

4-前端万能模板开发教程

初始化

本前端初始模板使用 Ant Design Pro。

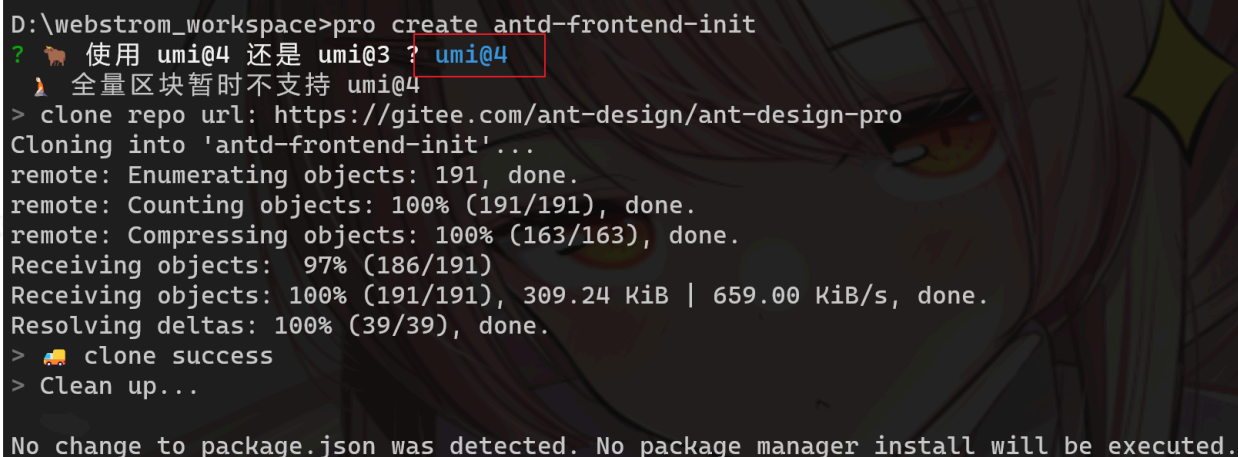
官方地址：开箱即用的中台前端/设计解决方案 - Ant Design Pro <<https://pro.ant.design/zh-CN/>>

Vue 版本官方地址：Ant Design Pro of Vue (antdv.com) <<https://pro.antdv.com/>>

初始化命令：

```
1 npm i @ant-design/pro-cli -g
2 pro create yuzi-generator-web-frontend
```

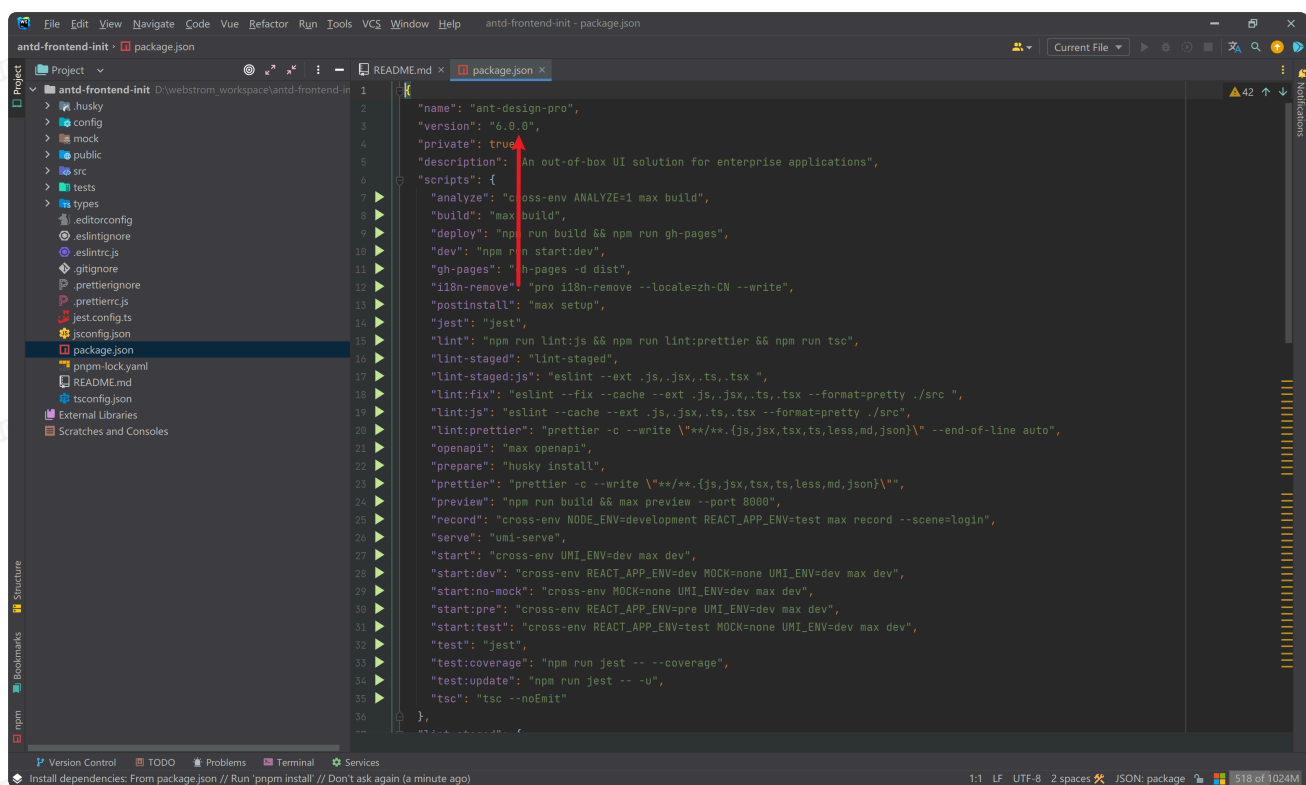
注意：一定要选择 umi@4!



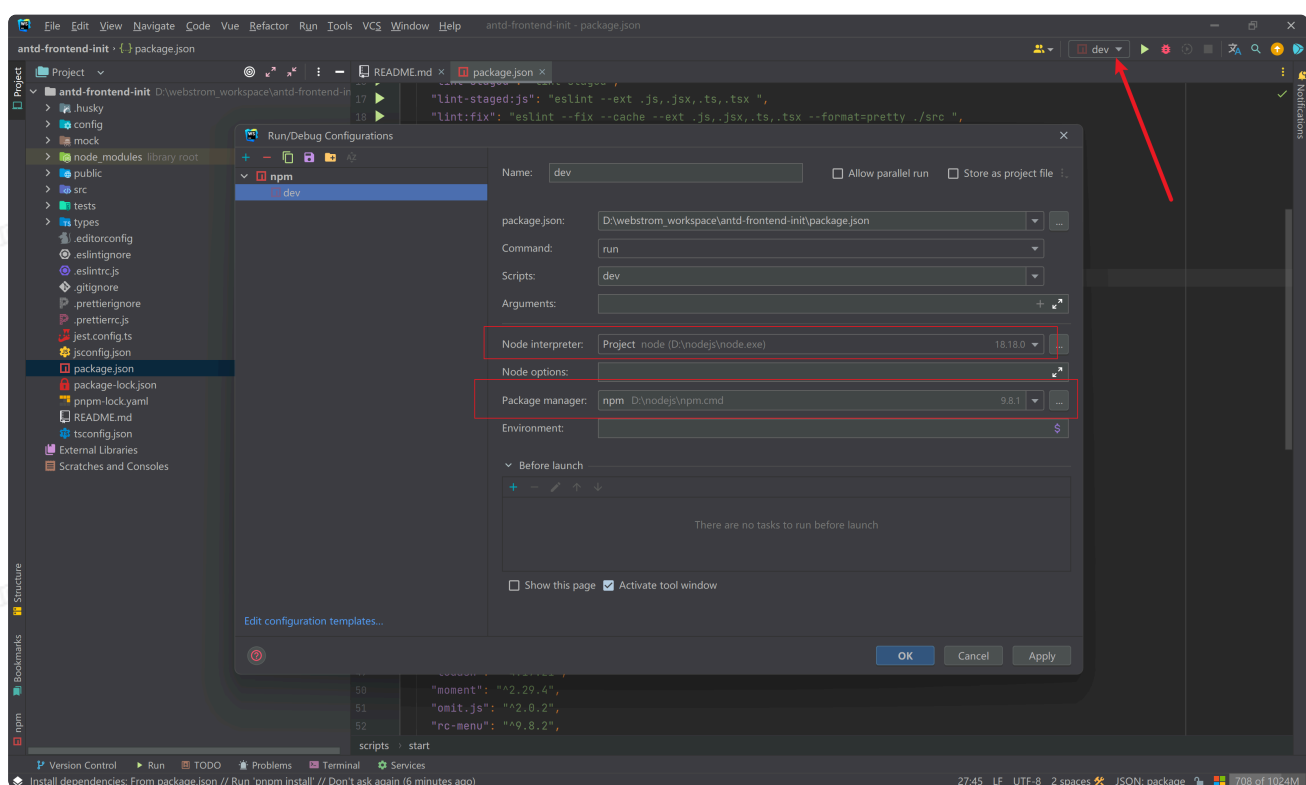
```
D:\webstrom_workspace>pro create antd-frontend-init
? 🐾 使用 umi@4 还是 umi@3 ? umi@4
  🚧 全量区块暂时不支持 umi@4
> clone repo url: https://gitee.com/ant-design/ant-design-pro
Cloning into 'antd-frontend-init'...
remote: Enumerating objects: 191, done.
remote: Counting objects: 100% (191/191), done.
remote: Compressing objects: 100% (163/163), done.
Receiving objects: 97% (186/191)
Receiving objects: 100% (191/191), 309.24 KiB | 659.00 KiB/s, done.
Resolving deltas: 100% (39/39), done.
> 🚚 clone success
> Clean up...

No change to package.json was detected. No package manager install will be executed.
```

