

Desenvolvimento de uma Lista de Tarefas com React e TypeScript

Por Escola Dnc

Introdução

Neste ebook, vamos explorar o desenvolvimento de uma aplicação de lista de tarefas (to-do list) utilizando React e TypeScript. Através deste projeto prático, aplicaremos conceitos fundamentais de tipagem, componentes e estado no contexto do desenvolvimento front-end moderno. Esta abordagem nos permitirá criar uma aplicação funcional e bem estruturada, aproveitando as vantagens do TypeScript em conjunto com o React.

Configuração Inicial do Projeto

Limpeza do Projeto Base

Para iniciar nosso desenvolvimento, é necessário limpar o projeto base gerado automaticamente:

1. Remova todo o conteúdo dentro dos parênteses da função principal no arquivo App.tsx
2. Elimine importações não utilizadas
3. Apague estados iniciais desnecessários

Após essa limpeza, teremos uma base limpa para começar nossa implementação.

Definição da Interface Principal

O primeiro passo na construção da nossa aplicação é definir a interface principal que representará cada item da nossa lista de tarefas:

```
interface TodoItem {  id: string;  texto: string;  completed: boolean;}
```

Esta interface define a estrutura básica de cada tarefa, contendo:

- **id**: identificador único da tarefa (string)
- **texto**: descrição da tarefa (string)
- **completed**: status de conclusão da tarefa (boolean)

Estado Inicial

```
const [todos, setTodos] = useState<TodoItem[]>([]);
```

Observe o uso de generics (`<TodoItem[]>`) para tipar corretamente nosso estado como um array de `TodoItem` .

Vamos criar a estrutura básica do nosso componente:

```
return ( <div className="app">    <div className="container">
<h1>Lista de Tarefas</h1>      { /* Conteúdo adicional será
adicionado aqui */}           </div> </div>);
```

Para permitir a adição de novas tarefas, implementaremos um input e um botão:

```
const [newTodo, setNewTodo] = useState('');// ... dentro do
return<div className="inputContainer"> <input      type="text"
value={newTodo}      onChange={e => setNewTodo(e.target.value)}  />
<button onClick={adicionarTarefa}>Adicionar Tarefa</button></div>
```

Função de Adição de Tarefas

A função `adicionarTarefa` será responsável por criar e adicionar novas tarefas à lista:

```
const adicionarTarefa = () => {  if (newTodo !== '') {    const  
newId = crypto.randomUUID();    const newTodoItem: TodoItem = {  
id: newId,      texto: newTodo,      completed: false    };  
setTodos([...todos, newTodoItem]);    setNewTodo('');  }  };
```

Esta função:

1. Verifica se o input não está vazio
2. Gera um ID único para a nova tarefa
3. Cria um novo objeto `TodoItem`
4. Adiciona o novo item à lista de tarefas
5. Limpa o input

Renderização da Lista de Tarefas

```
<ol> {todos.map((todo) => (    <li key={todo.id}>          <input
type="checkbox"                checked={todo.completed}          onChange=
{() => marcarCompleto(todo.id)}          />          <span style={{
textDecoration: todo.completed ? 'line-through' : 'none'          }}>
{todo.texto}          </span>          </li>    )}</ol>
```

```
const marcarCompleto = (id: string) => { const todosAtualizados =
  todos.map(todo =>      todo.id === id ? {...todo, completed:
    !todo.completed} : todo  ); setTodos(todosAtualizados);};
```

1. Recebe o ID da tarefa a ser atualizada
2. Cria uma nova lista com o status da tarefa específica invertido
3. Atualiza o estado com a nova lista

Estilização e Melhorias Visuais

Para melhorar a aparência da nossa aplicação, podemos adicionar algumas estilizações básicas:

```
.app { /* Estilos gerais da aplicação */}.container { /* Estilos do container principal */}.inputContainer { /* Estilos para o container de input */}li { /* Estilos para os itens da lista */}/* Adicione mais estilos conforme necessário */
```

Nota: Para manter o foco no TypeScript e React, os estilos detalhados não foram incluídos neste ebook. Recomenda-se criar um arquivo CSS separado para uma melhor organização do código.

Conclusão


Neste ebook, exploramos a criação de uma aplicação de lista de tarefas utilizando React e TypeScript. Abordamos conceitos importantes como:

- Configuração inicial do projeto
- Definição de interfaces com TypeScript
- Uso de hooks do React (`useState`)
- Implementação de lógica de adição e atualização de tarefas
- Renderização condicional e estilização dinâmica

Esta aplicação serve como um excelente ponto de partida para projetos mais complexos, demonstrando como o TypeScript pode ser integrado efetivamente em aplicações React para criar código mais robusto e fácil de manter.

Para expandir este projeto, considere adicionar funcionalidades como:

- Persistência de dados (localStorage ou backend)
- Filtros para visualizar tarefas concluídas/não concluídas
- Edição e exclusão de tarefas
- Testes unitários e de integração



Lembre-se de que a prática constante é fundamental para dominar estas tecnologias. Continue explorando e expandindo este projeto para aprimorar suas habilidades em React e TypeScript.