Тестовое задание

1. Запуск проекта, подключения стора

Установим библиотеки:

**axios** – для запросов на сервер

**react-router-dom, @types/react-router-dom** – для постраничной навигации

**moment** – для работы с датами https://momentjs.com/ библиотеки antd

**redux, react-redux, @types/reac-redux** – управление состоянием

**redux-thunk** – для асинхронных запросов

Итак,

Оборачиваем приложение в **<Provider>,** чтобы связать приложение React с Redux

Затем оборачиваем приложение в **<BrowserRouter>**, чтобы в дальнейшем работать с постраничной навигацией

ReactDOM.render(  
 <Provider store={}>  
 <BrowserRouter>  
 <App/>  
 </BrowserRouter>  
 </Provider>  
 ,  
 *document*.getElementById('root')  
);

Также определим две страницы: Pages и Events. Соответственно создадим папку **pages** и в ней два файла Pages.tsx и Events.tsx

Далее создадим папку с компонентами **components**.

Далее создадим папку с компонентами **store**. В ней будет написан redux

Далее создадим папку с компонентами **models**. В ней будут описаны типы сущностей проекта.

Итого мы создали изначальные файлы и папки, которые понадобятся для дальнейшей работы.

1. Постраничная навигация

**- Описываем роутер**

Создадим папку **router** в которой будет находиться файл index.ts. В нем будем **описывать** все маршруты, используемые в приложении.

В файле будем использовать enum. Он будет словарем, в котором мы опишем пути и сможем их потом быстро поменять не переписывая все приложение:

import Login from "../pages/Login";  
import Event from "../pages/Event";  
  
export interface IRoute {  
 path: string;  
 component: React.ComponentType;  
 exact?: boolean;  
}  
  
// словарь, в котором будут храниться маршруты к компонентам  
export enum RouteNames {  
 *LOGIN* = '/login',  
 *EVENT* = '/',  
}  
  
// этот массив роутов будет показываться НЕавторизованным пользователям, т.к. есть логинизация  
export const publicRoutes: IRoute[] = [  
 {path: RouteNames.*LOGIN*, exact: true, component: Login}  
];  
  
// этот массив роутов будет показываться авторизованным пользователям, т.к. есть логинизация  
export const privateRoutes: IRoute[] = [  
 {path: RouteNames.*EVENT*, exact: true, component: Event}  
];

**- Физически запиливаем роутер**

Создадим в компонентах файл AppRouter.

Здесь будем использовать Redirect и условие по авторизации.

import React from 'react';  
import {Switch, Route, Redirect} from "react-router-dom";  
import {privateRoutes, publicRoutes, RouteNames} from "../router";  
  
const AppRouter = () => {  
 const auth = false; // если true, попадаем на events; если false, попадаем на login  
 return (  
 auth ? // проверка авторизации  
 <Switch>  
 {privateRoutes.map(*route* =>  
 <Route path={*route*.path}  
 exact={*route*.exact}  
 component={*route*.component}  
 key={*route*.path} // укажем в качестве ключа путь к компоненту, т.к. он будет уникальным  
 />  
 )}  
 <Redirect to={RouteNames.*EVENT*}/>  
 </Switch>  
 :  
 <Switch>  
 {publicRoutes.map(*route* =>  
 <Route path={*route*.path}  
 exact={*route*.exact}  
 component={*route*.component}  
 key={*route*.path} // укажем в качестве ключа путь к компоненту, т.к. он будет уникальным  
 />  
 )}  
 <Redirect to={RouteNames.*LOGIN*}/>  
 </Switch>  
 );  
};  
  
export default AppRouter;

Далее подключим роутер в приложение в App.tsx

import React, {FC} from 'react';  
import './App.css';  
import AppRouter from "./components/AppRouter";  
  
const App: FC = () => {  
 return (  
 <div>  
 <AppRouter/>  
 </div>  
 );  
};  
  
export default App;