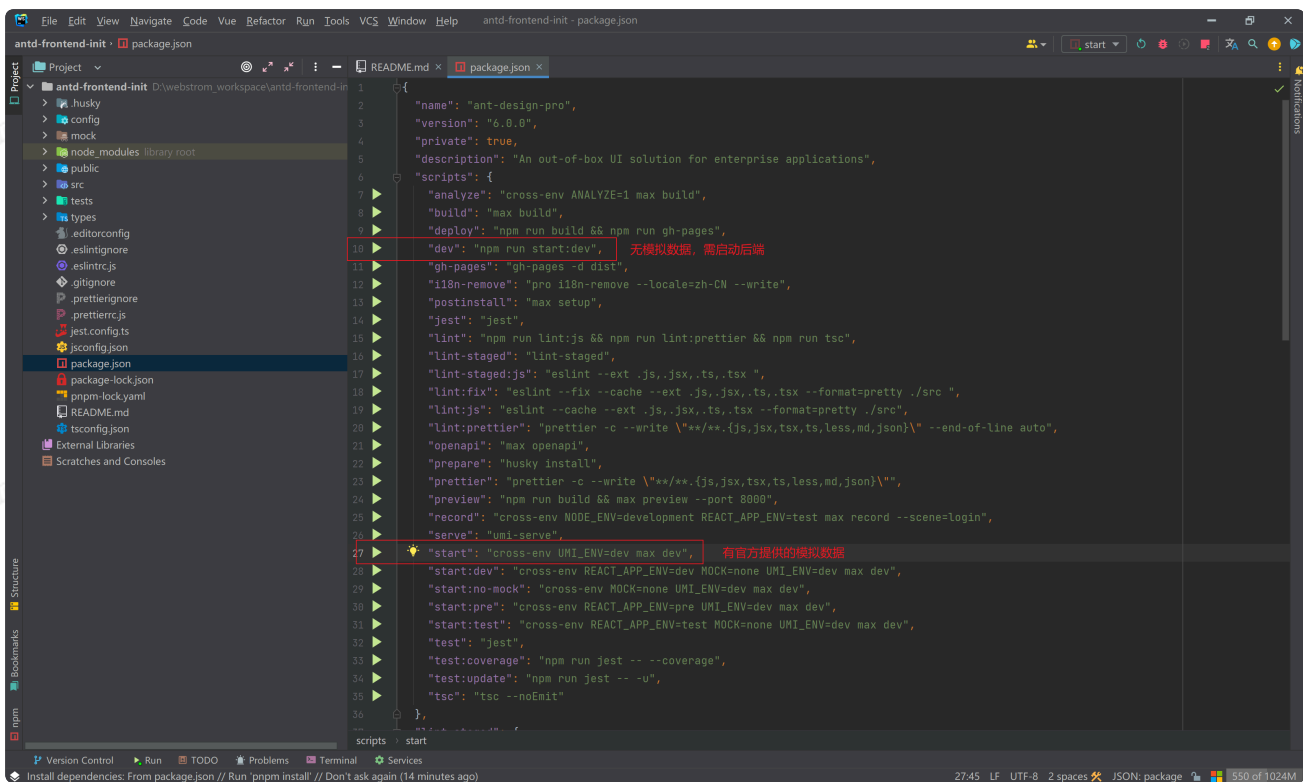
项目版本最好一致：6.0.0



可以使用 npm、yarn、pnpm、cnpm 下载依赖，建议版本不要过低，也就是 node 版本最好在 16 及以上。



运行 npm dev 测试执行，运行 npm start 可以开启 Mock 数据。



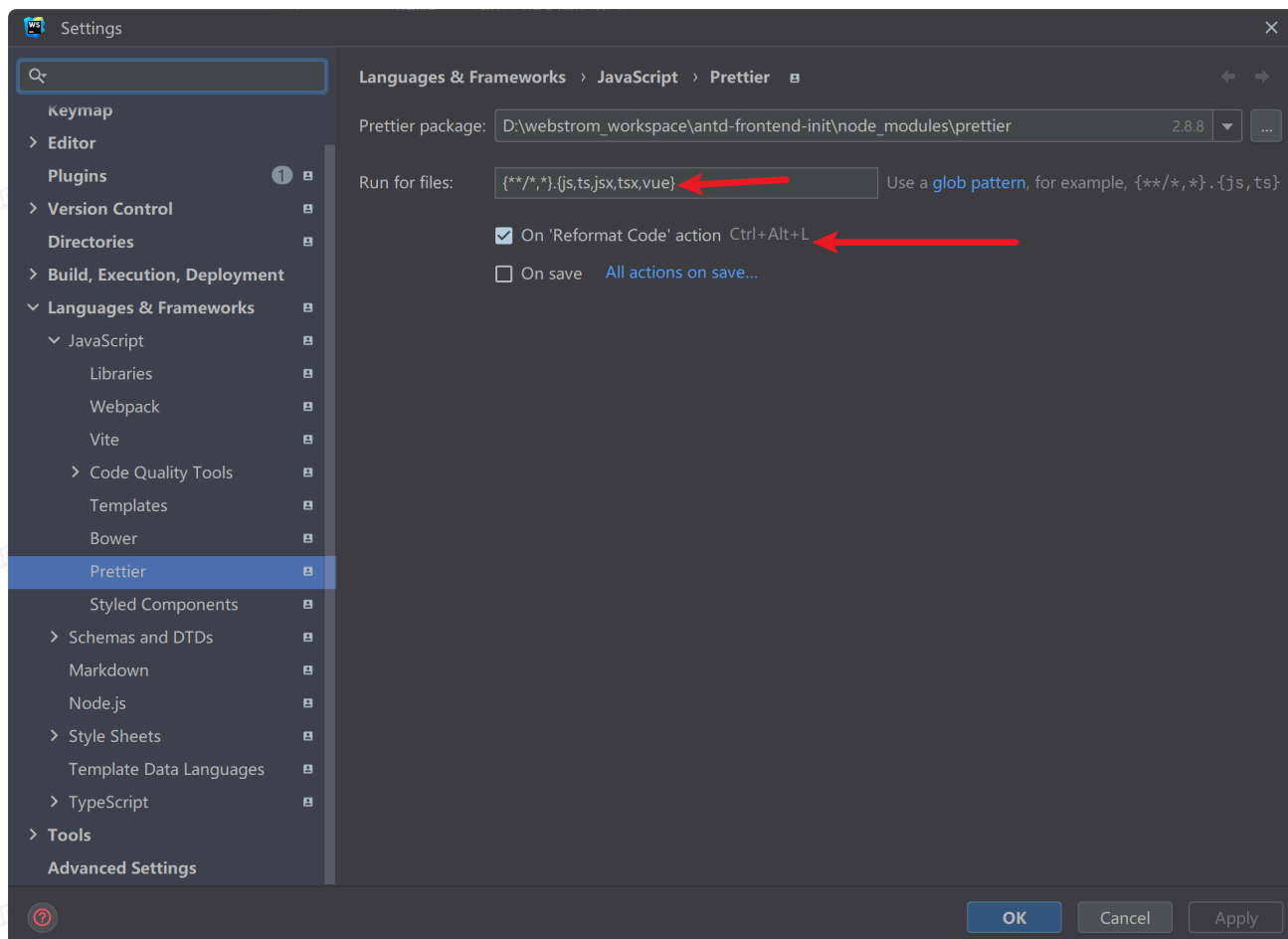
初始化可能遇到的问题

1) 可能会因为缺少 .git 文件导致 husky 执行报错，忽略即可。

开发规范

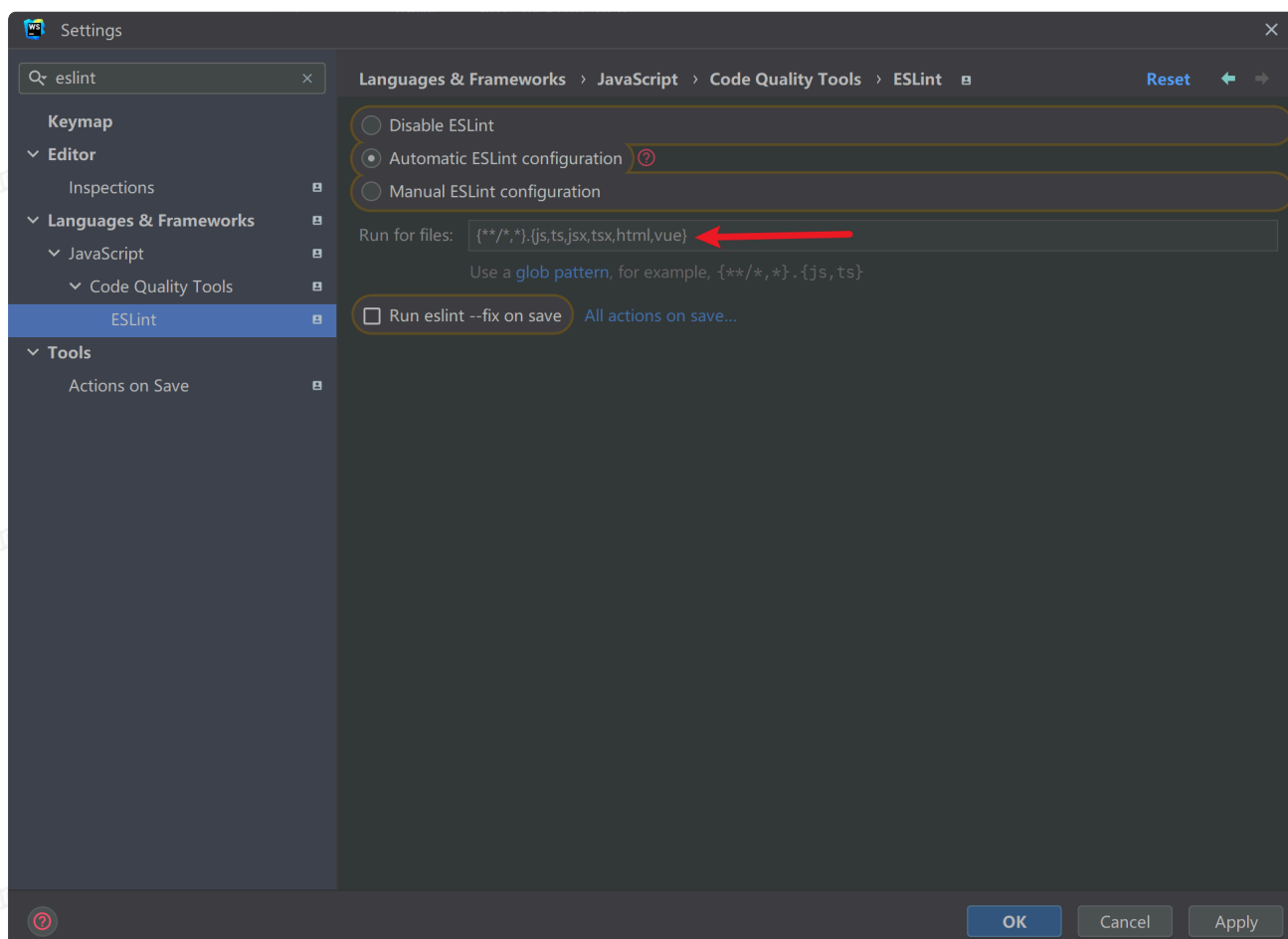
Prettier 格式化工具，用来统一代码格式

格式化快捷键建议勾选，Vue 技术栈同学建议手动加 .Vue 后缀



Eslint 保持代码风格，减少代码出错。

同样的，Vue 技术栈手动加上即可



模板瘦身

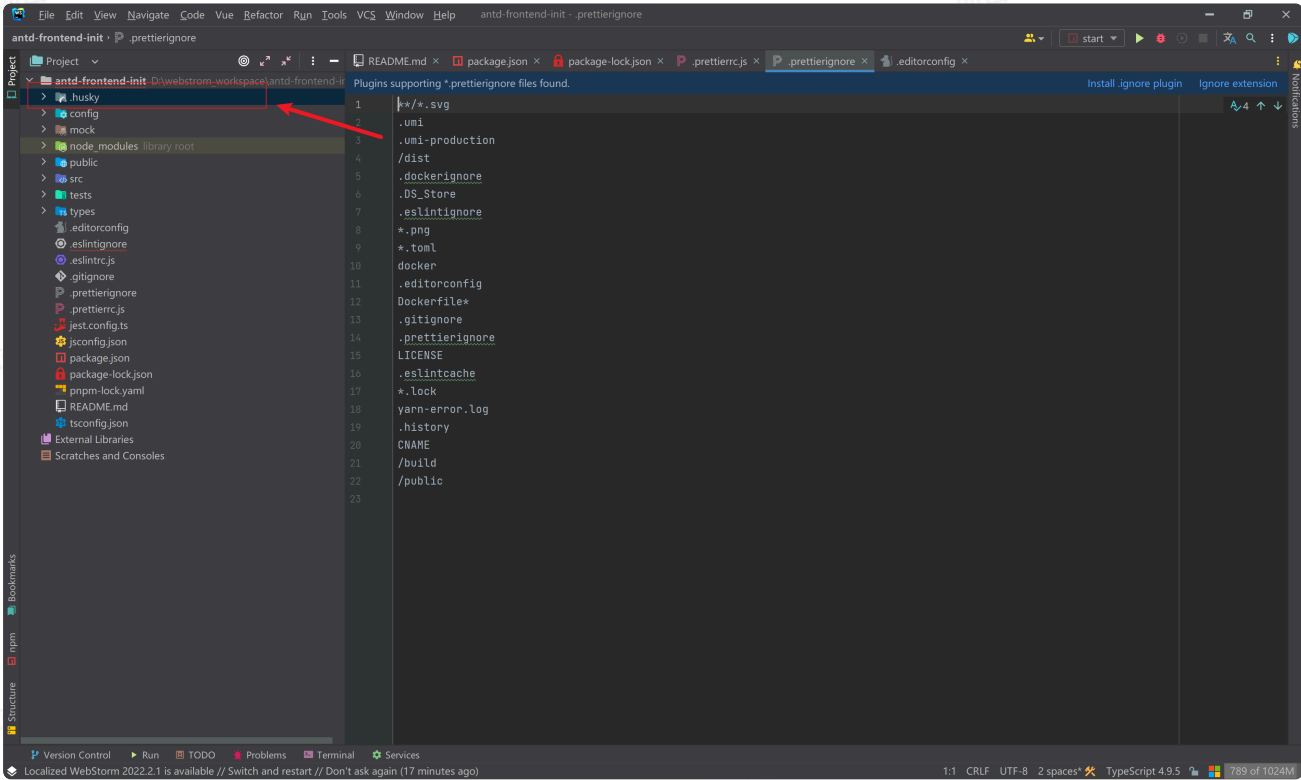
移除模块

注意事项（重点）

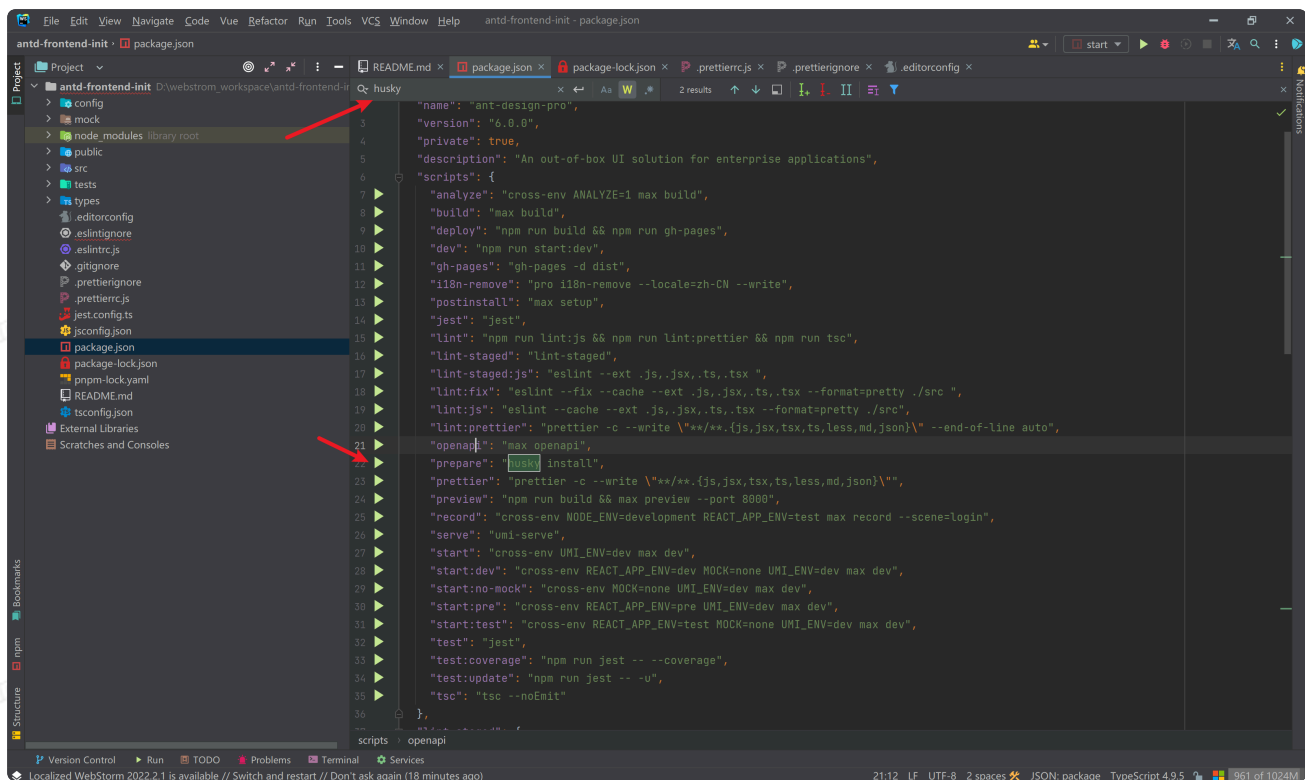
一次移除后，去重启一下项目，看看能否正常运行，控制变量法，如果不行，直接回滚一下就可以了，一次性删除太多，容易找不到源头。

移除 husky

一个用来提交前检查代码的规范，保证代码的一致性，一般用于团体协作，个人没有必要。

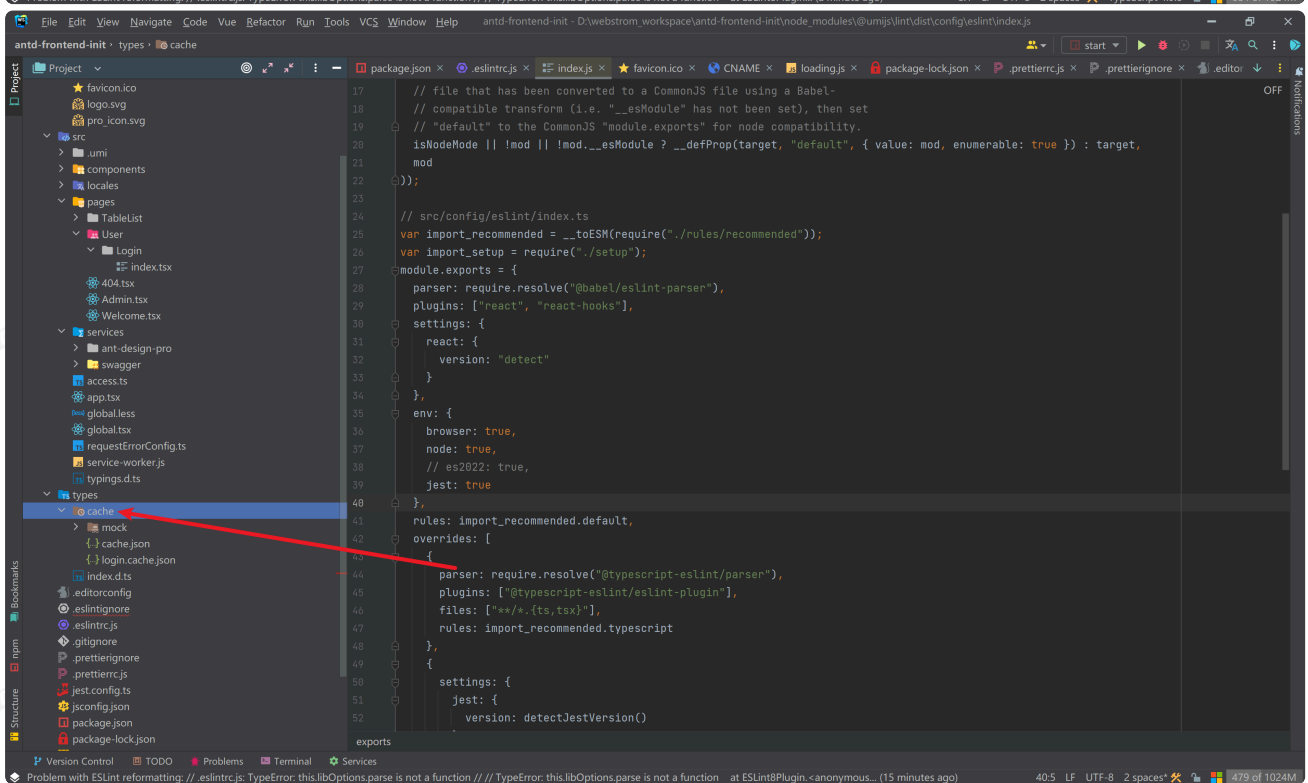
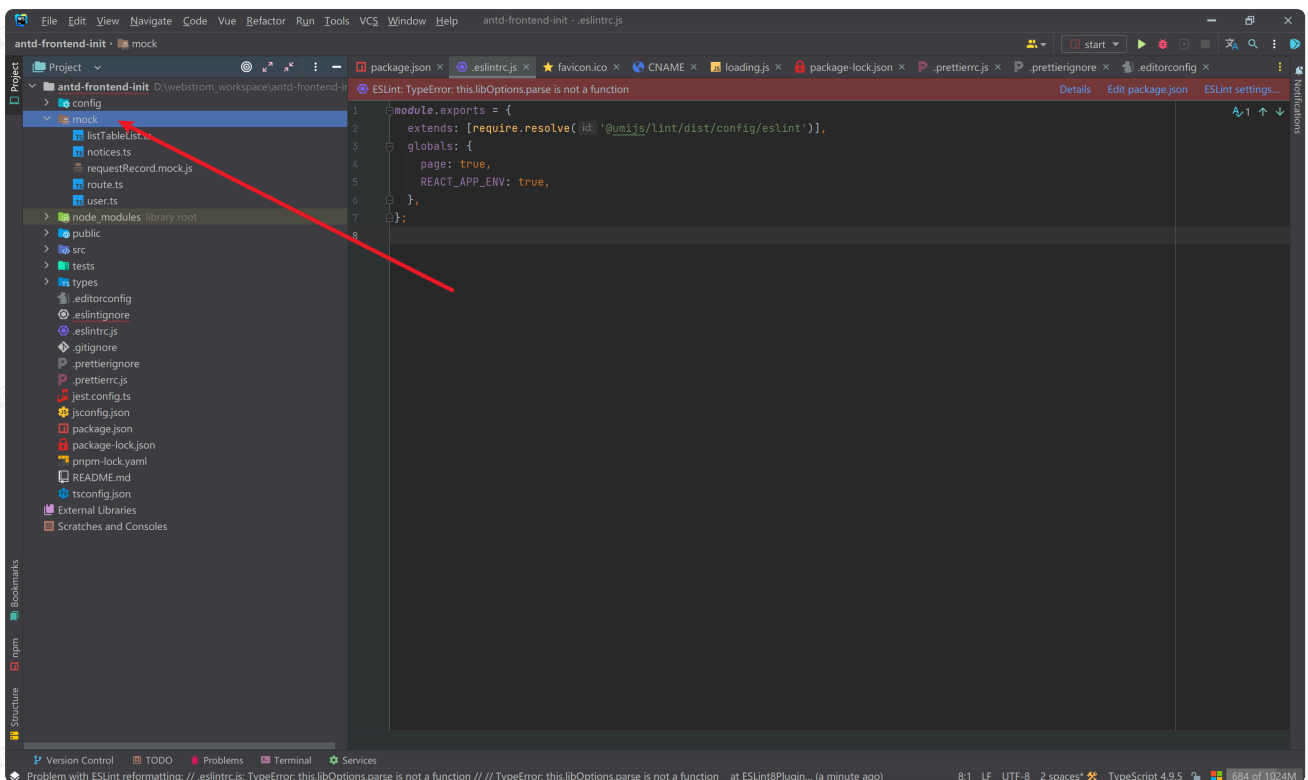


