import {

  Authenticated,

  GitHubBanner,

  LegacyAuthProvider,

  Refine,

} from "@refinedev/core";

import { RefineKbar, RefineKbarProvider } from "@refinedev/kbar";

import {

  ErrorComponent,4

  // Layout,

  notificationProvider,

  RefineSnackbarProvider,

  ReadyPage,

} from "@refinedev/mui";

import { CssBaseline, GlobalStyles } from "@mui/material";

import routerBindings, {

  CatchAllNavigate,

  NavigateToResource,

  UnsavedChangesNotifier,

} from "@refinedev/react-router-v6";

import dataProvider from "@refinedev/simple-rest";

import axios, { AxiosRequestConfig } from "axios";

import { CredentialResponse } from "interfaces/google";

// import {

//   CategoryCreate,

//   CategoryEdit,

//   CategoryList,

//   CategoryShow,

// } from "pages/categories";

// import { Login } from "pages/login";

// import {

//   ProductCreate,

//   ProductEdit,

//   ProductList,

//   ProductShow,

// } from "pages/products";

import {

  PropertyCreate,

  PropertyEdit,

  PropertyList,

  PropertyShow,

} from "pages/properties";

import { BrowserRouter, Outlet, Route, Routes } from "react-router-dom";

import { parseJwt } from "utils/parse-jwt";

// import { Header } from "./components/header";

import { Title, Sider, Layout, Header } from "components/layout";

import {

  Login,

  Home,

  Agents,

  MyProfile,

  PropertyDetails,

  AllProperties,

  CreateProperty,

  AgentProfile,

  EditProperty,

} from "pages";

import { ColorModeContextProvider } from "./contexts/color-mode";

import {

  AccountCircleOutlined,

  ChatBubbleOutline,

  PeopleAltOutlined,

  StarOutlineRounded,

  VillaOutlined,

} from "@mui/icons-material";

const axiosInstance = axios.create();

axiosInstance.interceptors.request.use((request: AxiosRequestConfig) => {

  const token = localStorage.getItem("token");

  if (request.headers) {

    request.headers["Authorization"] = `Bearer ${token}`;

  } else {

    request.headers = {

      Authorization: `Bearer ${token}`,

    };

  }

  return request;

});

function App() {

  const authProvider: LegacyAuthProvider = {

    login: ({ credential }: CredentialResponse) => {

      const profileObj = credential ? parseJwt(credential) : null;

      if (profileObj) {

        localStorage.setItem(

          "user",

          JSON.stringify({

            ...profileObj,

            avatar: profileObj.picture,

          })

        );

      }

      localStorage.setItem("token", `${credential}`);

      return Promise.resolve();

    },

    logout: () => {

      const token = localStorage.getItem("token");

      if (token && typeof window !== "undefined") {

        localStorage.removeItem("token");

        localStorage.removeItem("user");

        axios.defaults.headers.common = {};

        window.google?.accounts.id.revoke(token, () => {

          return Promise.resolve();

        });

      }

      return Promise.resolve();

    },

    checkError: () => Promise.resolve(),

    checkAuth: async () => {

      const token = localStorage.getItem("token");

      if (token) {

        return Promise.resolve();

      }

      return Promise.reject();

    },

    getPermissions: () => Promise.resolve(),

    getUserIdentity: async () => {

      const user = localStorage.getItem("user");

      if (user) {

        return Promise.resolve(JSON.parse(user));

      }

    },

  };

