Configurando um Projeto React com Boas Práticas

Por Escola Dnc

Introdução

Este ebook aborda a configuração inicial de um projeto React, focando em boas práticas de desenvolvimento e ferramentas que melhoram a qualidade e a manutenção do código. Vamos explorar a criação do projeto, a instalação de dependências essenciais, a configuração de formatadores de código e linters, bem como a implementação de imports absolutos para uma melhor organização do código.

Criação do Projeto e Instalação de Dependências

Criando um Novo Projeto React

Para iniciar nosso projeto, utilizamos o comando `npm create bit@latest`. Este comando nos permite criar um novo projeto React com configurações modernas. Após executar o comando, seguimos os passos:

1. Definimos o nome do projeto como "dnc-sales-dashboard"
2. Inserimos o nome do desenvolvedor
3. Selecionamos React como nosso framework de escolha

Instalando Dependências de Desenvolvimento

Após a criação do projeto, instalamos algumas dependências essenciais para melhorar nosso ambiente de desenvolvimento:

```
npm install -D prettier eslint eslint-config-prettier eslint-plugin-prettier eslint-plugin-import
```

Estas dependências incluem:

- **Prettier**: Um formatador de código que ajuda a manter um estilo consistente
- **ESLint**: Uma ferramenta de análise estática que identifica e corrige problemas no código
- **Plugins do ESLint**: Configurações adicionais para integrar o Prettier com o ESLint

Configuração de Formatadores e Linters

Configurando o Prettier

Criamos um arquivo `.prettierrc` na raiz do projeto com as seguintes configurações:

```
{  "semi": false,  "singleQuote": true,  "printWidth": 80,  "tabWidth": 2,  "trailingComma": "es5"}
```

Estas configurações definem:

- Não usar ponto e vírgula no final das linhas
- Usar aspas simples
- Limitar o comprimento das linhas a 80 caracteres
- Usar 2 espaços para indentação
- Usar vírgulas no final de objetos e arrays (estilo ES5)

Configurando o ESLint

Editamos o arquivo `.eslintrc.cjs` para incluir a configuração do Prettier:

```
module.exports = {  // ... outras configurações  extends: [    // ... outras extensões    'eslint-config-prettier'  ],  // ... resto do arquivo}
```

Esta configuração garante que o ESLint e o Prettier trabalhem em harmonia, evitando conflitos de regras.

Implementação de Imports Absolutos

Para melhorar a legibilidade e manutenção do código, configuramos imports absolutos. Isso nos permite importar módulos de forma mais clara e concisa.

Configurando o tsconfig.json

Adicionamos as seguintes configurações ao arquivo `tsconfig.json`:

```
{  "compilerOptions": {    "baseUrl": ".",    "paths": {
"@/*": ["src/*"]    }  }}
```

Esta configuração permite que usemos o prefixo `@` para importar módulos a partir da pasta `src`.

Configurando o vite.config.ts

No arquivo `vite.config.ts`, adicionamos a seguinte configuração:

```
import path from 'path'export default defineConfig({  // ... outras
configurações  resolve: {    alias: {      '@':
path.resolve(__dirname, './src')    }  }})
```

Esta configuração informa ao Vite como resolver os imports absolutos que usamos no código.

Configuração do Ambiente de Desenvolvimento

Extensões do VS Code

Recomendamos a instalação das seguintes extensões no VS Code:

- ESLint
- Prettier

Estas extensões ajudam a manter a consistência do código e a identificar problemas enquanto desenvolvemos.

Configurações do VS Code

Criamos uma pasta `.vscode` na raiz do projeto e adicionamos um arquivo `settings.json` com as seguintes configurações:

```
{  "editor.defaultFormatter": "esbenp.prettier-vscode",
"editor.formatOnPaste": true,  "editor.formatOnType": false,
"editor.formatOnSave": true,  "editor.formatOnSaveMode": "file",
"files.autoSave": "off"}
```

Estas configurações garantem que:

- O Prettier seja usado como formatador padrão
- O código seja formatado ao colar
- O código seja formatado ao salvar

- O auto-save esteja desativado para evitar formatações indesejadas

Configuração para Implantação

Configurando o Vercel

Para garantir que nossas rotas funcionem corretamente quando implantadas na Vercel, criamos um arquivo `vercel.json` na raiz do projeto:

```
{  "rewrites": [    {      "source": "/(.*)",      "destination": "/"    }  ]}
```

Esta configuração assegura que todas as rotas sejam redirecionadas para o arquivo principal do React, permitindo que o React Router funcione corretamente em um ambiente de produção.

Conclusão

Neste ebook, abordamos as configurações iniciais essenciais para um projeto React moderno e bem estruturado. Implementamos ferramentas de formatação e linting, configuramos imports absolutos e preparamos o ambiente para uma implantação eficiente. Seguindo estas práticas, você estará preparado para desenvolver aplicações React de alta qualidade, com código limpo e fácil de manter.

Lembre-se de que estas configurações são um ponto de partida e podem ser ajustadas conforme as necessidades específicas do seu projeto e equipe. Mantenha-se atualizado com as melhores práticas da comunidade React e não hesite em adaptar estas configurações à medida que seu projeto evolui.