移除相关的命令



移除 mock

mock 是官方提供的模拟数据，而我们自己是有真实的后端接口要对接的，因此移除。

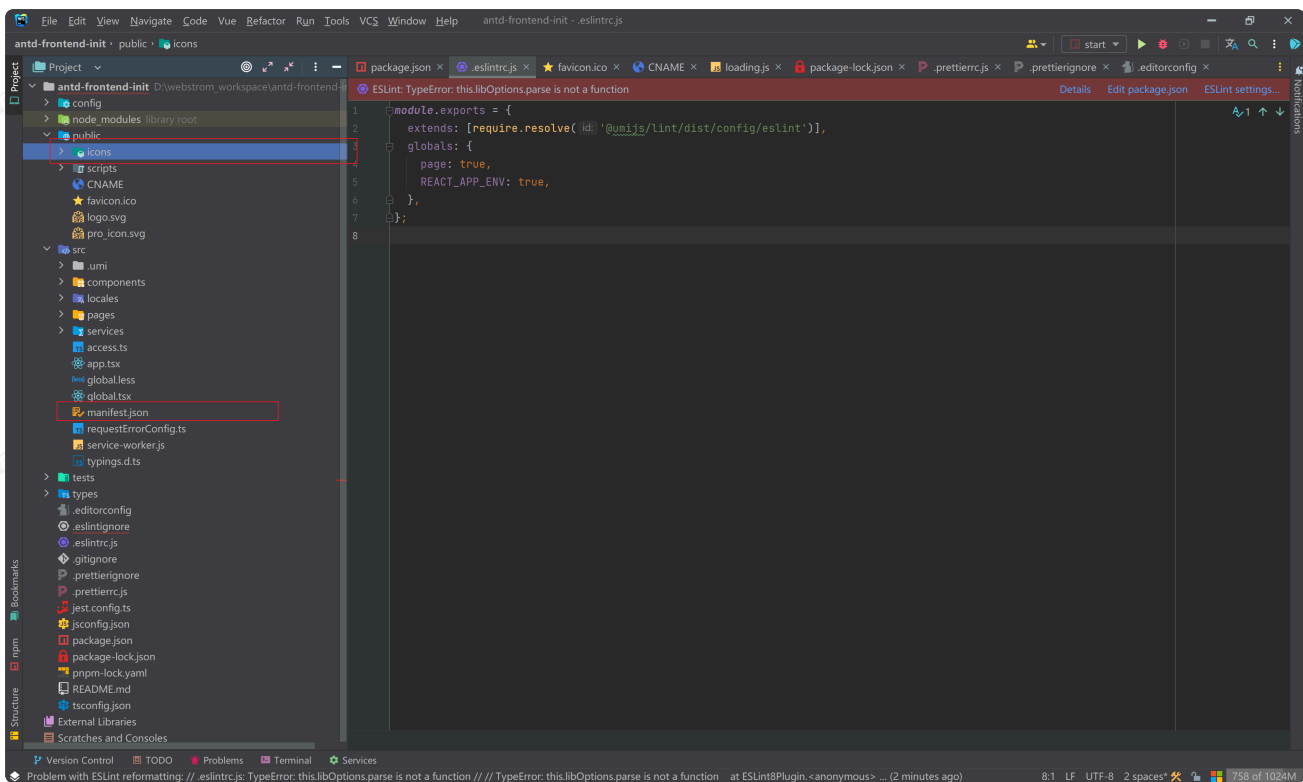


移除 icons 和 manifest.json

图标和适配移动端所需要的 Json，直接删除即可。

项目-更新

项目-更新

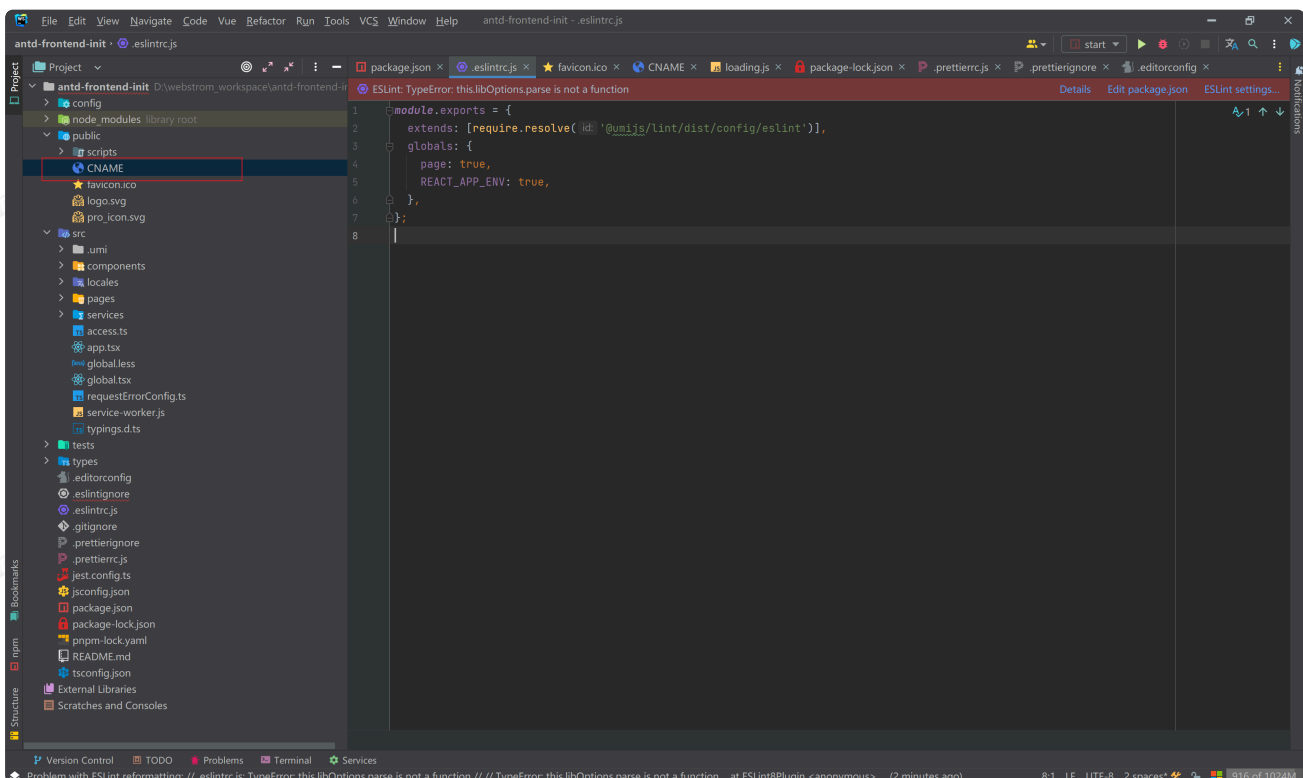


项目-更新

项目-更新

移除 cname

域名映射，官方提供的，和我们自己的域名无关。



移除国际化

一般我们自己的上线的项目只对本国用户去开放，而且访问人数也较少，没有必要去用国际化，会增加打包体积，而且页面加载速度也会变慢。

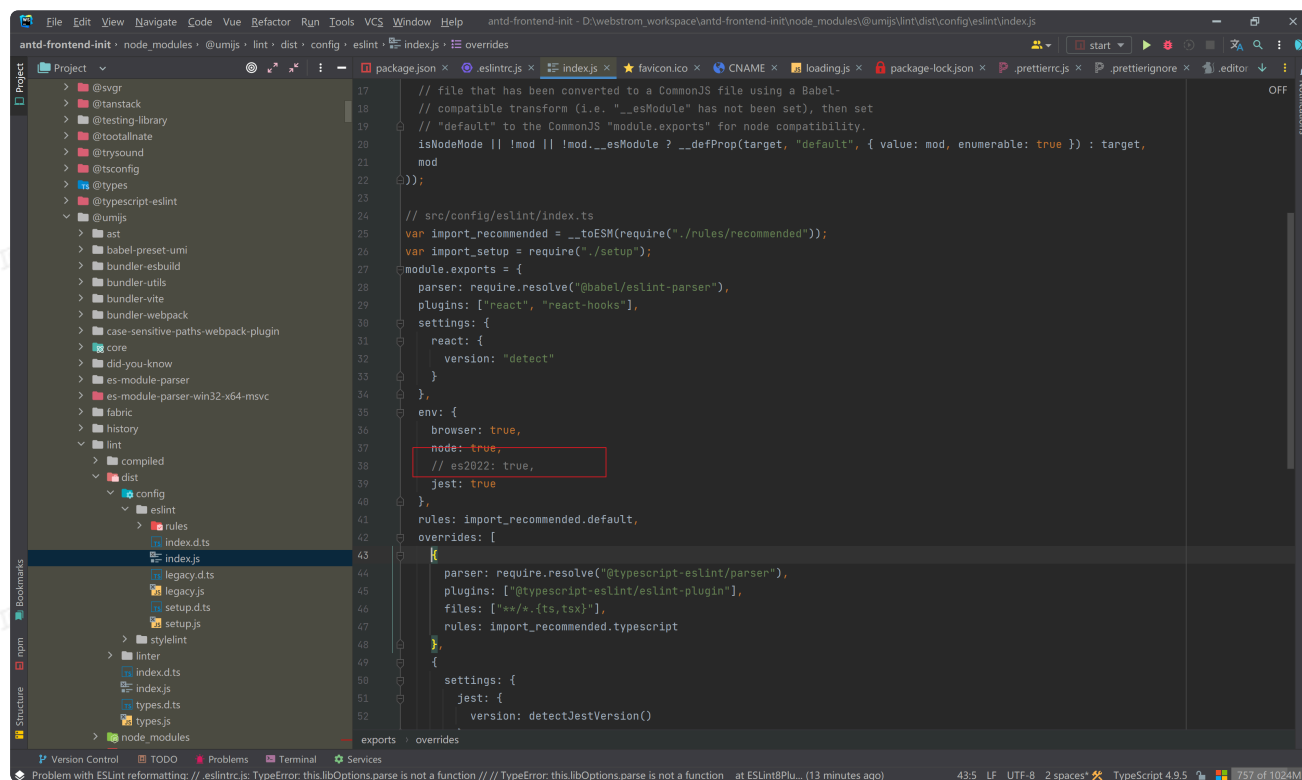
步骤：

1) 前端本地执行:

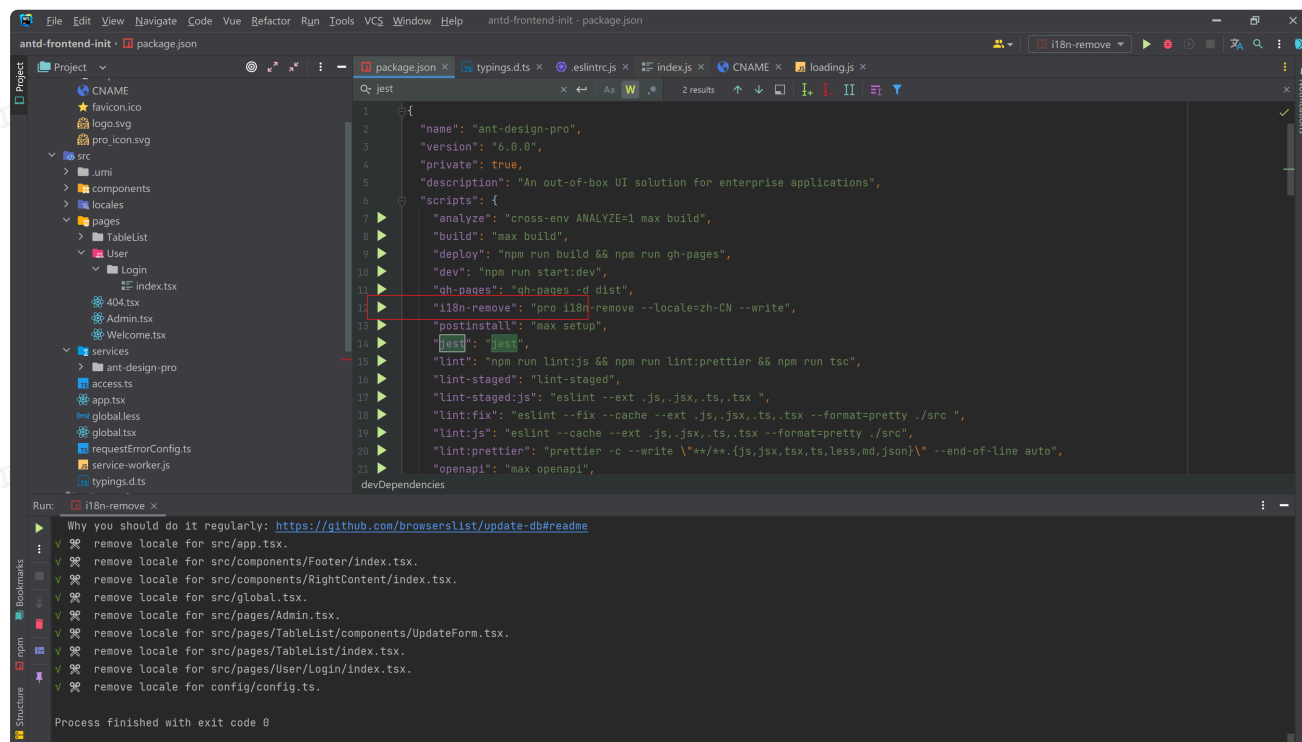
```
yarn add eslint-config-prettier --dev yarn add eslint-plugin-unicorn --dev
```

2) 注释 es2022

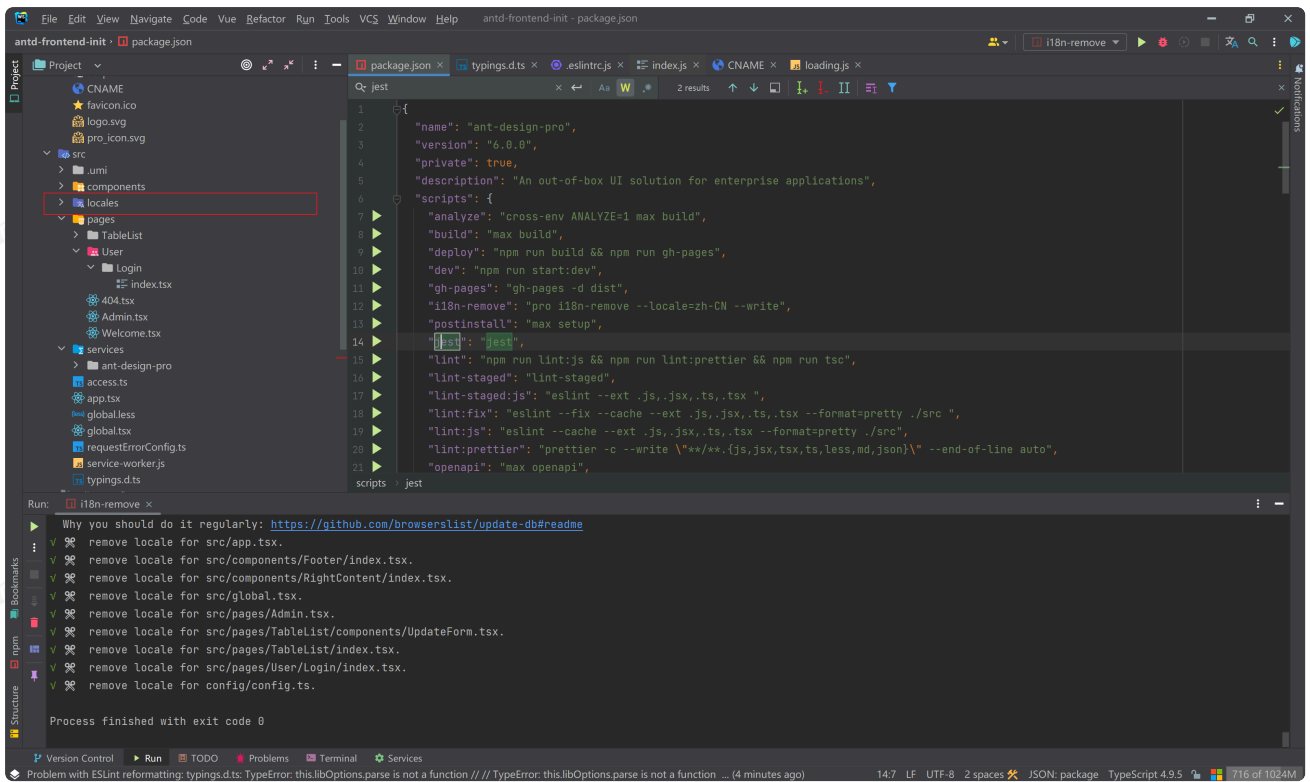
找到 node modules / @umijs/lint/dist/config/eslint/index.js, 注释掉 es2022 那行