  return (

    <BrowserRouter>

      {/\* <GitHubBanner /> \*/}

      <RefineKbarProvider>

        <ColorModeContextProvider>

          <CssBaseline />

          <GlobalStyles styles={{ html: { WebkitFontSmoothing: "auto" } }} />

          <RefineSnackbarProvider>

            <Refine

              dataProvider={dataProvider("https://api.fake-rest.refine.dev")}

              notificationProvider={notificationProvider}

              // resources={[

              //   {

              //     name: "products",

              //     list:  "/products",

              //     create: "/products/create",

              //     edit: "/products/edit/:id",

              //     show: "/products/show/:id",

              //     canDelete: true,

              //   },

              //   {

              //     name: "categories",

              //     list: "/categories",

              //     create: "/categories/create",

              //     edit: "/categories/edit/:id",

              //     show: "/categories/show/:id",

              //     canDelete: true,

              //   },

              // ]}

              resources={[

                {

                    name: "properties",

                    list: AllProperties,

                    show: PropertyDetails,

                    create: CreateProperty,

                    edit: EditProperty,

                    icon: <VillaOutlined />,

                },

                {

                    name: "agents",

                    list: Agents,

                    show: AgentProfile,

                    icon: <PeopleAltOutlined />,

                },

                {

                    name: "reviews",

                    list: Home,

                    icon: <StarOutlineRounded />,

                },

                {

                    name: "messages",

                    list: Home,

                    icon: <ChatBubbleOutline />,

                },

                {

                    name: "my-profile",

                    options: { label: "My Profile " },

                    list: MyProfile,

                    icon: <AccountCircleOutlined />,

                },

            ]}

              routerProvider={routerBindings}

              legacyAuthProvider={authProvider}

              options={{

                syncWithLocation: true,

                warnWhenUnsavedChanges: true,

              }}

            >

              <Routes>

                <Route

                  element={

                    <Authenticated fallback={<CatchAllNavigate to="/login" />}>

                      <Layout Header={Header}>

                        <Outlet />

                      </Layout>

                    </Authenticated>

                  }

                >

                  {/\* <Route

                    index

                    element={<NavigateToResource resource="products" />}

                  /> \*/}

                  {/\* <Route path="/products">

                    <Route index element={<ProductList />} />

                    <Route path="create" element={<ProductCreate />} />

                    <Route path="edit/:id" element={<ProductEdit />} />

                    <Route path="show/:id" element={<ProductShow />} />

                  </Route>

                  <Route path="/categories">

                    <Route index element={<CategoryList />} />

                    <Route path="create" element={<CategoryCreate />} />

                    <Route path="edit/:id" element={<CategoryEdit />} />

                    <Route path="show/:id" element={<CategoryShow />} />

                  </Route> \*/}

                  <Route path="/properties"> \*/

                    <Route index element={<PropertyList />} />

                    <Route path="create" element={<PropertyCreate />} />

                    <Route path="edit/:id" element={<PropertyEdit />} />

                    <Route path="show/:id" element={<PropertyShow />} />

                  </Route>

                </Route>

                <Route

                  element={

                    <Authenticated fallback={<Outlet />}>

                      <NavigateToResource />

                    </Authenticated>

                  }

                >

                  <Route path="/login" element={<Login />} />

                </Route>

                <Route

                  element={

                    <Authenticated>

                      <Layout Header={Header}>

                        <Outlet />

                      </Layout>

                    </Authenticated>

                  }

                >

                  <Route path="\*" element={<ErrorComponent />} />

                </Route>

              </Routes>

              <RefineKbar />

              <UnsavedChangesNotifier />

            </Refine>

           </RefineSnackbarProvider>

         </ColorModeContextProvider>

       </RefineKbarProvider>

     </BrowserRouter>

   );

}

export default App;

somehow this version stopped showing the sidebar after first use:

import React from "react";

import {

    GitHubBanner,

    Refine,

    LegacyAuthProvider as AuthProvider,

} from "@refinedev/core";

import {

    notificationProvider,

    RefineSnackbarProvider,

    ReadyPage,

    ErrorComponent,

} from "@refinedev/mui";

import { CssBaseline, GlobalStyles } from "@mui/material";

import {

    AccountCircleOutlined,

    ChatBubbleOutline,

    PeopleAltOutlined,

    StarOutlineRounded,

    VillaOutlined,

} from "@mui/icons-material";

import dataProvider from "@refinedev/simple-rest";

import routerProvider from "@refinedev/react-router-v6/legacy";

import routerBindings, {

  CatchAllNavigate,

  NavigateToResource,

  UnsavedChangesNotifier,

} from "@refinedev/react-router-v6";

