10 — Rotas e proteção

10 — Rotas e proteção

> Três arquivos de dados (`paths.ts`, `routes.ts`, `menuTree.ts`) e um de comportamento (`AppRouter.tsx`). Toda rota é declarada em tabela, nunca em JSX espalhado. Todo componente de página é carregado com `lazy()`.

Separação dados / comportamento

Pasta	Contém	Pode ter JSX?
`src/routes/`	Dados (constantes)	Não
`src/router/`	Montagem do router	Sim

Isso permite consumir a tabela de rotas para outros fins (menu, breadcrumb, verificação de permissão) sem arrastar o React Router junto.

`paths.ts`

```
// src/routes/paths.ts
export const PATHS = {
  LOGIN: "login",
  DASHBOARD: "dashboard",
  CADASTRO: {
    FORNECEDOR: "cadastro/fornecedor",
    TIPO_FORNECEDOR: {
```

```
      INCLUIR: "cadastro/tipo-fornecedor/incluir",      ALTERAR: "cadastro/tipo-
fornecedor/alterar",
      EXCLUIR: "cadastro/tipo-fornecedor/excluir"
    }
  },
  RELATORIOS: {
    MENSAL: "relatorios/mensal",
    ANUAL: "relatorios/anual"
  }
} as const;
```

Regras:

- **Sem barra inicial.** `AppRouter` concatena (`path={`/${path}`}`). Uma barra a mais gera `//rota`.
- `as const` no final — dá tipos literais e impede mutação acidental.
- Chaves em `SCREAMING_SNAKE_CASE`; valores em `kebab-case`.
- Aninhamento espelha a hierarquia de URL, e normalmente a do menu.
- Português nos segmentos de URL (é sistema interno em pt-BR). **Escolha um idioma e mantenha** — ver os anti-padrões em 13.

routes.ts

```
// src/routes/routes.ts
import { lazy } from "react";
import { PATHS } from "./paths";
import { RelatoriosLayout } from "@pages/Relatorios";
export type RouteGuard = "public" | "protected" | "auth-only";
export interface AppRoute {
  path: string;
  element: React.LazyExoticComponent<React.ComponentType>;
  guard: RouteGuard;
  /** Layout intermediário opcional entre o shell e a página */
  layout?: React.ComponentType;
}
export const routes: AppRoute[] = [
```

```
  {
    path: PATHS.LOGIN,
    element: lazy(() => import("@pages/Login")),
    guard: "auth-only"
  },
  {
    path: PATHS.DASHBOARD,
    element: lazy(() => import("@pages/Dashboard")),
    guard: "protected"
  },
  {
    path: PATHS.CADASTRO.FORNECEDOR,    element: lazy(() =>
import("@pages/Cadastro/Fornecedor")),
    guard: "protected"
  },
  {
    path: PATHS.RELATORIOS.MENSAL,    element: lazy(() =>
import("@pages/Relatorios/Mensal")),
    guard: "protected",
    layout: RelatoriosLayout
  }
];
export const NotFoundPage = lazy(() => import("@pages/NotFound"));
```

Guards

Guard	Comportamento	Uso
`auth-only`	Só para **não** autenticados. Autenticado é redirecionado ao dashboard	Login
`public`	Acessível sempre, sem shell	Termos, ajuda
`protected`	Exige sessão; renderiza dentro do shell (`Main`)	Todo o resto

Não existe default: `guard` é obrigatório. Isso é proposital — esquecer torna a rota pública por acidente, e o compilador impede.

Lazy sempre

```
element: lazy(() => import("@pages/Cadastro/Fornecedor"));
```

Toda página é lazy. Sem exceção — até a de login. Cada tela vira um chunk próprio; o bundle inicial não cresce com o sistema. Por isso `pages/<Tela>/index.ts` com `export default` é obrigatório.

Layouts intermediários

`layout` insere um componente entre o shell e a página, para grupos de telas que compartilham sub-navegação (abas internas, cabeçalho de seção). O `AppRouter` agrupa rotas por layout automaticamente. Omita quando não houver.