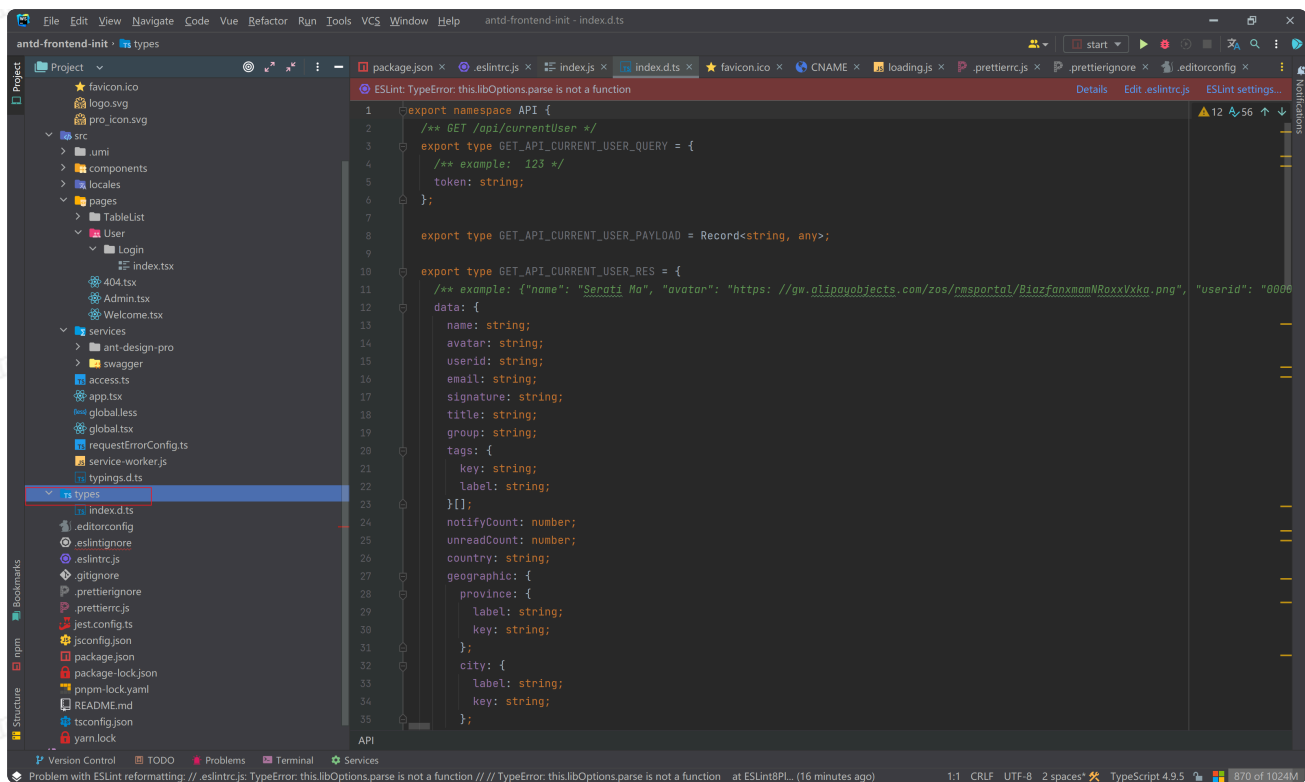
3) 执行 i18n-remove 命令



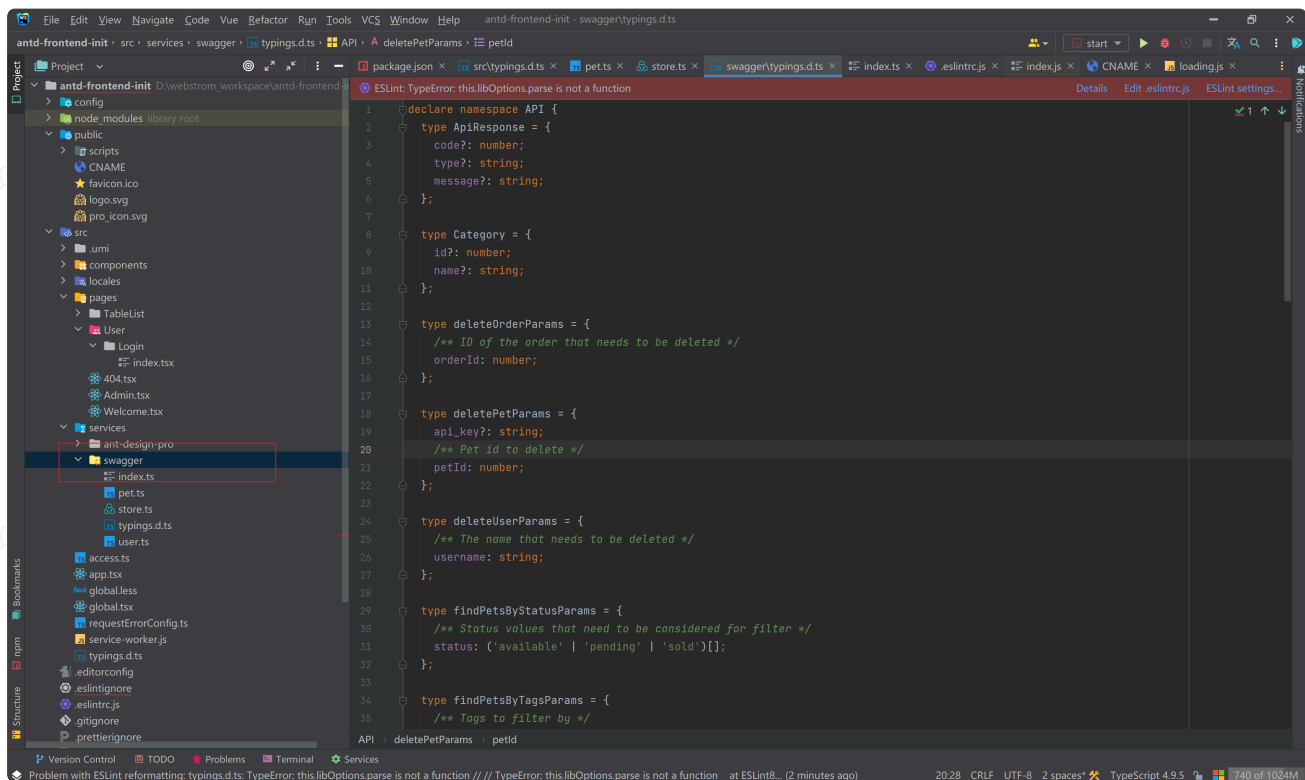
4) 删除对应的文件夹 locales



移除单元测试（Jest 相关的全部移除）

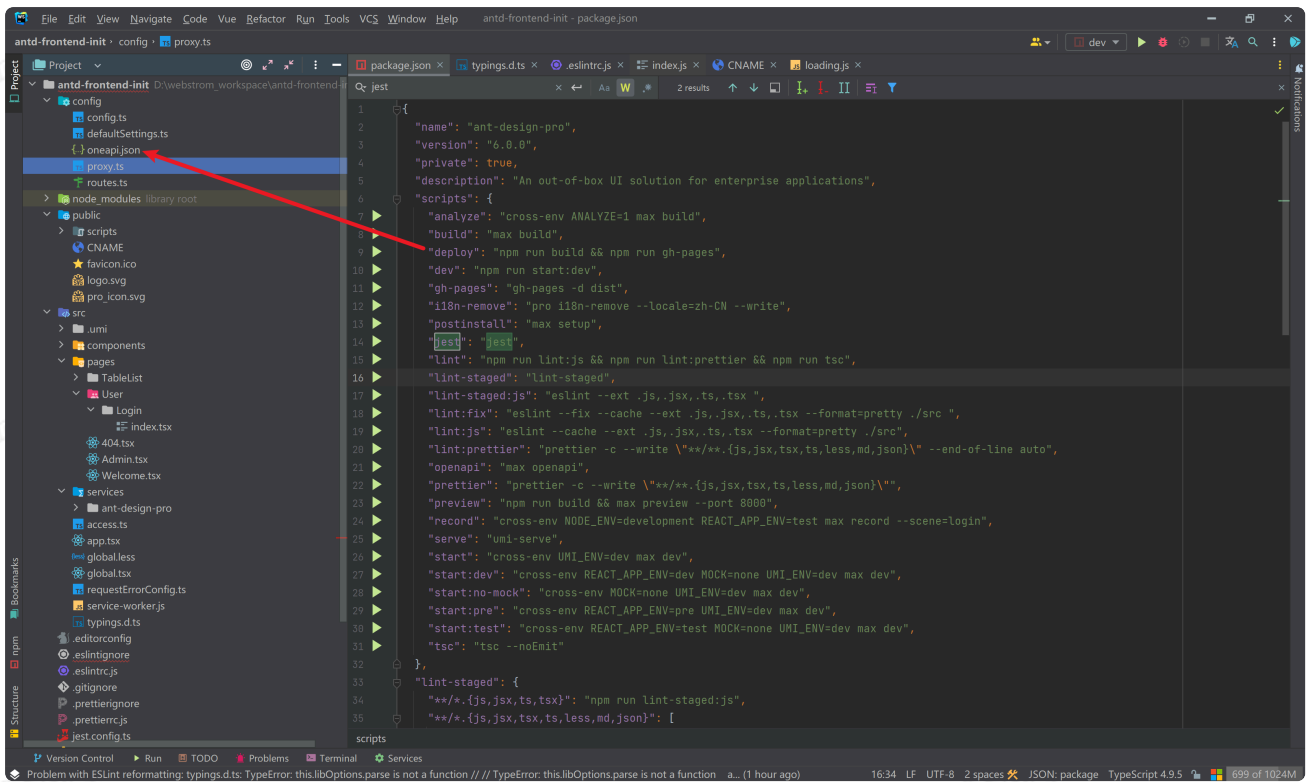


移除 swagger



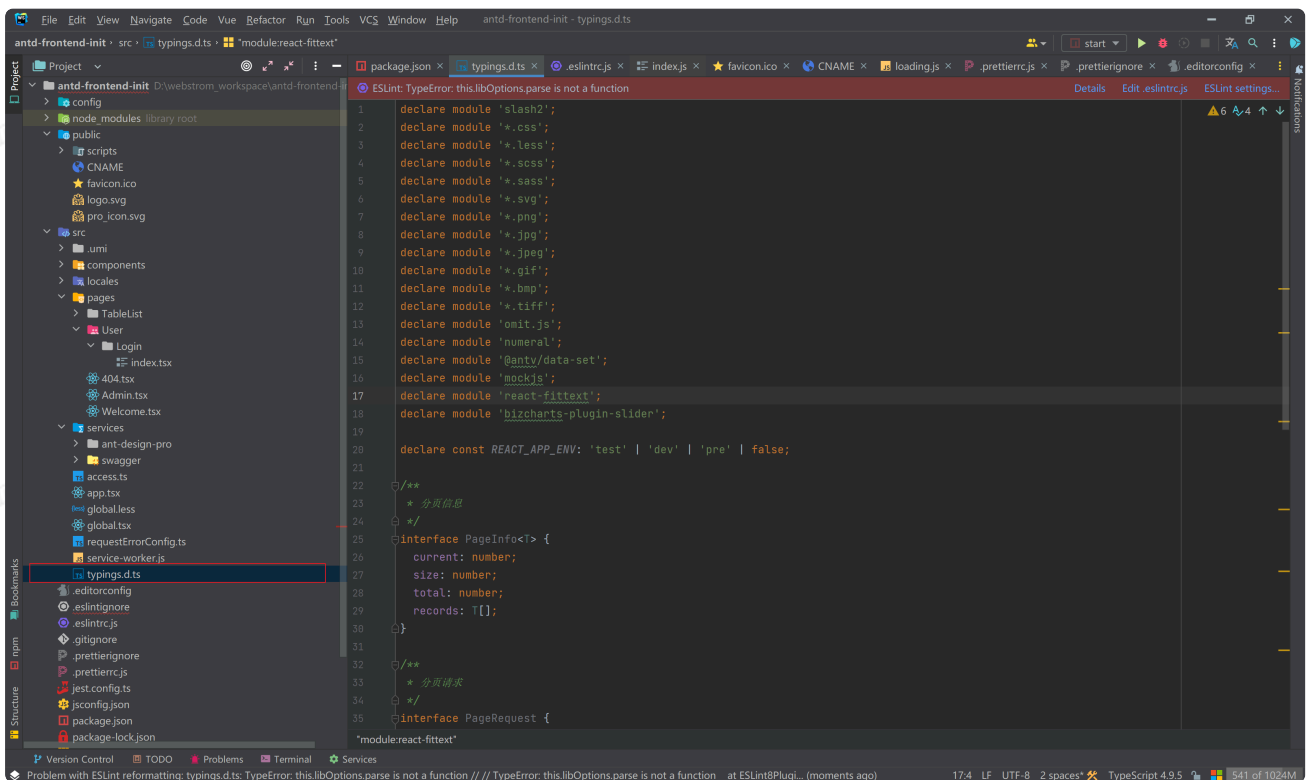
移除 openapi.json

我们有自己的后端接口地址，不需要官方提供。



基本类型

修改 typings.d.ts



```

1  declare module 'slash2';
2  declare module '*.css';
3  declare module '*.less';
4  declare module '*.scss';
5  declare module '*.sass';
6  declare module '*.svg';
7  declare module '*.png';
8  declare module '*.jpg';
9  declare module '*.jpeg';
10 declare module '*.gif';
11 declare module '*.bmp';
12 declare module '*.tiff';
13 declare module 'omit.js';
14 declare module 'numeral';
15 declare module '@antv/data-set';
16 declare module 'mockjs';
17 declare module 'react-fittext';
18 declare module 'bizcharts-plugin-slider';
19
20 declare const REACT_APP_ENV: 'test' | 'dev' | 'pre' | false;
21
22 /**
23  * 分页信息
24  */
25 interface PageInfo<T> {
26   current: number;
27   size: number;
28   total: number;
29   records: T[];
30 }
31
32 /**
33  * 分页请求
34  */
35 interface PageRequest {
36   current?: number;
37   pageSize?: number;
38   sortField?: string;
39   sortOrder?: 'ascend' | 'descend';
40 }
41
42 /**
43  * 删除请求
44  */
45 interface DeleteRequest {
46   id: number;
47 }
48
49 /**
50  * 返回封装

```

```

51  */
52  interface BaseResponse<T> {
53    code: number;
54    data: T;
55    message?: string;
56  }
57
58  /**
59   * 全局初始化状态
60  */
61  interface InitialState {
62    currentUser?: API.LoginUserV0;
63  }

```

请求

请求代码生成

注意项：由于 Swagger V3 版本文件上传请求有 BUG，还是推荐用 Swagger 2 版本。

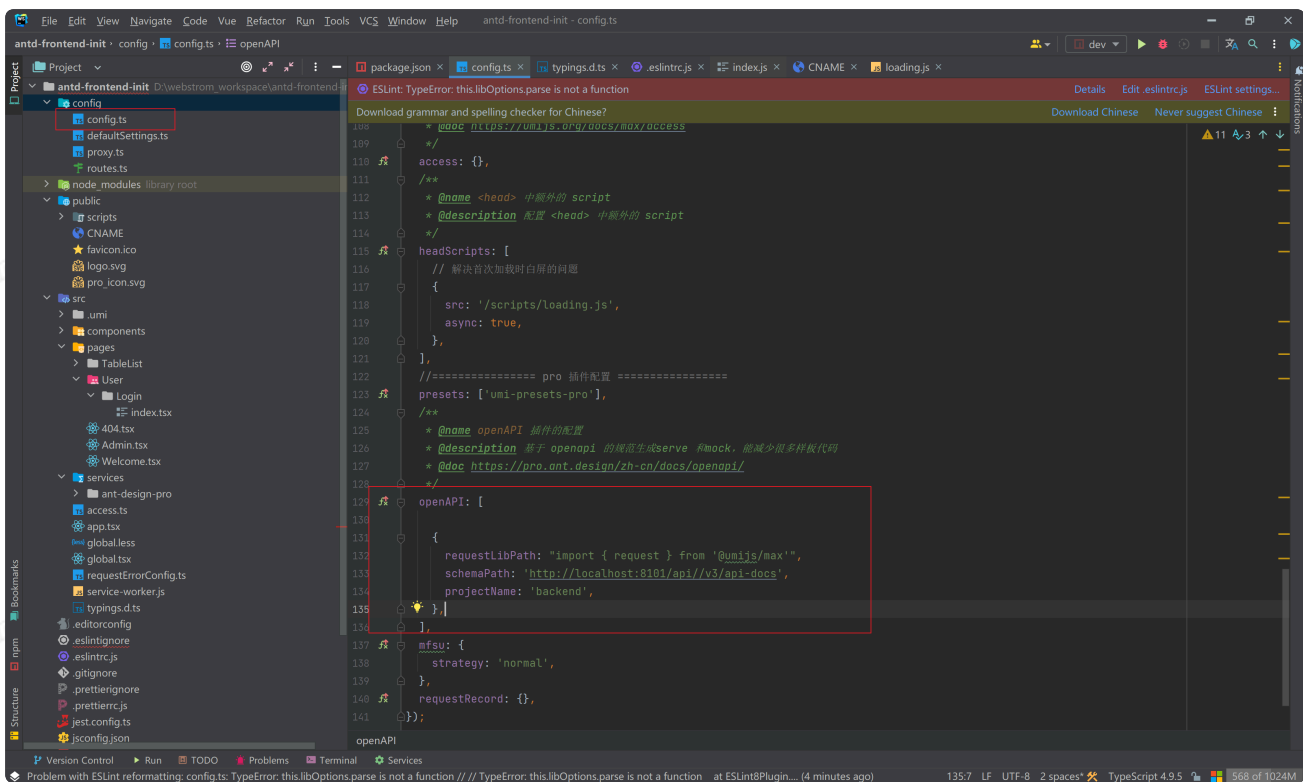
接口文档地址：<http://localhost:8101/api/v2/api-docs> <<http://localhost:8101/api/v3/api-docs>>