import axios, { AxiosRequestConfig } from "axios";

import { Title, Sider, Layout, Header } from "components/layout";

import { ColorModeContextProvider } from "./contexts/color-mode";

import { CredentialResponse } from "interfaces/google";

import { parseJwt } from "utils/parse-jwt";

import {

    Login,

    Home,

    Agents,

    MyProfile,

    PropertyDetails,

    AllProperties,

    CreateProperty,

    AgentProfile,

    EditProperty,

} from "pages";

const axiosInstance = axios.create();

axiosInstance.interceptors.request.use((request: AxiosRequestConfig) => {

    const token = localStorage.getItem("token");

    if (request.headers) {

        request.headers["Authorization"] = `Bearer ${token}`;

    } else {

        request.headers = {

            Authorization: `Bearer ${token}`,

        };

    }

    return request;

});

function App() {

    const authProvider: AuthProvider = {

        login: async ({ credential }: CredentialResponse) => {

            const profileObj = credential ? parseJwt(credential) : null;

            if (profileObj) {

                const response = await fetch(

                    "http://localhost:8080/api/v1/users",

                    {

                        method: "POST",

                        headers: { "Content-Type": "application/json" },

                        body: JSON.stringify({

                            name: profileObj.name,

                            email: profileObj.email,

                            avatar: profileObj.picture,

                        }),

                    },

                );

                const data = await response.json();

                if (response.status === 200) {

                    localStorage.setItem(

                        "user",

                        JSON.stringify({

                            ...profileObj,

                            avatar: profileObj.picture,

                            userid: data.\_id,

                        }),

                    );

                } else {

                    return Promise.reject();

                }

            }

            localStorage.setItem("token", `${credential}`);

            return Promise.resolve();

        },

        logout: () => {

            const token = localStorage.getItem("token");

            if (token && typeof window !== "undefined") {

                localStorage.removeItem("token");

                localStorage.removeItem("user");

                axios.defaults.headers.common = {};

                window.google?.accounts.id.revoke(token, () => {

                    return Promise.resolve();

                });

            }

            return Promise.resolve();

        },

        checkError: () => Promise.resolve(),

        checkAuth: async () => {

            const token = localStorage.getItem("token");

            if (token) {

                return Promise.resolve();

            }

            return Promise.reject();

        },

        getPermissions: async () => null,

        getUserIdentity: async () => {

            const user = localStorage.getItem("user");

            if (user) {

                return Promise.resolve(JSON.parse(user));

            }

        },

    };

    return (

        <ColorModeContextProvider>

            <GitHubBanner />

            <CssBaseline />

            <GlobalStyles styles={{ html: { WebkitFontSmoothing: "auto" } }} />

            <RefineSnackbarProvider>

                <Refine

                    dataProvider={dataProvider("http://localhost:8080/api/v1")}

                    notificationProvider={notificationProvider}

                    ReadyPage={ReadyPage}

                    catchAll={<ErrorComponent />}

                    resources={[

                        {

                            name: "properties",

                            list: AllProperties,

                            show: PropertyDetails,

                            create: CreateProperty,

                            edit: EditProperty,

                            icon: <VillaOutlined />,

                        },

                        {

                            name: "agents",

                            list: Agents,

                            show: AgentProfile,

                            icon: <PeopleAltOutlined />,

                        },

                        {

                            name: "reviews",

                            list: Home,

                            icon: <StarOutlineRounded />,

                        },

                        {

                            name: "messages",

                            list: Home,

                            icon: <ChatBubbleOutline />,

                        },

                        {

                            name: "my-profile",

                            options: { label: "My Profile " },

                            list: MyProfile,

                            icon: <AccountCircleOutlined />,

                        },

                    ]}

                    Title={Title}

                    Sider={Sider}

                    Layout={Layout}

                    Header={Header}

                    routerProvider={routerBindings}

                    legacyAuthProvider={authProvider}

                    LoginPage={Login}

                    DashboardPage={Home}

                />

            </RefineSnackbarProvider>

        </ColorModeContextProvider>

    );

}

export default App;