**App.tsx**

Este código define el componente principal App en React, el cual organiza la estructura de una aplicación que maneja autenticación de usuarios, detección de tamaño de ventana y navegación de rutas.

|  |  |
| --- | --- |
| import { useEffect, useState } from 'react';  import { Routes, Route, useLocation } from 'react-router-dom';  import { ToastContainer } from 'react-toastify';  import { useWindowSize } from '@app/hooks/useWindowSize';  import { calculateWindowSize } from '@app/utils/helpers';  import { setWindowSize } from '@app/store/reducers/ui';  import ReactGA from 'react-ga4';  import Dashboard from '@pages/Dashboard';  import Blank from '@pages/Blank';  import SubMenu from '@pages/SubMenu';  import Profile from '@pages/profile/Profile';  import PublicRoute from './routes/PublicRoute';  import PrivateRoute from './routes/PrivateRoute';  import { setCurrentUser } from './store/reducers/auth';  import { firebaseAuth } from './firebase';  import { onAuthStateChanged } from 'firebase/auth';  import { useAppDispatch, useAppSelector } from './store/store';  import { Loading } from './components/Loading';  import Geoportal from '@pages/geoportal/Geoportal'; | ****Importación de Módulos****  **useEffect** y **useState**: Son hooks que permiten manejar el ciclo de vida de los componentes y el estado local, respectivamente.   **Routes** y **Route**: Son parte de react-router-dom, que permite definir y manejar la navegación entre rutas en la aplicación.   **useLocation**: Es un hook que devuelve la ubicación actual del navegador (ruta actual).  **ToastContainer**: De la biblioteca react-toastify, permite la visualización de notificaciones tipo toast. |
| import Main from '@modules/main/Main';  import Login from '@modules/login/Login';  import Register from '@modules/register/Register';  import ForgetPassword from '@modules/forgot-password/ForgotPassword';  import RecoverPassword from '@modules/recover-password/RecoverPassword'; | **Importación de Componentes** Se importan varios componentes que representan diferentes páginas o vistas en la aplicación:   * **Main**: Componente principal que contendrá otras rutas protegidas. * **Login, Register, ForgetPassword, RecoverPassword**: Componentes para manejar el inicio de sesión, registro y recuperación de contraseñas. |
|  |  |
| const { VITE\_NODE\_ENV } = import.meta.env; |  |
| const App = () => {    const windowSize = useWindowSize();    const screenSize = useAppSelector((state) => state.ui.screenSize);    const dispatch = useAppDispatch();    const location = useLocation();    const [isAppLoading, setIsAppLoading] = useState(true);    useEffect(() => {      onAuthStateChanged(        firebaseAuth,        (user) => {          if (user) {            dispatch(setCurrentUser(user));          } else {            dispatch(setCurrentUser(null));          }          setIsAppLoading(false);        },        (e) => {          console.log(e);          dispatch(setCurrentUser(null));          setIsAppLoading(false);        }      );    }, []);    useEffect(() => {      const size = calculateWindowSize(windowSize.width);      if (screenSize !== size) {        dispatch(setWindowSize(size));      }    }, [windowSize]);    useEffect(() => {      if (location && location.pathname && VITE\_NODE\_ENV === 'production') {        ReactGA.send({          hitType: 'pageview',          page: location.pathname,        });      }    }, [location]);    if (isAppLoading) {      return <Loading />;    } | **IMPORTACIÓN DE HOOKS Y FUNCIONES**   1. **useWindowSize:** Se utiliza para obtener el tamaño actual de la ventana del navegador. 2. **useAppSelector y useAppDispatch:** Son parte del patrón de manejo de estado de Redux.    * **useAppSelector:** Selecciona un valor del estado global.    * **useAppDispatch:** Envia acciones para actualizar el estado global. 3. **useLocation:** Es un hook de React Router que devuelve el objeto de ubicación actual, lo que permite saber en qué ruta estás. 4. **useState y useEffect:** Hooks fundamentales en React:    * **useState: Maneja el estado local del componente.**    * **useEffect: Ejecuta efectos secundarios (código que ocurre en respuesta a cambios en el estado o las propiedades del componente**).   **ESTADO LOCAL**   1. **isAppLoading:** Este estado local se usa para controlar si la aplicación está cargando. Comienza como true y se cambiará a false una vez que se verifique si el usuario está autenticado.   **PRIMER USEEFFECT: VERIFICACIÓN DE AUTENTICACIÓN** Este useEffect verifica el estado de autenticación del usuario utilizando onAuthStateChanged (probablemente de *Firebase Authentication*):   * Si el usuario está autenticado, despacha la acción setCurrentUser con la información del usuario. * Si no hay un usuario autenticado o ocurre un error, se despacha setCurrentUser(null) para eliminar cualquier usuario en el estado. * Se establece setIsAppLoading(false) para indicar que la aplicación ha terminado de cargar después de comprobar el estado de autenticación.   **Segundo useEffect: Detección del Tamaño de la Ventana**  Este useEffect escucha cambios en el tamaño de la ventana (a través de windowSize.width):   * Usa calculateWindowSize (una función externa no mostrada aquí) para obtener el tamaño de la pantalla. * Si el tamaño detectado es diferente del estado global (screenSize), se despacha la acción setWindowSize para actualizar el estado global con el nuevo tamaño.   **Tercer useEffect: Seguimiento de Páginas**  Este useEffect se ejecuta cuando cambia la ubicación (ruta) de la aplicación:   * Si la aplicación está en un entorno de producción (VITE\_NODE\_ENV === 'production'), se envía un evento de visualización de página a Google Analytics utilizando ReactGA.send.   **Comportamiento de Carga**   * **if (isAppLoading)**: Si la aplicación todavía está cargando (por ejemplo, está verificando el estado de autenticación), muestra el componente <Loading />. Este componente probablemente renderiza una pantalla de carga mientras se espera el resultado de la autenticación. |
| return (      <>        <Routes>          <Route path="/login" element={<PublicRoute />}>            <Route path="/login" element={<Login />} />          </Route>          <Route path="/register" element={<PublicRoute />}>            <Route path="/register" element={<Register />} />          </Route>          <Route path="/forgot-password" element={<PublicRoute />}>            <Route path="/forgot-password" element={<ForgetPassword />} />          </Route>          <Route path="/recover-password" element={<PublicRoute />}>            <Route path="/recover-password" element={<RecoverPassword />} />          </Route>          <Route path="/" element={<PrivateRoute />}>            <Route path="/" element={<Main />}>              <Route path="/sub-menu-2" element={<Blank />} />              <Route path="/sub-menu-1" element={<SubMenu />} />              <Route path="/blank" element={<Blank />} />              <Route path="/profile" element={<Profile />} />              <Route path="/" element={<Dashboard />} />            </Route>          </Route>          <Route path="/geoportal" element={<PublicRoute />}>            <Route path="/geoportal" element={<Geoportal />} />          </Route>        </Routes> | Rutas La aplicación utiliza React Router para manejar la navegación:   * **Rutas públicas** (sin necesidad de autenticación):   + /login: Muestra el componente de inicio de sesión (<Login />).   + /register: Muestra el componente de registro (<Register />).   + /forgot-password y /recover-password: Muestran los componentes para la recuperación de contraseña.   + /geoportal: Muestra el componente del geoportal. * **Rutas privadas** (requieren autenticación):   + La ruta raíz / utiliza un componente <PrivateRoute /> que verifica si el usuario está autenticado.   + Dentro de la ruta privada, se muestra el componente <Main /> que tiene subrutas para el perfil, dashboard y otras páginas del menú. |
| <ToastContainer          autoClose={3000}          draggable={false}          position="top-right"          hideProgressBar={false}          newestOnTop          closeOnClick          rtl={false}          pauseOnHover        />      </>    );  }; | ToastContainer Este componente se encarga de mostrar las notificaciones en la aplicación. Aquí se configura de manera similar al ejemplo anterior:   * Cierra automáticamente las notificaciones después de 3000 ms. * Las notificaciones se muestran en la esquina superior derecha. * Las más nuevas aparecen primero. * El usuario puede cerrarlas con un clic. * Pausa la notificación cuando se pasa el cursor por encima. |