修改配置：

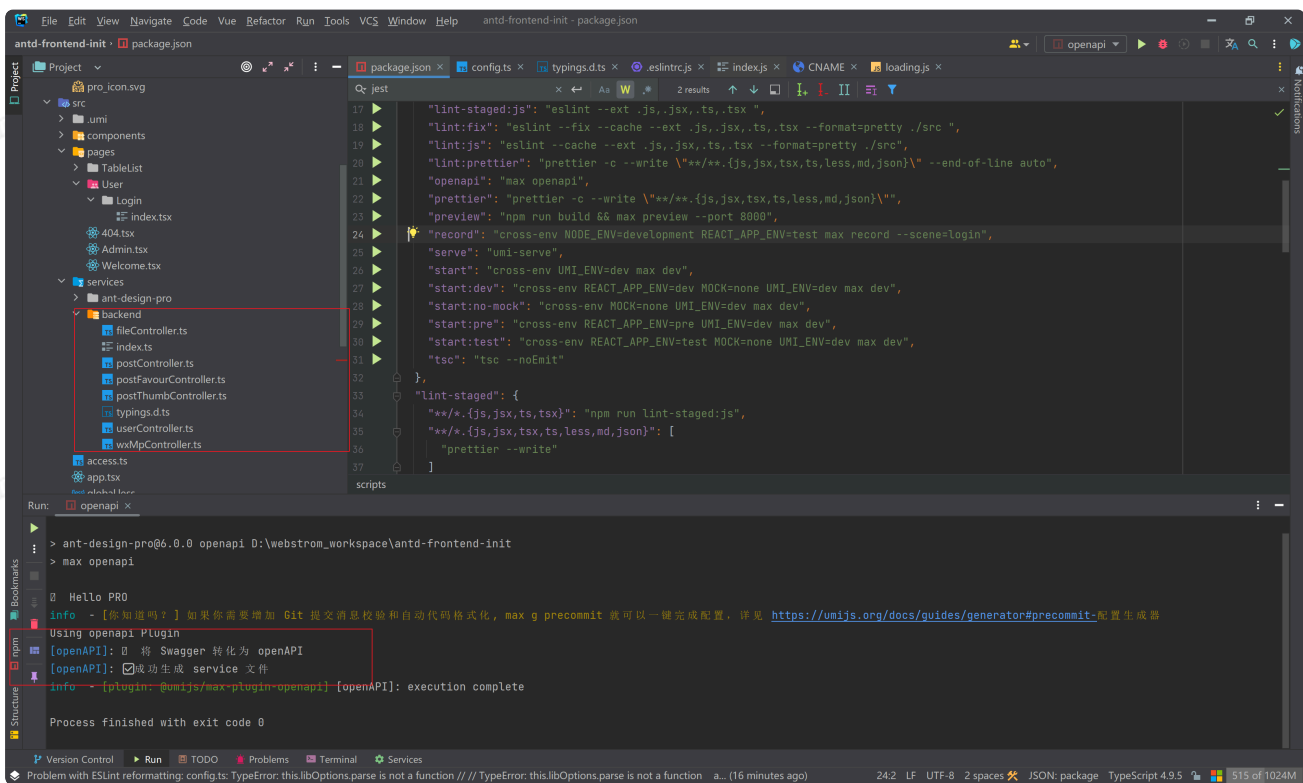
```

1  openAPI: [
2    {
3      requestLibPath: "import { request } from '@umijs/max'",
4      schemaPath: "http://localhost:8101/api/v2/api-docs",
5      projectName: 'yuzi-generator-web-backend',
6    },
7  ],