`AppRouter.tsx`

```
// src/router/AppRouter.tsx
import React, { Suspense } from "react";import { Routes, Route, Navigate } from "react-router-dom";
import { useAuth } from "@context/AuthProvider";import ProtectedRoute from "@components/ProtectedRoute";
import LoadingScreen from "@components/LoadingScreen";import Main from "@components/layout/Main";
import { routes, NotFoundPage, type AppRoute } from "@/routes";
// agrupa por layout pra montar uma <Route> pai por layoutfunction groupByLayout(routeList: AppRoute[]) {  return routeList.reduce<Map<React.ComponentType | null, AppRoute[]>>((map, route) => {
    const key = route.layout ?? null;
    if (!map.has(key)) map.set(key, []);
    map.get(key)!.push(route);
    return map;
  }, new Map());
}
const AppRouter: React.FC = () => {
  const { isAuth } = useAuth();
```

```
  const authOnlyRoutes = routes.filter((r) => r.guard === "auth-only");  const publicRoutes = routes.filter((r) => r.guard === "public");  const protectedRoutes = routes.filter((r) => r.guard === "protected");
  const protectedByLayout = groupByLayout(protectedRoutes);
  return (
    <Suspense fallback={<LoadingScreen />}>
      <Routes>        <Route path="/" element={<Navigate to={isAuth ? "/dashboard" : "/login"} replace />} />
        {authOnlyRoutes.map(({ path, element: Element }) => (          <Route key={path} path={`/${path}`} element={isAuth ? <Navigate to="/dashboard" replace /> : <Element />} />
        ))}
        {publicRoutes.map(({ path, element: Element }) => (          <Route key={path} path={`/${path}`} element={<Element />} />
        ))}
        <Route
          element={
            <ProtectedRoute>
              <Main />
            </ProtectedRoute>
          }>          {[...protectedByLayout.entries()].map(([Layout, layoutRoutes]) => {
            const children = layoutRoutes.map(({ path, element: Element }) => (
<Route key={path} path={`/${path}`} element={<Element />} />
            ));
            if (!Layout) return <React.Fragment key="__root">{children}</React.Fragment>;
            return (              <Route key={Layout.displayName ?? Layout.name} element={<Layout />}>
                {children}
              </Route>
            );
          })}
        </Route>
        <Route path="*" element={<NotFoundPage />} />
      </Routes>
    </Suspense>
```

```
  );
};
export default AppRouter;
```

Pontos que não são acidentais:

- **Um** `<Suspense>` **na raiz** cobre todos os `lazy()`. Não envolva página por página.
- **Rotas protegidas ficam sob uma única** `<Route>` **pai** com `ProtectedRoute` + `Main` — o shell não remonta ao navegar entre telas protegidas.
- `path="*"` **por último** captura o 404.
- `/` **redireciona** conforme `isAuth`.

ProtectedRoute

```
// src/components/ProtectedRoute/ProtectedRoute.tsx
import React from "react";
import { Navigate, useLocation } from "react-router-dom";import { useAuth } from
"@context/AuthProvider";
import LoadingScreen from "@components/LoadingScreen";
const ProtectedRoute: React.FC<{ children: React.ReactNode }> = ({ children }) => {  const {
isAuth, isLoading } = useAuth();
  const location = useLocation();
  // enquanto reconecta por token, nao decidir ainda
  if (isLoading) return <LoadingScreen />;  if (!isAuth) return <Navigate to="/login" state={{
from: location }} replace />;
  return <>{children}</>;
};
export default ProtectedRoute;
```

O `isLoading` **é essencial.** Sem ele, um refresh de página redireciona para o login antes da reconexão por token terminar. Foi por isso que `App.tsx` também exibe `LoadingScreen` enquanto `isLoading` — ver 05.

`state={{ from: location }}` preserva o destino para redirecionar depois do login.

Adicionar uma rota

```
// 1. src/routes/paths.ts
CADASTRO: {
  FORNECEDOR: "cadastro/fornecedor",
  CLIENTE: "cadastro/cliente"          // novo
}
// 2. src/routes/routes.ts
{
  path: PATHS.CADASTRO.CLIENTE,
  element: lazy(() => import("@pages/Cadastro/Cliente")),
  guard: "protected"
}
// 3. src/routes/menuTree.ts
{ label: "Cliente", path: PATHS.CADASTRO.CLIENTE }
```

E crie `src/pages/Cadastro/Cliente/index.ts` com `export default`, senão o `lazy()` falha em runtime sem erro de compilação.

Rotas com parâmetro

O padrão declarativo suporta parâmetro na string:

```
{
  path: "cadastro/fornecedor/:oid",  element: lazy(() =>
import("@pages/Cadastro/Fornecedor/Detalhe")),
  guard: "protected"
}
```

Na página, `useParams()`. **Não coloque rota parametrizada no `menuTree`** — não há valor de parâmetro para navegar a partir do menu.

Com navegação por abas, rotas parametrizadas costumam ser dispensáveis: o contexto passa pelo `TabsContext` em vez de pela URL. Se o projeto novo dispensar abas, rotas parametrizadas voltam a ser o caminho natural.

Revision #2
Created Thu, Aug 20, 2026 4:57 PM by Geraldo Barbosa
Updated Tue, Aug 25, 2026 4:53 PM by Geraldo Barbosa