Documentação - To-do list

Descrição:

A **To-Do List** é uma aplicação simples para gerenciamento de tarefas. A ideia é fornecer uma interface simples e intuitiva que ajude os usuários a se manterem organizados, garantindo que suas atividades estejam sempre à vista e fáceis de gerenciar.

Índice

Introdução

A aplicação foi desenvolvida para ajudar na organização das tarefas do dia a dia. O usuário pode facilmente adicionar novas tarefas, atualizar o status das atividades e remover tarefas concluídas. A simplicidade da ferramenta permite que qualquer pessoa, independentemente de conhecimentos técnicos, possa usá-la para se manter produtiva.

Criada com **React**, essa aplicação demonstra como gerenciar estados e interações do usuário de forma eficiente, enquanto oferece uma base sólida para expansão com novas funcionalidades, como filtros de tarefas, categorias, e integração com outras ferramentas de produtividade.

Instalação

Pré-requisitos

Liste os pré-requisitos necessários para rodar o aplicativo.

- Node.js >= 14

- npm ou yarn

Passos para instalar

1. Clone o repositório:

git clone https://github.com/pedrogles/todolist.git

2. Navegue até a pasta do projeto:

cd todolist

3. Instale as dependências:

npm install

ou

yarn install

4. Inicie o servidor de desenvolvimento:

npm start *inicia a aplicação.
npm run json *Inicia o **JSON Server.**

5. *Acesse a aplicação em http://localhost:3000*

Uso

Após a inicialização do projeto e sua abertura no navegador, você verá uma entrada de texto (input) onde poderá inserir o título de uma nova tarefa. Logo abaixo da entrada, há um botão para adicionar a tarefa à lista de tarefas. Ao adicionar uma nova tarefa, ela será exibida na listagem abaixo do campo de texto. Cada item na lista de tarefas incluirá um checkbox para atualizar o status da tarefa como feita/não feita, o título da tarefa e dois botões: um para editar o título e outro para deletar a tarefa.

Comandos Úteis

- `npm start`: Inicia o servidor de desenvolvimento.
- `npm run json`: Inicia o **JSON Server**, um servidor que simula uma API REST

Estrutura do Projeto

```
├── public/                # Arquivos estáticos
├── src/                   # Código-fonte
│   ├── components/        # Componentes reutilizáveis
│   ├── context/           # Context API
│   ├── hooks/             # Hooks customizados
│   ├── layouts/           # Layouts reutilizáveis ou componentes de
estrutura da página
│   ├── utils/             # Funções utilitárias
│   ├── App.js             # Componente principal
│   └── index.js           # Ponto de entrada da aplicação
│   └── index.css          # Estilos globais da aplicação
├── .gitignore              # Arquivos e diretórios que devem ser
ignorados pelo Git
├── db.json                 # Arquivo do banco de dados simulado para
JSON Server
├── package-lock.json      # Controle das versões exatas das
dependências instaladas
├── package.json           # Dependências e scripts
└── README.md              # Documentação
```

Componentes

1. Button

Botão reutilizável que dispara uma ação ao ser clicado.

Props:

- onClick: Função que será executada quando o botão for clicado.
- type: Parâmetro que especifica o tipo do botão.
- `children`: Conteúdo que será exibido dentro do botão.

2. CheckBoxInput

Campo de checkbox para controlar o status de uma tarefa.

Props:

- **onChange**: Função que será executada quando o estado do checkbox for alterado, ou seja, quando ele for marcado ou desmarcado.
- checked: Parâmetro que especifica o status da atividade.

3. Input

Campo de entrada de texto personalizável com label e placeholder.

Props:

- **label**: Texto que será exibido junto ao campo de entrada para descrever seu propósito.
- **id**: Identificador único associado ao campo de entrada, usado para associar o rótulo ao campo.
- **type**: Define o tipo do campo de entrada (como texto, número, senha, etc.), especificando o tipo de dado que pode ser inserido.
- **placeholder**: Texto exibido dentro do campo de entrada como sugestão, desaparecendo quando o usuário começa a digitar.
- **ref**: Referência ao elemento de entrada, permitindo manipulação direta ou foco no campo via código.
- **onChange**: Função que será executada quando o estado do checkbox for alterado, ou seja, quando ele for marcado ou desmarcado.

4. Navbar

Cabeçalho da aplicação contendo o título "To Do List".

Props:

- **onChange**: Função que será executada quando o estado do checkbox for alterado, ou seja, quando ele for marcado ou desmarcado.
- checked: Parâmetro que especifica o status da atividade.

5. NewTaskForm

Formulário para criar uma nova tarefa com input e botão.

6. Title

Componente que exibe um título (h1).

Props:

- `children`: Conteúdo que será exibido como titulo.

7. `Subtitle`

Componente que exibe um subtítulo (h2).

Props:

- `children`: Conteúdo que será exibido como Sub titulo.

8. `ToDoList`

Lista de tarefas, exibindo itens ou uma mensagem quando vazia.

9. `ToDoListItem`

Item individual da lista de tarefas, com opções para editar, deletar e alterar o status.

Props:

- **id**: Identificador único associado ao campo de entrada, usado para associar o rótulo ao campo.
- **name**: Titulo da tarefa.
- **status**: Indica o estado atual da atividade, como "feito" ou "não feito", sendo usado para controlar e exibir o progresso da tarefa.

Hooks

1. `UseToDoList`

Hook customizado que facilita o acesso ao contexto da lista de tarefas, permitindo que os componentes utilizem suas funcionalidades e estado de forma simples.

Contextos

1. ToDoListContext

Contexto que gerencia a lista de tarefas, fornecendo funções para inserir, editar, alterar status e deletar tarefas, além de armazenar o estado da lista.