```

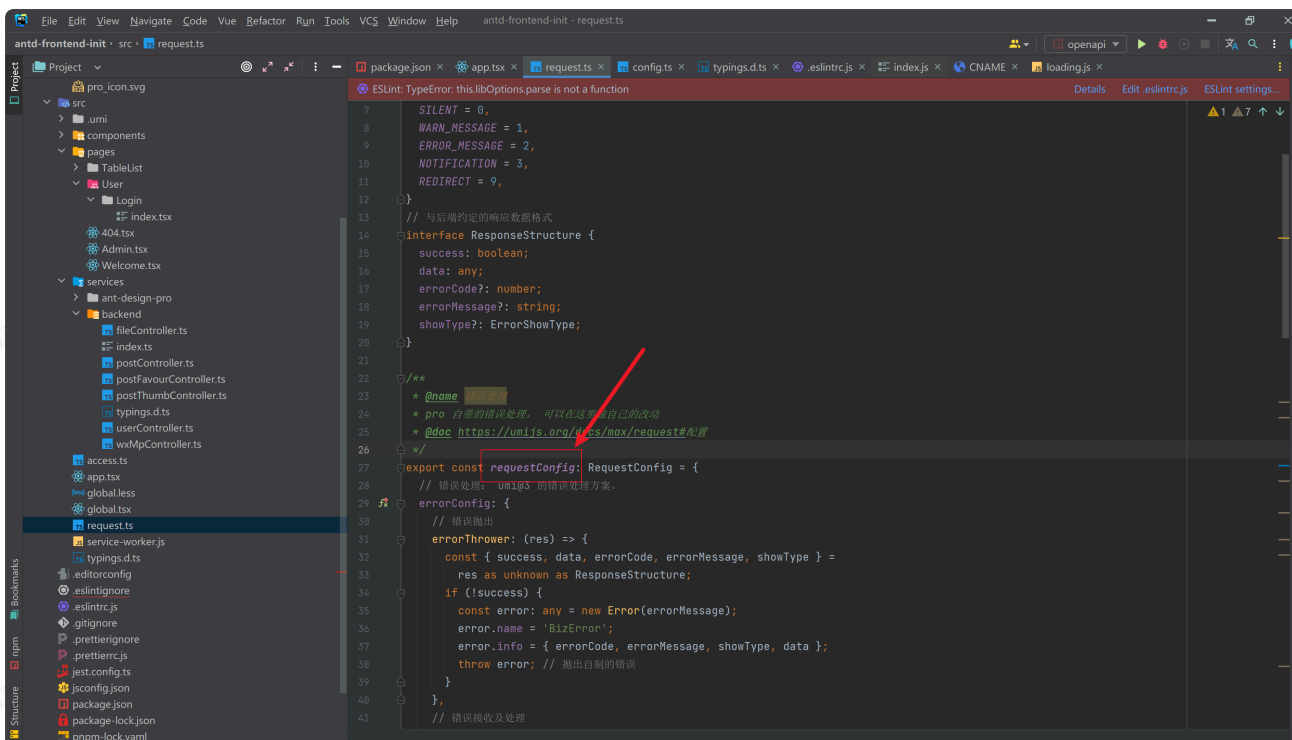


生成结果：

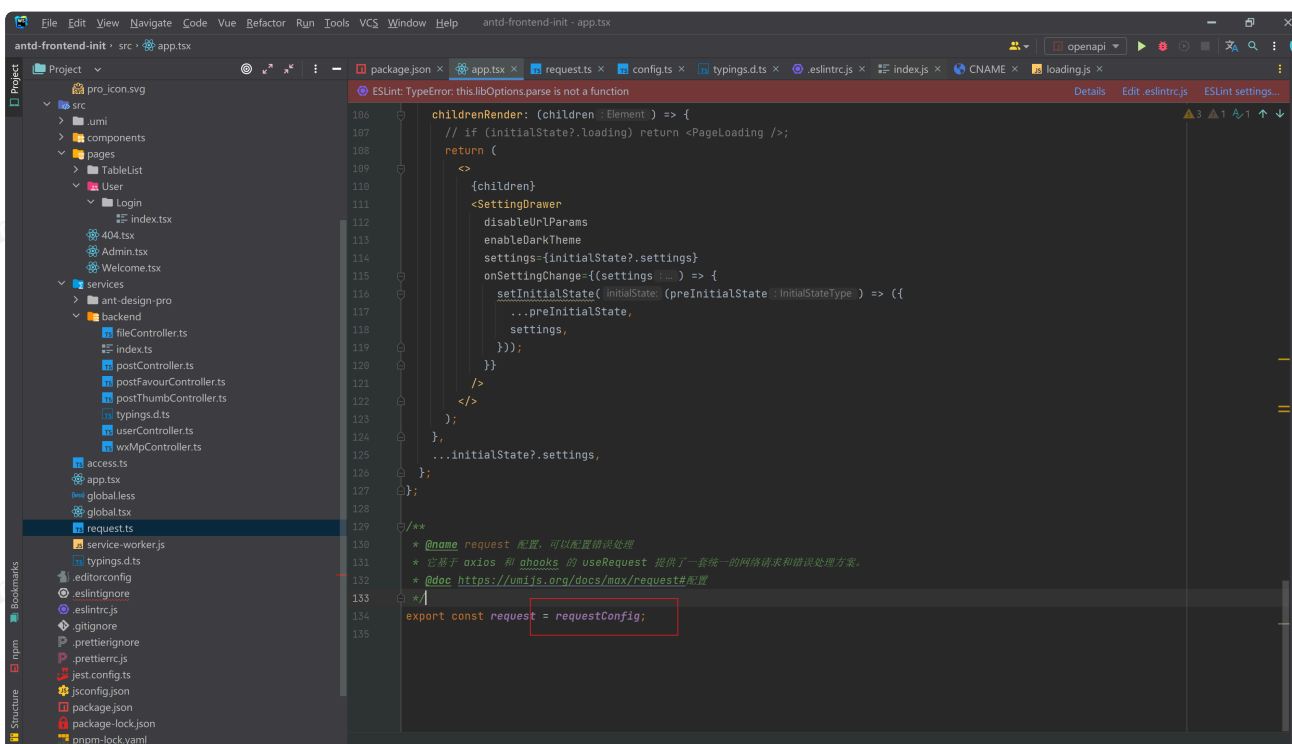


全局请求处理

- 1) 修改 requestErrorConfig.ts 的名称为 requestConfig.ts
- 2) 修改昵称至 requestConfig



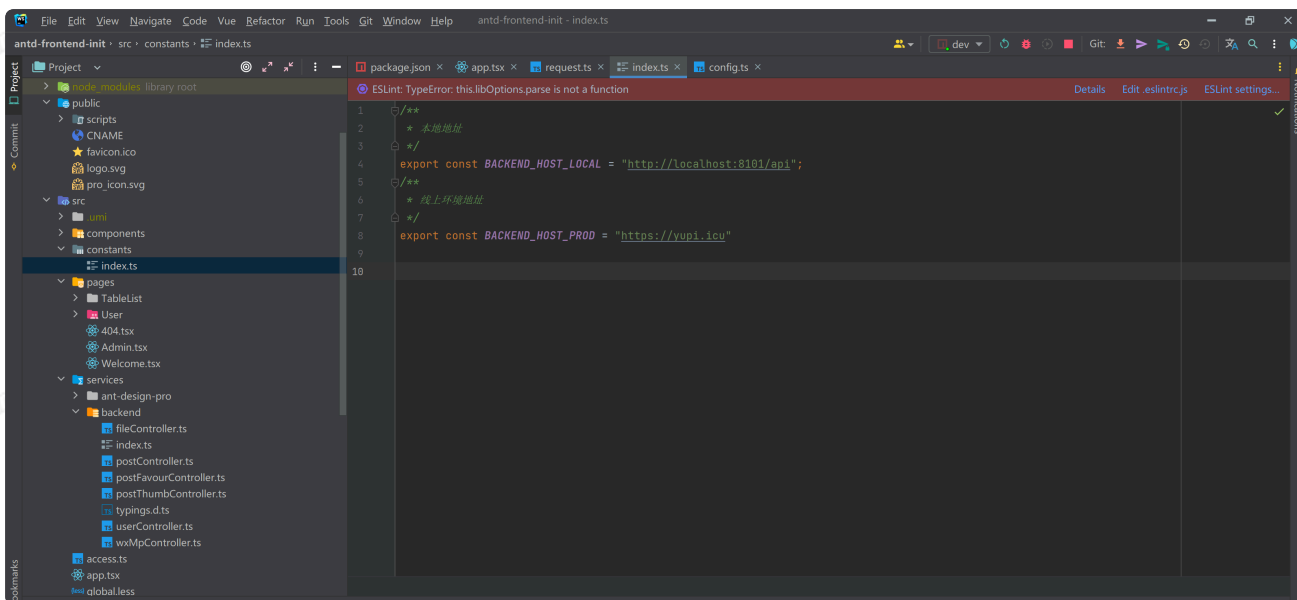
3) app.tsx 中引入



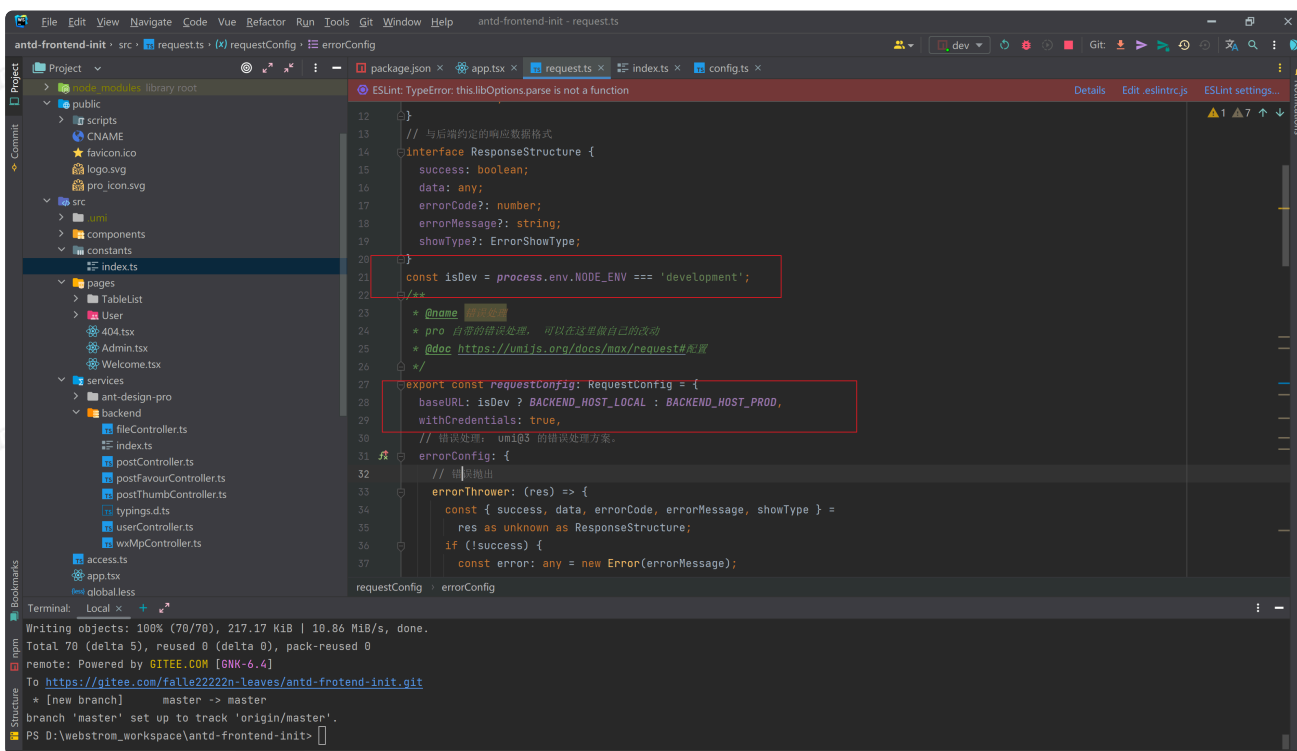
4) 创建 index.ts 常量区别开发环境和部署环境

项目-更新

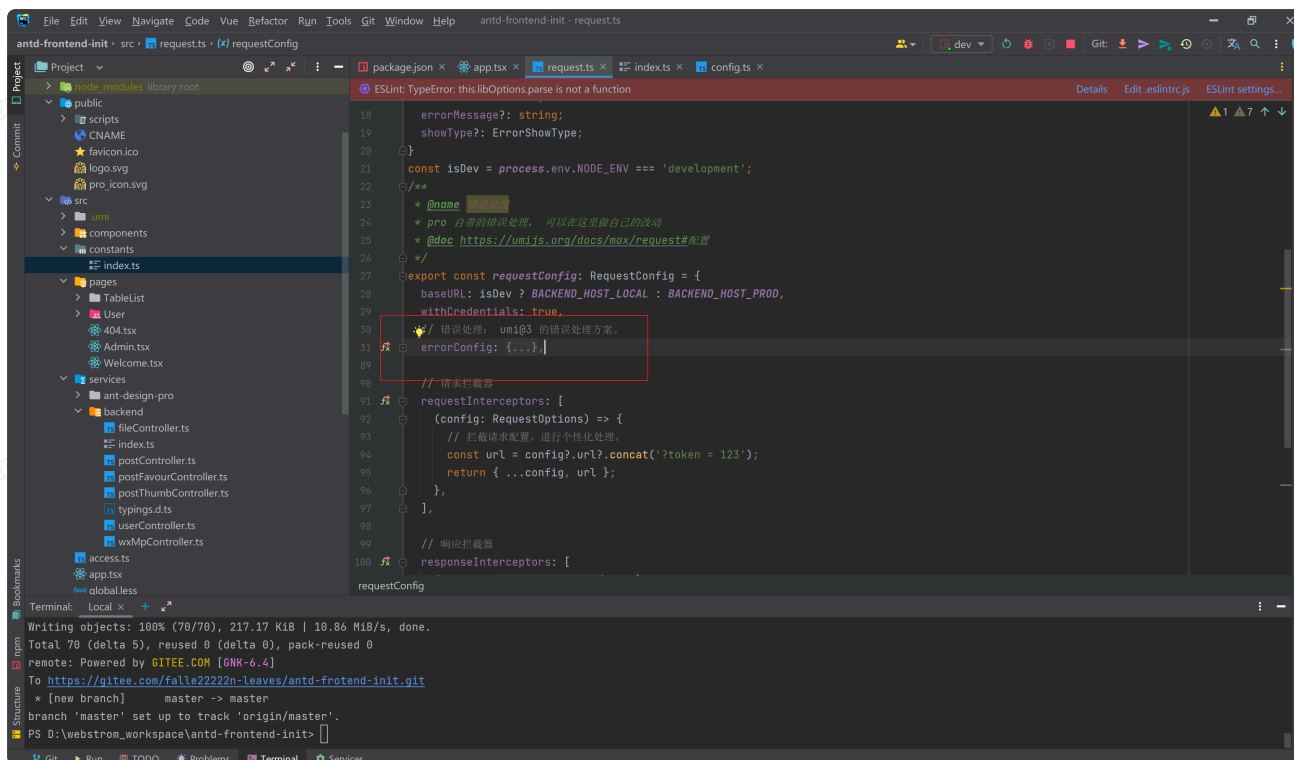
项目-更新



5) 引入环境变量，根据环境变量区分不同环境请求地址

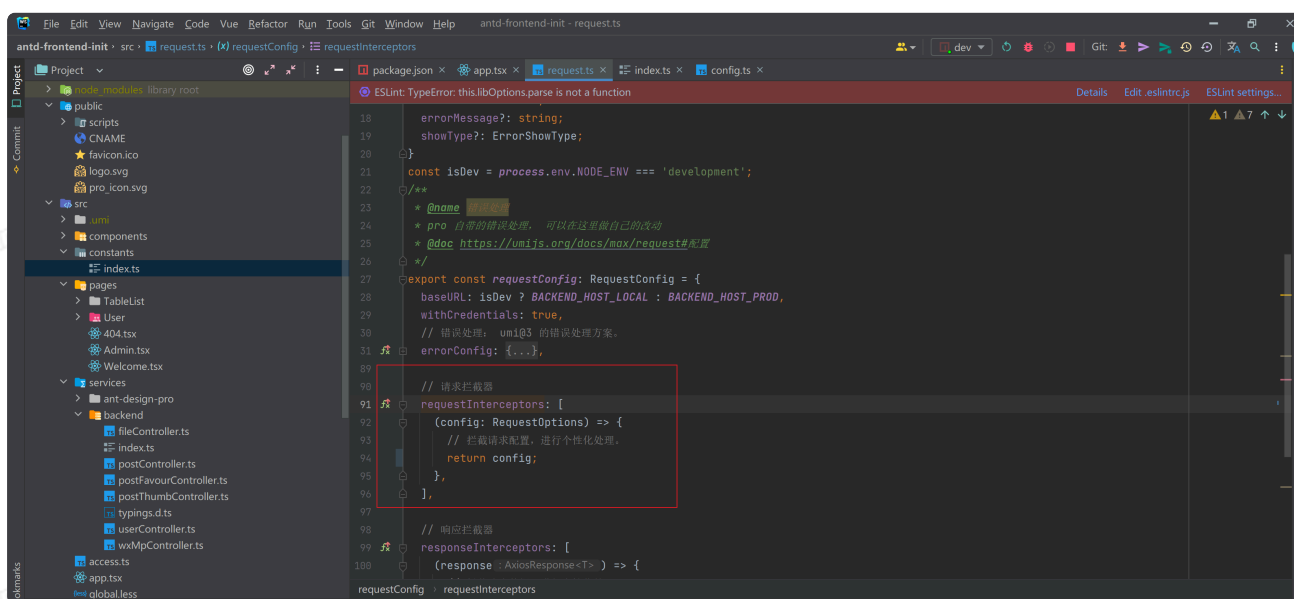


6) 删除错误处理，官方给的过于复杂，这边直接删除。

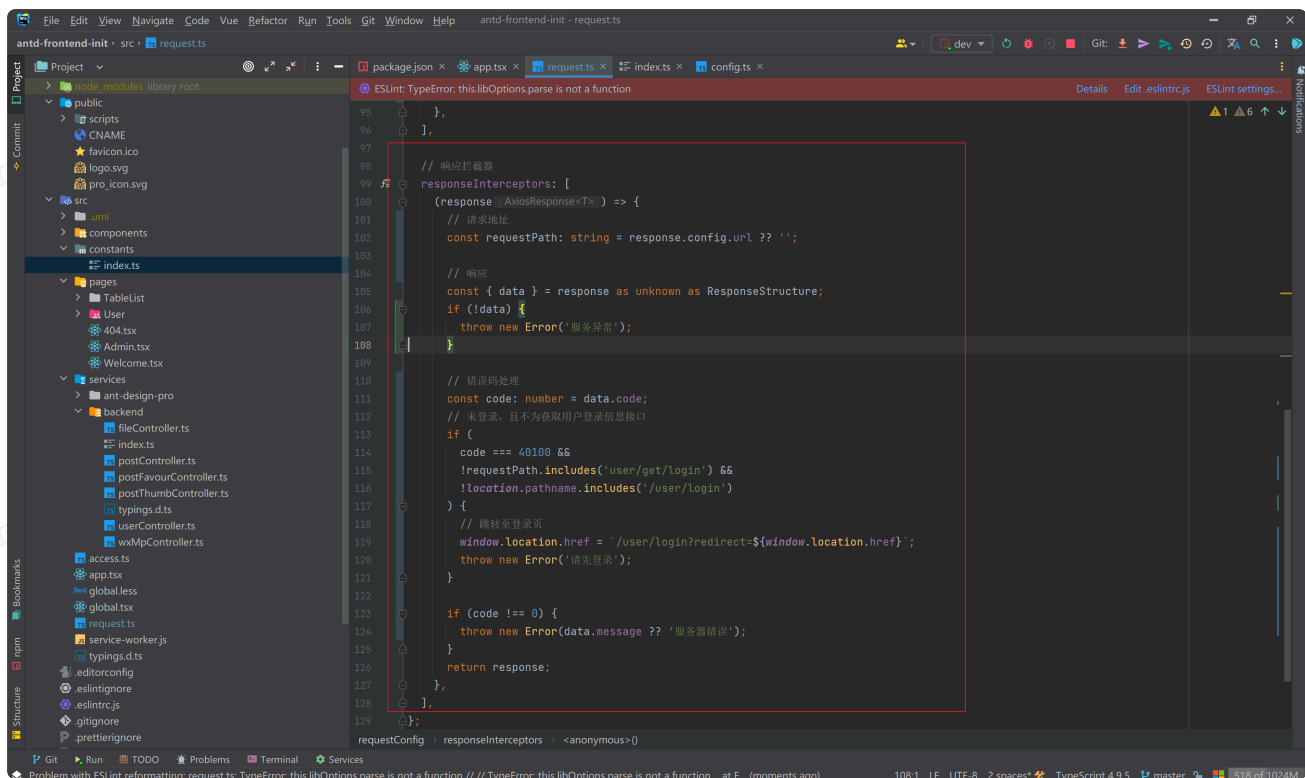


7) 删除官方的拼接 Token，一般你要使用 Token，可以直接在Authorization 请求头中携带 Token。

```
1 // 请求拦截器
2 requestInterceptors: [
3   (config: RequestOptions) => {
4     // 拦截请求配置，进行个性化处理。
5     return config;
6   },
7 ],
```



8) 修改封装好的响应拦截器



// 响应拦截器

```
responseInterceptors: [
  (response) => {
    // 请求地址
    const requestPath: string = response.config.url ?? '';
```

```
1 // 响应
2 const { data } = response as unknown as ResponseStructure;
3 if (!data) {
4   throw new Error('服务异常');
5 }
6
7 // 错误码处理
8 const code: number = data.code;
9 // 未登录，且不为获取用户登录信息接口
10 if (
11   code === 40100 &&
12   !requestPath.includes('user/get/login') &&
13   !location.pathname.includes('/user/login')
14 ) {
15   // 跳转至登录页
16   window.location.href = `/user/login?redirect=${window.location.href}`;
17   throw new Error('请先登录');
18 }
19
20 if (code !== 0) {
21   throw new Error(data.message ?? '服务器错误');
22 }
23 return response;
```



```
},  
],
```

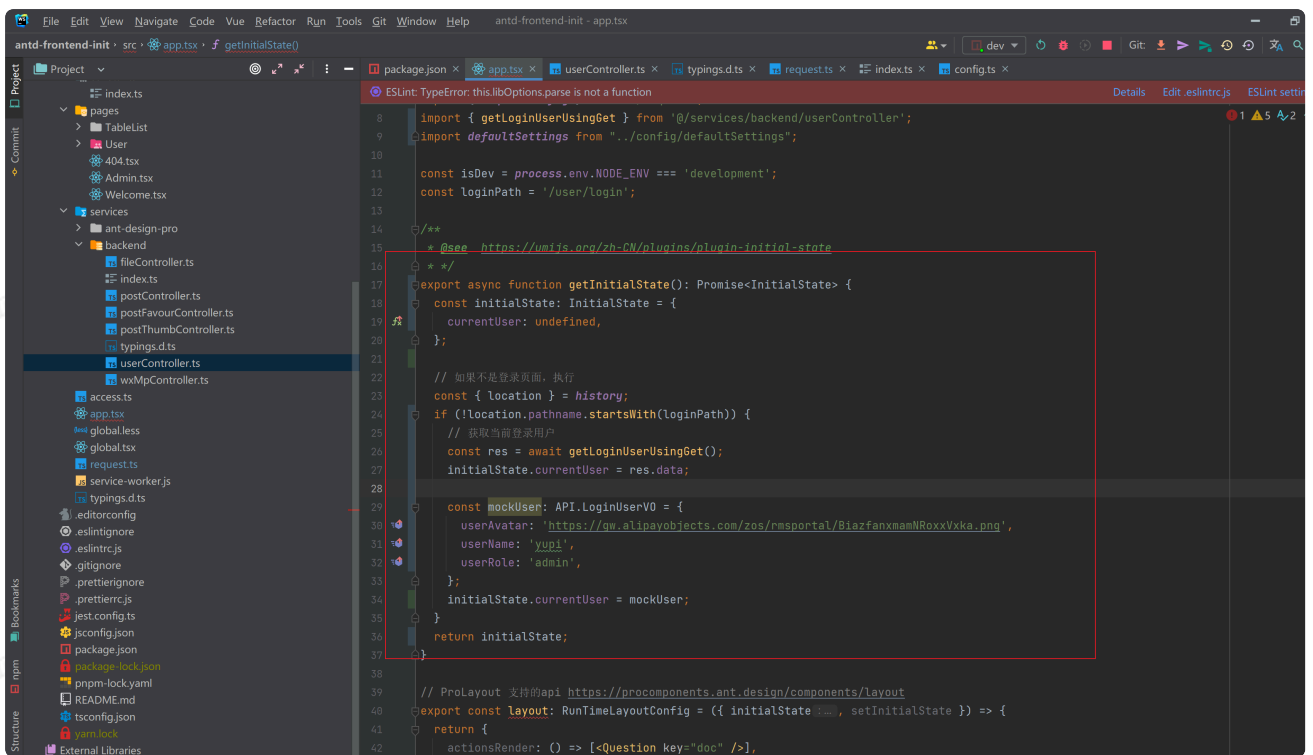
```
1  
2  ## 临时登录  
3  1) 修改 app.tsx 中的 getInitialState, 获取用户初始化状态, 先利用 Mock 模拟数据, 替  
4  
5  ```tsx  
6  /**  
7   * @see https://umijs.org/zh-CN/plugins/plugin-initial-state  
8   * */  
9  export async function getInitialState(): Promise<InitialState> {  
10    const initialState: InitialState = {  
11      currentUser: undefined,  
12    };  
13  
14    // 如果不是登录页面, 执行  
15    const { location } = history;  
16    if (!location.pathname.startsWith(loginPath)) {  
17      // 获取当前登录用户  
18      const res = await getLoginUser();  
19      initialState.currentUser = res.data;  
20  
21      const mockUser: API.LoginUserVO = {  
22        userAvatar: 'https://gw.alipayobjects.com/zos/rmsportal/BiazfanxmamNRox  
23        userName: 'yupi',  
24        userRole: 'admin',  
25      };  
26      initialState.currentUser = mockUser;  
27    }  
28    return initialState;  
29  }
```

