Documentação do Projeto React – Sistema FitScore

1. Sidebar (components/Sidebar.tsx) Objetivo: Mostrar um menu lateral fixo com links para páginas diferentes do sistema (login, cadastro, dashboard, formulário, relatório). Hooks usados:
2. useLocation(): obtém o caminho atual da URL (pathname) para destacar o link ativo.
3. useNavigate(): permite navegar programaticamente para outra página.
4. useContext(AuthContext): acessa o estado de autenticação do usuário.
5. useState(true): controle interno para mostrar/esconder botões “Login” e “Cadastrar”. Funções importantes:
6. linkClass(path): cria uma classe CSS condicional para o link ativo.
7. handleLoginClick() e handleRegisterClick(): escondem os botões e redirecionam para login ou cadastro. Fluxo:
8. Sidebar fixa à esquerda (fixed).
9. Usuário não logado: mostra links de login e cadastro.
10. Usuário logado: mostra links de dashboard, formulário e relatório.
11. Exibe botões grandes de login/cadastro se usuário ainda não acessou páginas específicas.
12. AuthContext (contexts/AuthContext/index.tsx) Objetivo: Controlar se o usuário está logado ou não em todo o app. Hooks usados:
13. useState: mantém estaAutenticado (true/false). Funções importantes:
14. login(): define estaAutenticado como true.
15. logout(): define estaAutenticado como false. Fluxo:
16. Qualquer componente dentro do AuthProvider pode acessar o estado de autenticação.
17. Dashboard (pages/Dashboard.tsx) Objetivo: Mostrar uma tabela com candidatos avaliados e seus FitScores. Hooks usados:
18. useState([]): guarda os dados do FitScore.
19. useEffect(): carrega os dados do localStorage quando o componente é renderizado. Fluxo:
20. Pega item fitscore do localStorage.
21. Se existir, popula fitData.
22. Renderiza tabela com: Nome, Email, FitScore e Classificação.
23. Se não houver dados, exibe: “Nenhum candidato avaliado ainda”.
24. Formulário (pages/Form.tsx) Objetivo: Perguntar sobre performance, energia e cultura e calcular o FitScore. Hooks usados:
25. useState<Resposta[]>([]): guarda as respostas do formulário. Funções importantes:
26. handleChange(bloco, pergunta, valor): atualiza estado das respostas.
27. calcularFitScore(): soma valores e converte em porcentagem.
28. getClassificacao(score): define classificação do candidato.
29. handleSubmit(): salva dados no localStorage e mostra alerta com resultado. Fluxo:
30. Exibe perguntas agrupadas por bloco (performance, energia, cultura).
31. Usuário seleciona notas de 1 a 5.
32. Ao enviar, calcula FitScore e classificação.
33. Salva dados junto com usuário no localStorage.

34. Login (pages/Usuario/Login.tsx) Objetivo: Autenticar o usuário e navegar para o formulário. Hooks usados:

35. useState: guarda email, senha, erro e loading.
36. useContext(AuthContext): permite logar o usuário.
37. useNavigate(): redireciona após login. Funções importantes:
38. handleSubmit(): verifica usuário no localStorage, valida email/senha, mostra loading e navega para /form. Fluxo:
39. Usuário digita email e senha.
40. Se estiver correto, mostra tela de carregamento por 2 segundos e navega para /form.

41. Se incorreto, exibe mensagem de erro.

42. Registrar (pages/Usuario/Registrar.tsx) Objetivo: Cadastrar usuário e navegar para o formulário. Hooks usados:

43. Mesmos do Login. Funções importantes:
44. handleSubmit(): salva usuário no localStorage, mostra loading e redireciona. Fluxo:
45. Usuário preenche nome, email e senha.
46. Salva dados no localStorage.
47. Mostra tela de carregamento por 2 segundos.

48. Redireciona para /form.

49. App (App.tsx) Objetivo: Estrutura principal do app, gerenciando rotas e exibindo a Sidebar. Fluxo:

50. Sidebar fixa à esquerda.
51. Conteúdo das páginas renderizado à direita (flex-1).

52. Rotas definidas: /login, /registrar, /form, /dashboard.

53. Main (main.tsx) Objetivo: Renderizar o App no DOM. Fluxo:

54. BrowserRouter: permite rotas no React.
55. AuthProvider: permite usar estado de login em qualquer página.