Ótimo! Fico feliz em saber que o problema do "flash" foi resolvido com o estado de carregamento no AuthProvider. Isso significa que a inicialização da autenticação e o roteamento protegido estão se comportando como esperado.

Agora que temos a base do front-end funcional:

* Estrutura do projeto React com TypeScript (via Vite).
* Cliente HTTP (apiClient com Axios) configurado com interceptors para JWT e tratamento básico de 401.
* Roteamento básico (react-router-dom) implementado.
* Contexto de Autenticação (AuthContext) gerenciando o estado de login, token no localStorage e o estado inicial de carregamento.
* Componente Login funcionando (incluindo chamada API, atualização de estado e redirecionamento).
* Componente ProtectedRoute funcionando para proteger rotas.

**Como podemos dar sequência?**

Temos várias opções para continuar construindo as funcionalidades do "PumpTrain":

1. **Implementar o Registro de Usuários:** Criar o componente Register.tsx com um formulário para nome, email e senha, que chamará o endpoint POST /auth/register da sua API usando o apiClient. É um fluxo similar ao do Login.
2. **Buscar e Exibir Dados Protegidos:** Tornar o componente Dashboard.tsx (ou criar um novo, como WorkoutList.tsx) mais útil. Fazer com que ele, após montar, use o apiClient para chamar um endpoint protegido (ex: GET /api/workouts) e exiba as sessões de treino do usuário logado. Isso validará completamente o fluxo de requisições autenticadas.
3. **Listar Dados Públicos:** Criar um componente (talvez na HomePage ou em uma nova página /exercises) que chame o endpoint público GET /api/exercises e exiba a lista de exercícios disponíveis, demonstrando chamadas a endpoints que não exigem login.
4. **CRUD de Sessões/Atividades:** Começar a construir a interface e a lógica para as operações mais complexas: visualizar detalhes de uma sessão, criar uma nova, adicionar/editar/remover atividades (ActivityLog) dentro de uma sessão.
5. **Estilização / UI Kit:** Começar a integrar uma biblioteca de componentes UI (como Material UI, Chakra UI, Ant Design, etc.) ou focar em melhorar a estilização com CSS/Tailwind para dar uma cara mais profissional à aplicação.
6. **Gerenciamento de Estado Avançado (Opcional por agora):** Se a aplicação ficar muito complexa, podemos discutir o uso de bibliotecas como Zustand ou Redux Toolkit para gerenciar o estado global de forma mais escalável (além da autenticação).