项目-更新

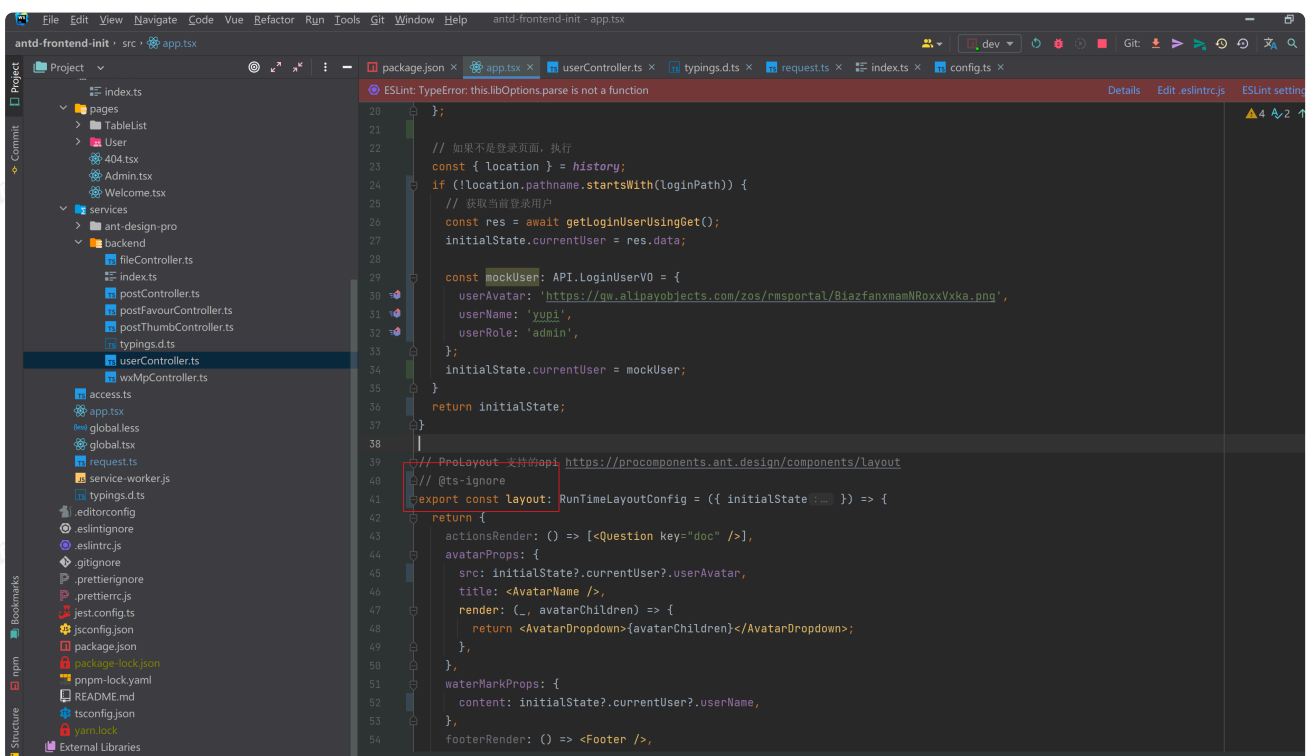
项目-更新

项目-更新

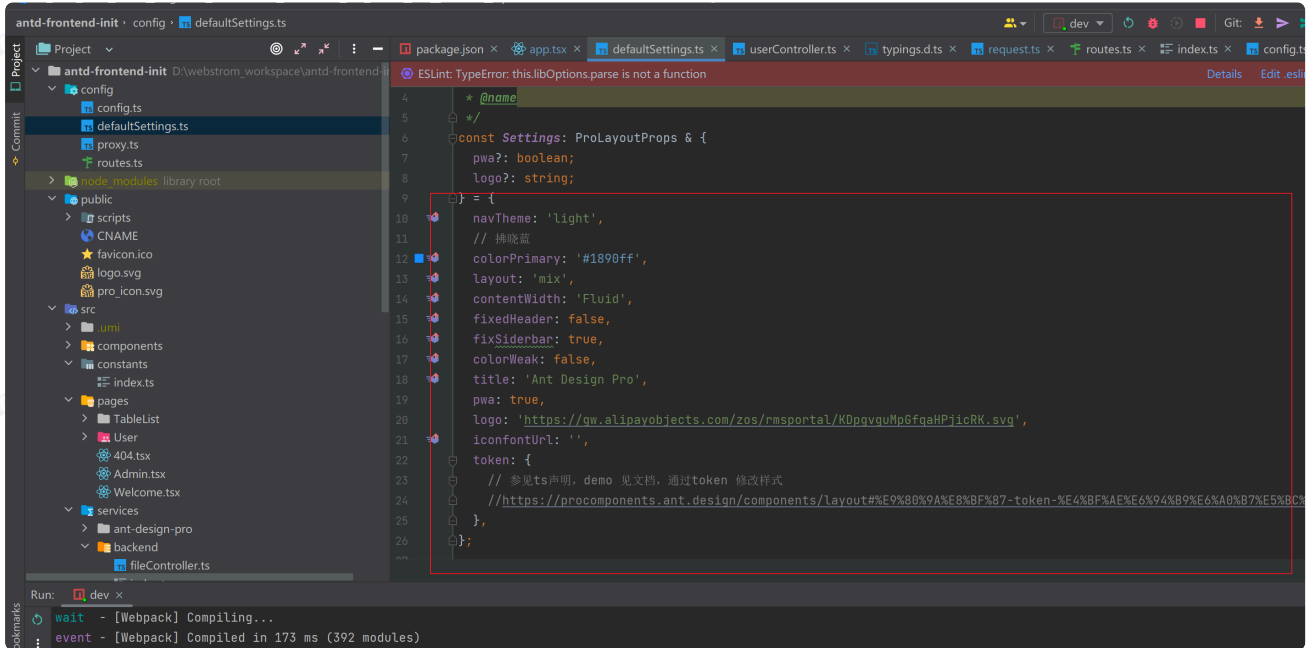
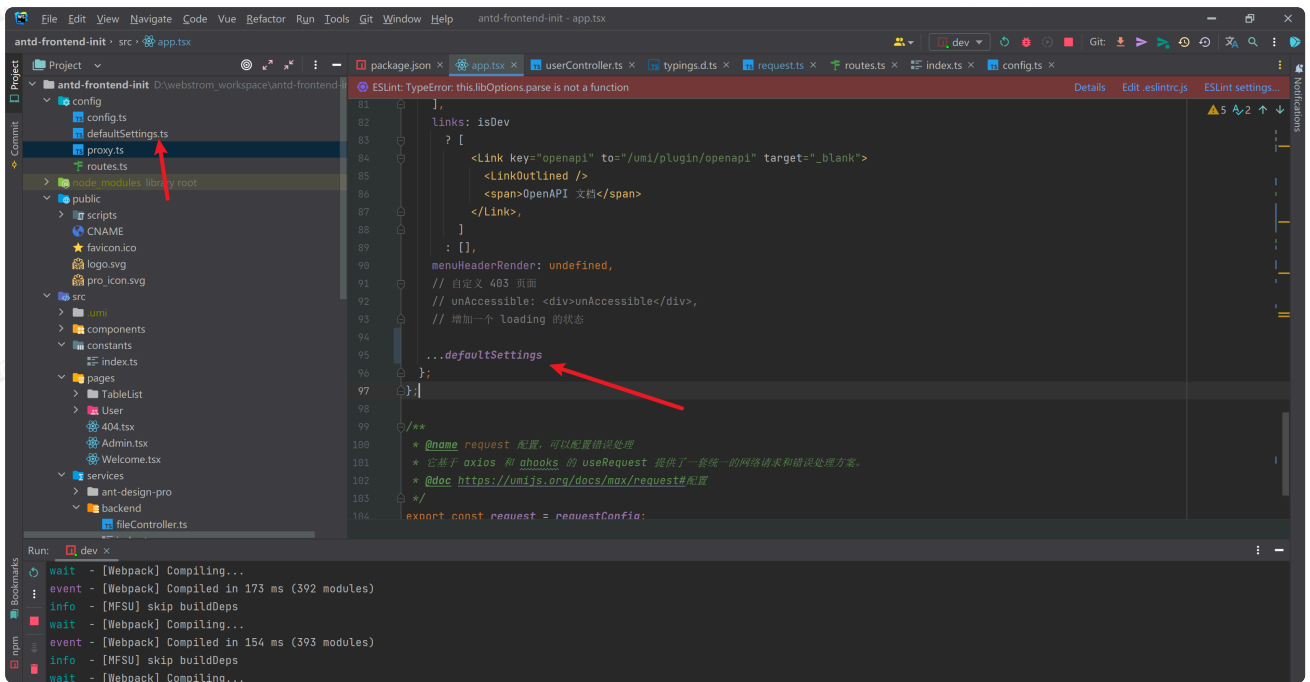
项目-更新



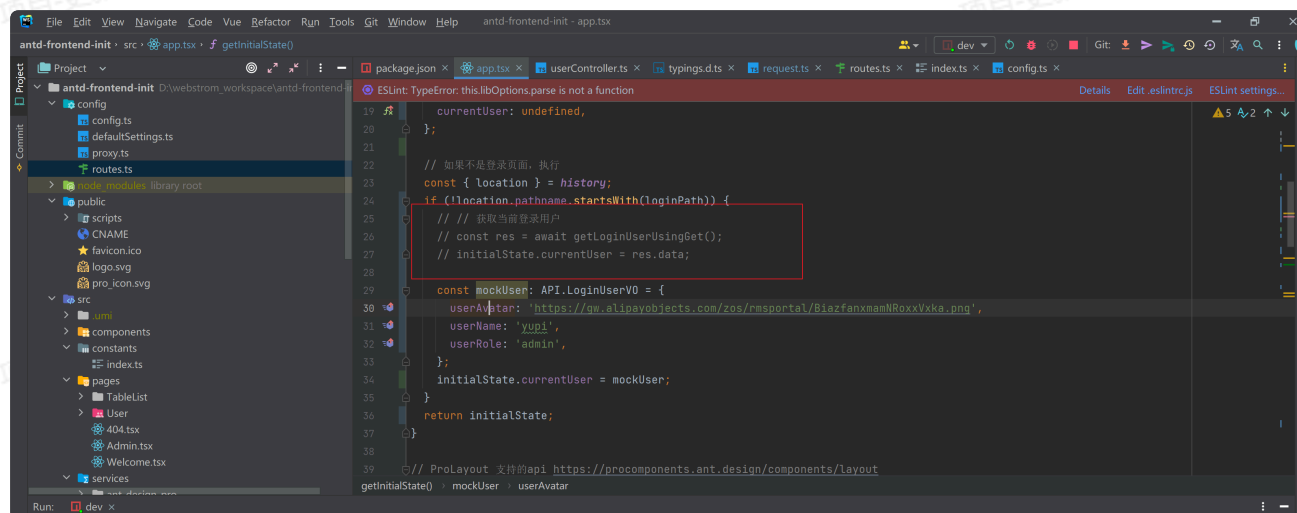
2) 先用 @ ts-ignore 忽略 ts 类型提示错误



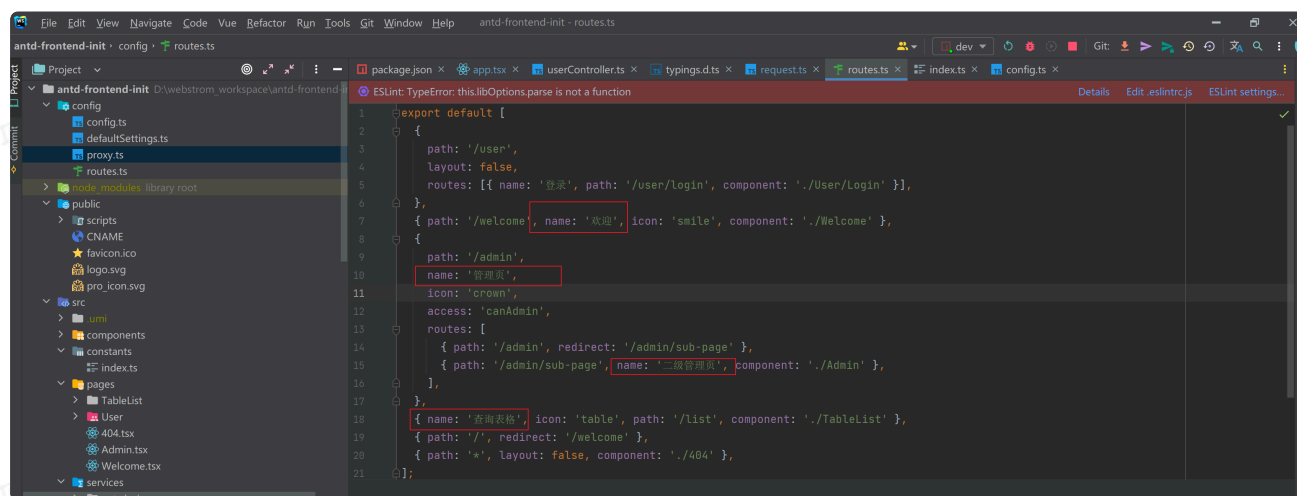
2.1) 移除无用配置，比如 setting drawers 可以不要，需要可视化得到配置可以使用在线地址：[分析页 - Ant Design Pro <https://preview.pro.ant.design/dashboard/analysis>](https://preview.pro.ant.design/dashboard/analysis)



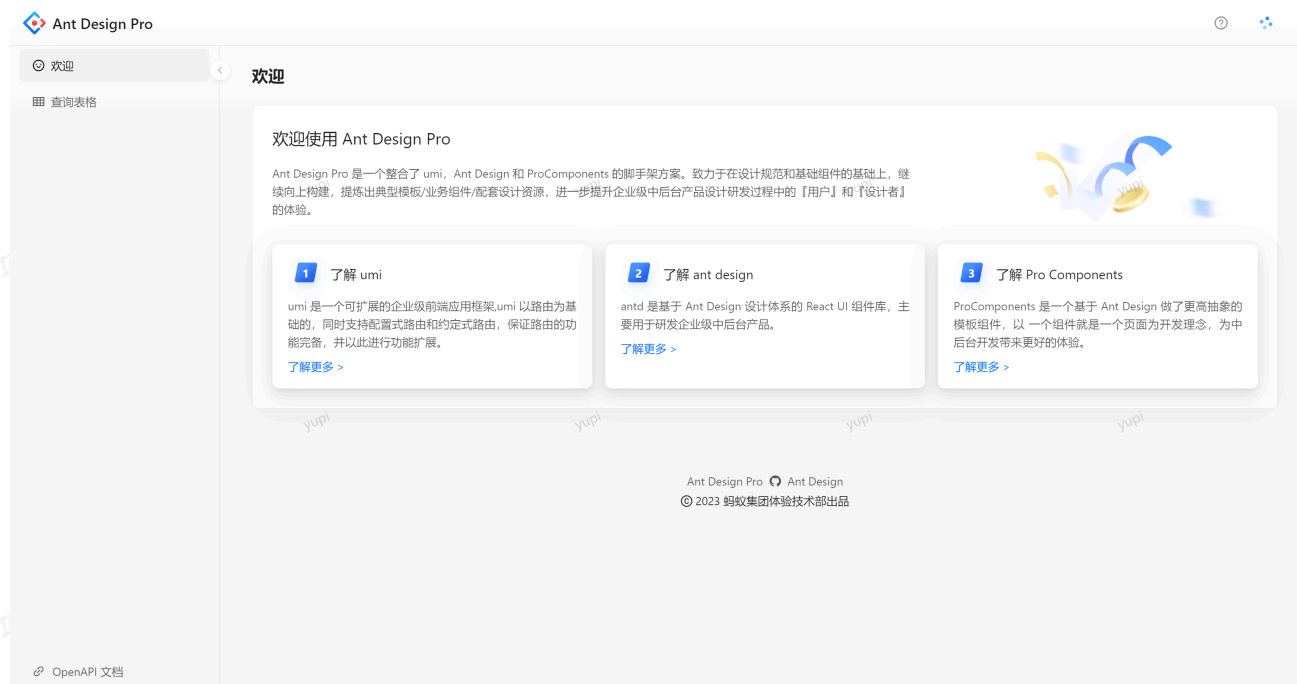
3) 注释请求后端的代码，访问主页面，不会再重定向到登录页



4) 如果侧边栏没有展示，给路由加上 name 属性即可



5) 查看登录效果

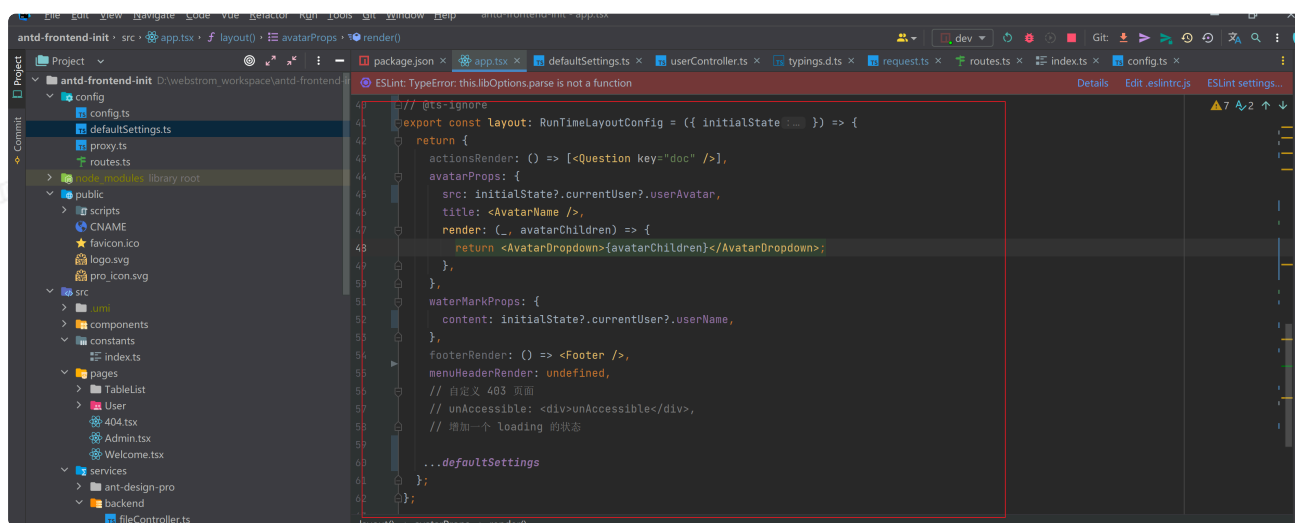


6) 优化 layout 的配置

```

1 // ProLayout 支持的api https://procomponents.ant.design/components/layout
2 // @ts-ignore
3 export const layout: RunTimeLayoutConfig = ({ initialState }) => {
4   return {
5     actionsRender: () => [<Question key="doc" />],
6     avatarProps: {
7       src: initialState?.currentUser?.userAvatar,
8       title: <AvatarName />,
9       render: (_, avatarChildren) => {
10         return <AvatarDropdown>{avatarChildren}</AvatarDropdown>;
11       },
12     },
13     watermarkProps: {
14       content: initialState?.currentUser?.userName,
15     },
16     footerRender: () => <Footer />,
17     menuHeaderRender: undefined,
18     // 自定义 403 页面
19     // unAccessible: <div>unAccessible</div>,
20     // 增加一个 loading 的状态
21     ...defaultSettings
22   };
23 };
24 };

