NotasContextType } from "../@types/notas.d";

import { FaTrashCan } from "react-icons/fa6";
import { FaRotate } from "react-icons/fa6";

const Notas = () => {

  const {
    editandoNota,
    deletarNota,
    notas,
    notasFiltradas,
    setNotasFiltradas,
  } = React.useContext(NotasContext) as NotasContextType;

  const filtrarNotas = (prioridade: string) => {
    console.log(prioridade);
    setNotasFiltradas(
      notas.filter((nota: INotas) => {
        if (prioridade === "") {
          return notas;
        } else {
          return nota.prioridade === prioridade;
        }
      })
    );
  };

  return (
    <div>
      <div className="circulos">
                <div  className="circle  todas"  title="Mostrar  todas  as  notas"  onClick={()  =>
filtrarNotas("")}></div>
        <div title="Mostar somente notas com prioridade baixa"
          className="circle baixa"
          onClick={() => filtrarNotas("baixa")}
        ></div>
        <div title="Mostar somente notas com prioridade media"
          className="circle media"
          onClick={() => filtrarNotas("media")}
        ></div>
        <div
        title="Mostar somente notas com prioridade alta"
        className="circle alta" onClick={() => filtrarNotas("alta")}></div>
      </div>
```

```
      <ul className="note">
        {notasFiltradas.map((nota: INotas) => (
          <li key={nota.id} className={nota.prioridade} data-testid={`nota-id-${nota.id}`} >
            <h2>{nota.titulo}</h2>
            <p>{nota.conteudo}</p>

            <p style={{ paddingTop: "50px", fontSize: "0.8rem" }}>
              {nota.data.toLocaleDateString()} -- {nota.prioridade}
            </p>
            <button title="Remover nota"  onClick={() => deletarNota(nota.id)} className="delete">
              <FaTrashCan className="icones" />
            </button>
                              <button title="Atualizar nota" onClick={() => editandoNota(nota.id)}
className="atualiza">
              <FaRotate className="icones" />
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default Notas;
```

Rodape.tsx

```
import React from "react";

function Rodape() {
  return (
    <div className="footer">
      <p>Ⓒ Desenvolvido por Luciana S. Felix</p>
    </div>
  );
}

export default Rodape;


import React from "react";

function Rodape() {
  return (
    <div className="footer">
      <p>Ⓒ Desenvolvido por Luciana S. Felix</p>
    </div>
  );
}

export default Rodape;
```

Testes

AdicionarNota.test.tsx

```
import React, {act} from "react";
import { fireEvent, render, screen } from "@testing-library/react";
import AdicionarNota from "./components/AdicionarNota";
import NotasProvider from "./context/notas.context";
import Notas from "./components/Notas";

describe("AdicionarNota component", () => {

   test("matches snapshot", () => {
   const { container } =     render(<AdicionarNota />,{ wrapper: NotasProvider})
   expect(container).toMatchSnapshot();
});


   test("renders AdicionarNota component", () => {
     render(<AdicionarNota />,{ wrapper: NotasProvider})


   const tituloInput = screen.getByPlaceholderText("Titulo");
   expect(tituloInput).toBeInTheDocument();

   const conteudoTextarea = screen.getByPlaceholderText("Conteudo");
   expect(conteudoTextarea).toBeInTheDocument();

   const prioridadeSelect = screen.getByLabelText("Prioridade");
   expect(prioridadeSelect).toBeInTheDocument();

   const adicionarButton = screen.getByTitle("Adicionar nota");
   expect(adicionarButton).toBeInTheDocument();
 });


 test("adiciona uma nova nota", () => {
   render(<AdicionarNota />,{ wrapper: NotasProvider})

   const tituloInput = screen.getByPlaceholderText("Titulo");
   fireEvent.change(tituloInput, { target: { value: "Nota 4" } });
   expect(tituloInput).toHaveValue("Nota 4");

   const conteudoTextarea = screen.getByPlaceholderText("Conteudo");
   fireEvent.change(conteudoTextarea, { target: { value: "Conteudo 4" } });
   expect(conteudoTextarea).toHaveValue("Conteudo 4");

   const prioridadeSelect = screen.getByLabelText("Prioridade");
   fireEvent.change(prioridadeSelect, { target: { value: "alta" } });
   expect(prioridadeSelect).toHaveValue("alta");
```

```
    const adicionarButton = screen.getByTitle("Adicionar nota");
    fireEvent.click(adicionarButton);

    const novo = screen.getByPlaceholderText("Titulo");
    expect(novo).toHaveValue("");
  });

});
```

Cabecalho.test.tsx

```
import React from "react";
import { render } from "@testing-library/react";
import Cabecalho from "./components/Cabecalho";

describe("Cabecalho component", () => {

  test("renders Cabecalho component", () => {
    const { getByAltText, getByText } = render(<Cabecalho />);
    const logoElement = getByAltText("logo");
    const headingElement = getByText("Keep");
    expect(logoElement).toBeInTheDocument();
    expect(headingElement).toBeInTheDocument();
  });

  test("matches snapshot", () => {
    const { asFragment } = render(<Cabecalho />);
    expect(asFragment()).toMatchSnapshot();
  });
});
```

Notas.test.tsx

```
import React from "react";
import { render, screen } from "@testing-library/react";
import Notas from "./components/Notas";
import NotasProvider from "./context/notas.context";

describe("Notas component", () => {
  test("renders Notas component", () => {

    render(<Notas />,{ wrapper: NotasProvider})

    const notasElement1 = screen.getByTestId("nota-id-1");
    expect(notasElement1).toBeInTheDocument();


    const notasElement2 = screen.getByTestId("nota-id-2");
    expect(notasElement2).toBeInTheDocument();
```

```
    const notasElement3 = screen.getByTestId("nota-id-3");
    expect(notasElement3).toBeInTheDocument();
  });



  test("matches snapshot", () => {
      const { asFragment } =  render(<Notas />,{ wrapper: NotasProvider})


    expect(asFragment()).toMatchSnapshot();
  });
});
```

Rodape.test.tsx

```
import React from "react";
import { render } from "@testing-library/react";
import Rodape from "./components/Rodape";

describe("Rodape component", () => {
  test("renders Rodape component", () => {
    const { getByText } = render(<Rodape />);
    const footerElement = getByText("Ⓒ Desenvolvido por Luciana S. Felix");
    expect(footerElement).toBeInTheDocument();
  });

  test("matches snapshot", () => {
    const { asFragment } = render(<Rodape />);
    expect(asFragment()).toMatchSnapshot();
  });
});
```