```



基础布局

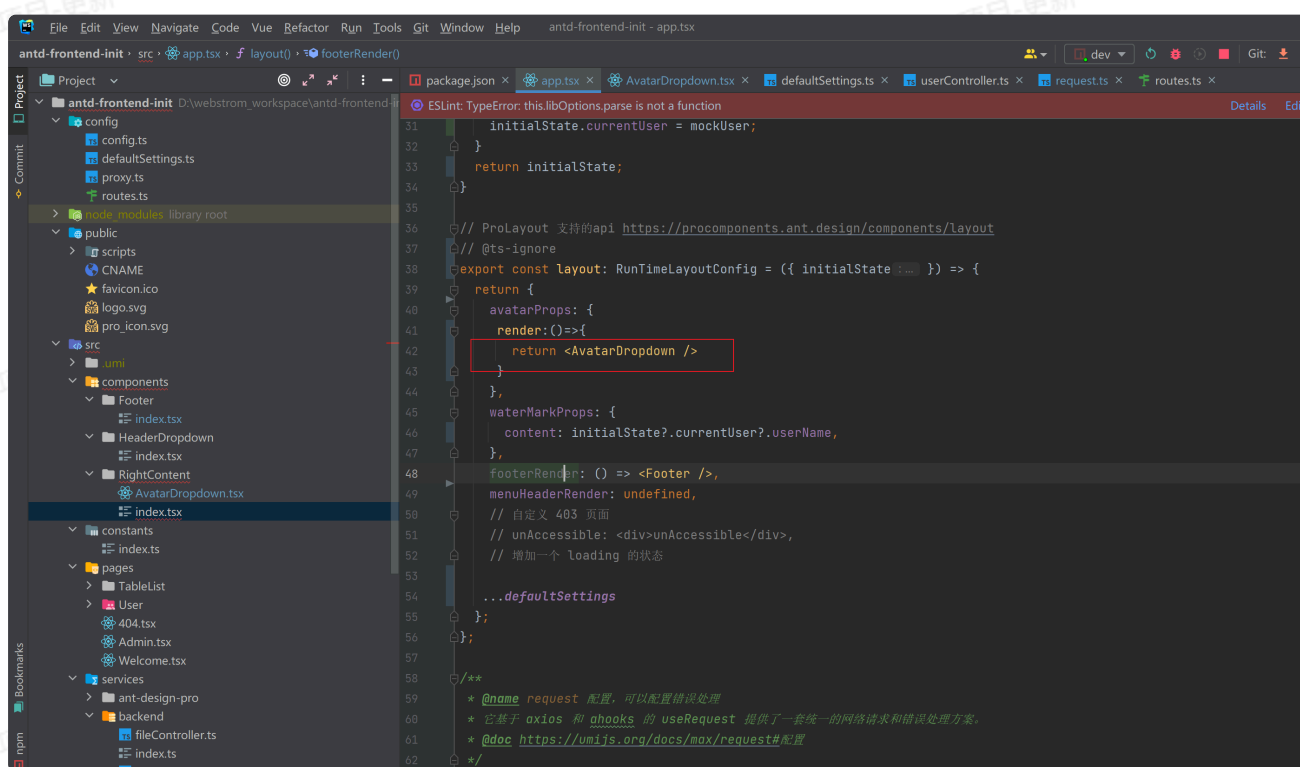
右上角信息

1) 修改 app.tsx 的 layout

```

1 // ProLayout 支持的api https://procomponents.ant.design/components/layout
2 // @ts-ignore
3 export const layout: RunTimeLayoutConfig = ({ initialState }) => {
4   return {
5     avatarProps: {
6       render: ()=>{
7         return <AvatarDropdown />
8       }
9     },
10    watermarkProps: {
11      content: initialState?.currentUser?.userName,
12    },
13    footerRender: () => <Footer />,
14    menuHeaderRender: undefined,
15    // 自定义 403 页面
16    // unAccessible: <div>unAccessible</div>,
17    // 增加一个 loading 的状态
18
19    ...defaultSettings
20  };
21 };

```



2) 修改 AvatarDropdown.tsx

```

1  import {userLogoutUsingPost} from '@services/backend/userController';
2  import {LogoutOutlined, SettingOutlined, UserOutlined} from '@ant-design/ico
3  import {history, useModel} from '@umijs/max';
4  import {Avatar, Button, Space} from 'antd';
5  import {stringify} from 'querystring';
6  import type {MenuInfo} from 'rc-menu/lib/interface';
7  import React, {useCallback} from 'react';
8  import {flushSync} from 'react-dom';
9  import {Link} from 'umi';
10 import HeaderDropdown from '../HeaderDropdown';
11
12 export type GlobalHeaderRightProps = {
13   menu?: boolean;
14   children?: React.ReactNode;
15 };
16
17 export const AvatarDropdown: React.FC<GlobalHeaderRightProps> = ({menu}) =>
18   /**
19    * 退出登录，并且将当前的 url 保存
20    */
21   const loginOut = async () => {
22     await userLogoutUsingPost();
23     const {search, pathname} = window.location;
24     const urlParams = new URL(window.location.href).searchParams;
25     /** 此方法会跳转到 redirect 参数所在的位置 */
26     const redirect = urlParams.get('redirect');
27     // Note: There may be security issues, please note
28     if (window.location.pathname !== '/user/login' && !redirect) {
29       history.replace({
30         pathname: '/user/login',
31         search: stringify({
32           redirect: pathname + search,
33         }),
34       });
35     }
36   };
37
38   const {initialState, setInitialState} = useModel('@@initialState');
39
40   const onMenuClick = useCallback(
41     (event: MenuInfo) => {
42       const {key} = event;
43       if (key === 'logout') {
44         flushSync(() => {
45           setInitialState((s) => ({...s, currentUser: undefined}));
46         });
47         loginOut();
48         return;
49       }
50       history.push(`/account/${key}`);

```

```

51     },
52     [setInitialState],
53   );
54
55   const {currentUser} = initialState || {};
56
57   // 未登录
58   if (!currentUser) {
59     return (
60       <Link to="/user/login">
61         <Button type="primary" shape="round">
62           登录
63         </Button>
64
65       </Link>
66     );
67   }
68
69   const menuItems = [
70     ...(menu
71       ? [
72         {
73           key: 'center',
74           icon: <UserOutlined/>,
75           label: '个人中心',
76         },
77         {
78           key: 'settings',
79           icon: <SettingOutlined/>,
80           label: '个人设置',
81         },
82         {
83           type: 'divider' as const,
84         },
85       ]
86       : []),
87     {
88       key: 'logout',
89       icon: <LogoutOutlined/>,
90       label: '退出登录',
91     },
92   ];
93
94   return (
95     <HeaderDropdown
96       menu={{
97         selectedKeys: [],
98         onClick: onMenuClick,
99         items: menuItems,
100       }}
101

```



```

102     >
103     <Space>
104       {currentUser?.userAvatar ? (<Avatar size="small" src={currentUser?.u
105       (<Avatar size="small" icon={<UserOutlined/>}/>)}
106
107       <span className="anticon">{currentUser?.userName ?? '无名'}</span>
108
109     </Space>
110
111   </HeaderDropdown>
112
113   );

```

效果图：



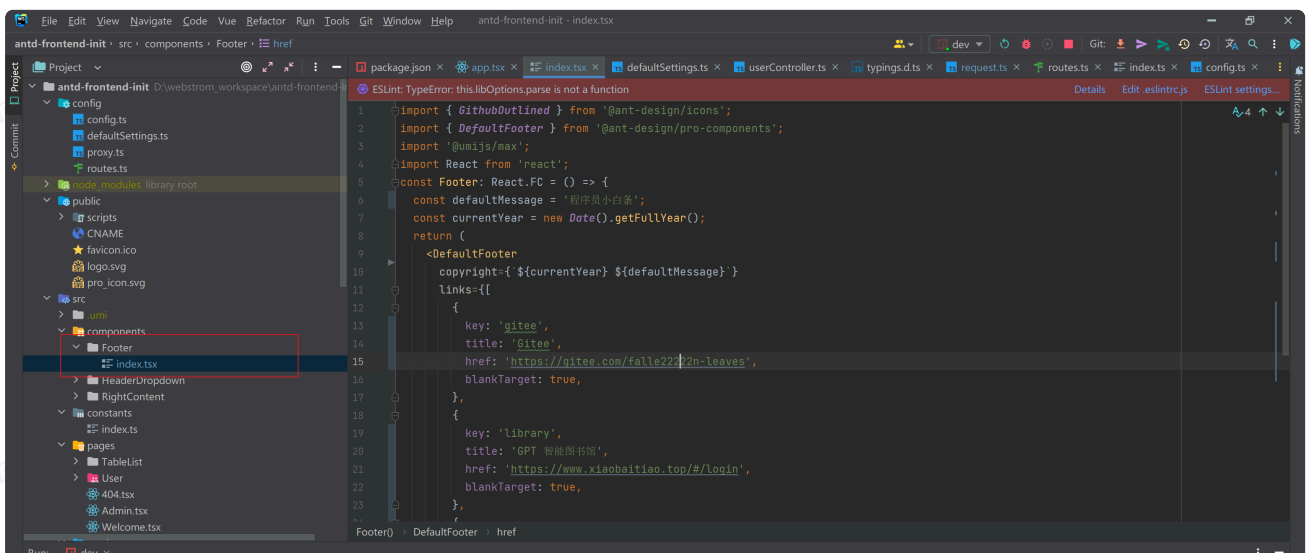
底部信息

修改成你自己的个人博客，GitHub等地址就可以了。

```

1  import { GithubOutlined } from '@ant-design/icons';
2  import { DefaultFooter } from '@ant-design/pro-components';
3  import '@umijs/max';
4  import React from 'react';
5  const Footer: React.FC = () => {
6    const defaultMessage = '程序员鱼皮';
7    const currentYear = new Date().getFullYear();
8    return (
9      <DefaultFooter
10        copyright={` ${currentYear} ${defaultMessage}`}
11        links={[
12          {
13            key: 'codeNav',
14            title: '编程导航',
15            href: 'https://yupi.icu',
16            blankTarget: true,
17          },
18          {
19            key: 'Ant Design',
20            title: '编程宝典',
21            href: 'https://codefather.cn',
22            blankTarget: true,
23          },
24          {
25            key: 'github',
26            title: <><GithubOutlined /> 鱼皮源码</>,
27            href: 'https://github.com/liyupi',
28            blankTarget: true,
29          },
30        ]}
31      />
32    );
33  };
34  export default Footer;

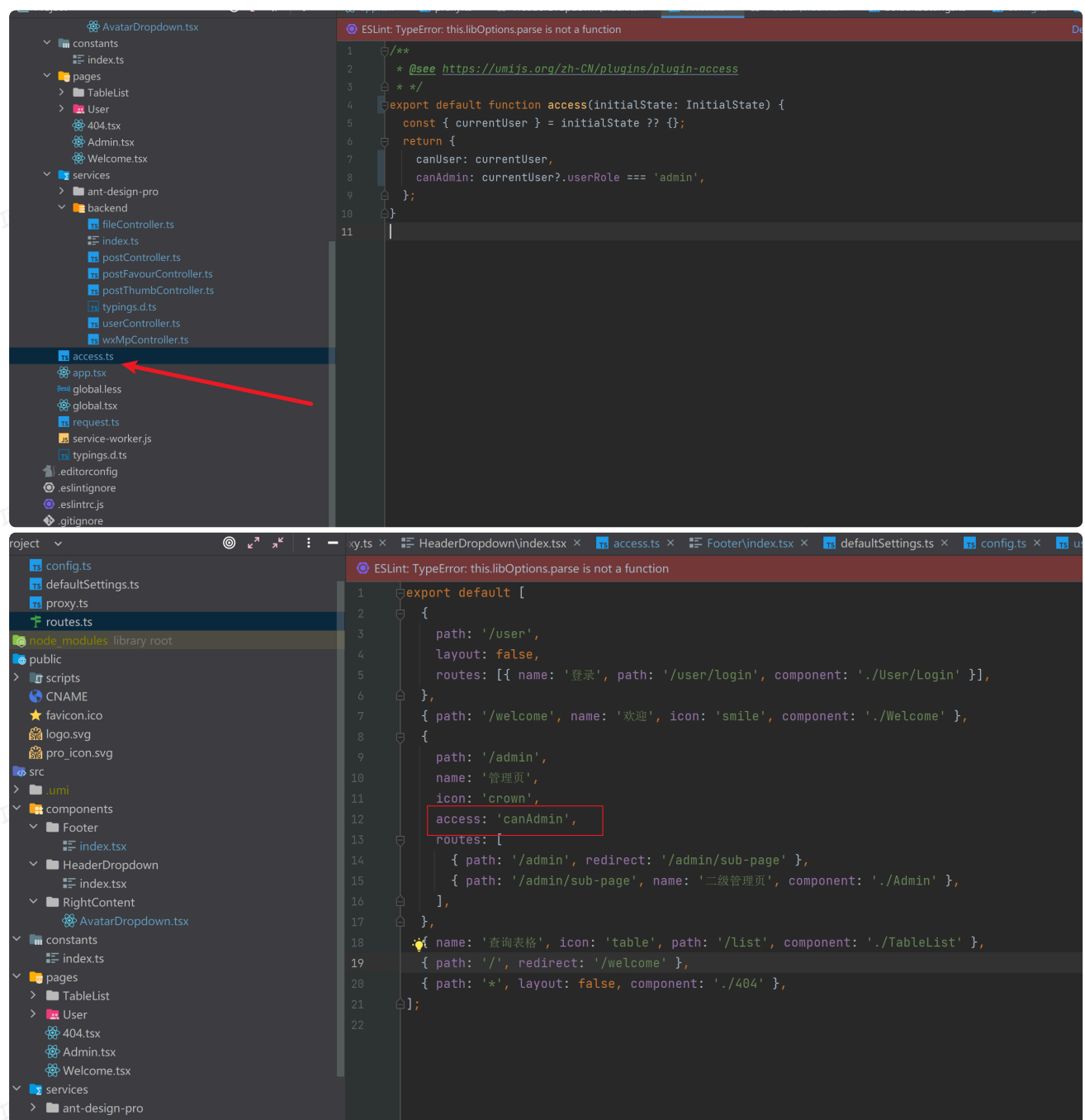
```



权限管理

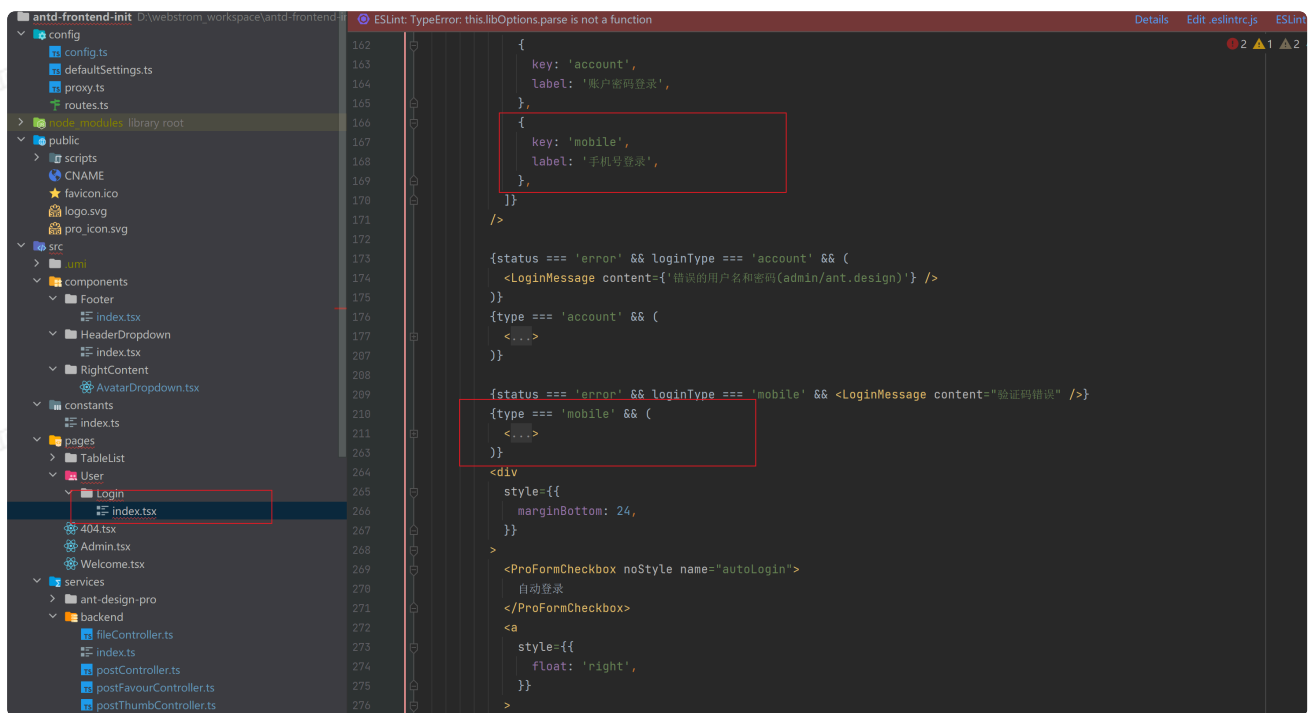
修改 access.ts, 给管理员添加 access: canAdmin

```
1  /**
2   * @see https://umijs.org/zh-CN/plugins/plugin-access
3   * */
4  export default function access(initialState: InitialState) {
5    const { currentUser } = initialState ?? {};
6    return {
7      canUser: currentUser,
8      canAdmin: currentUser?.userRole === 'admin',
9    };
10 }
```

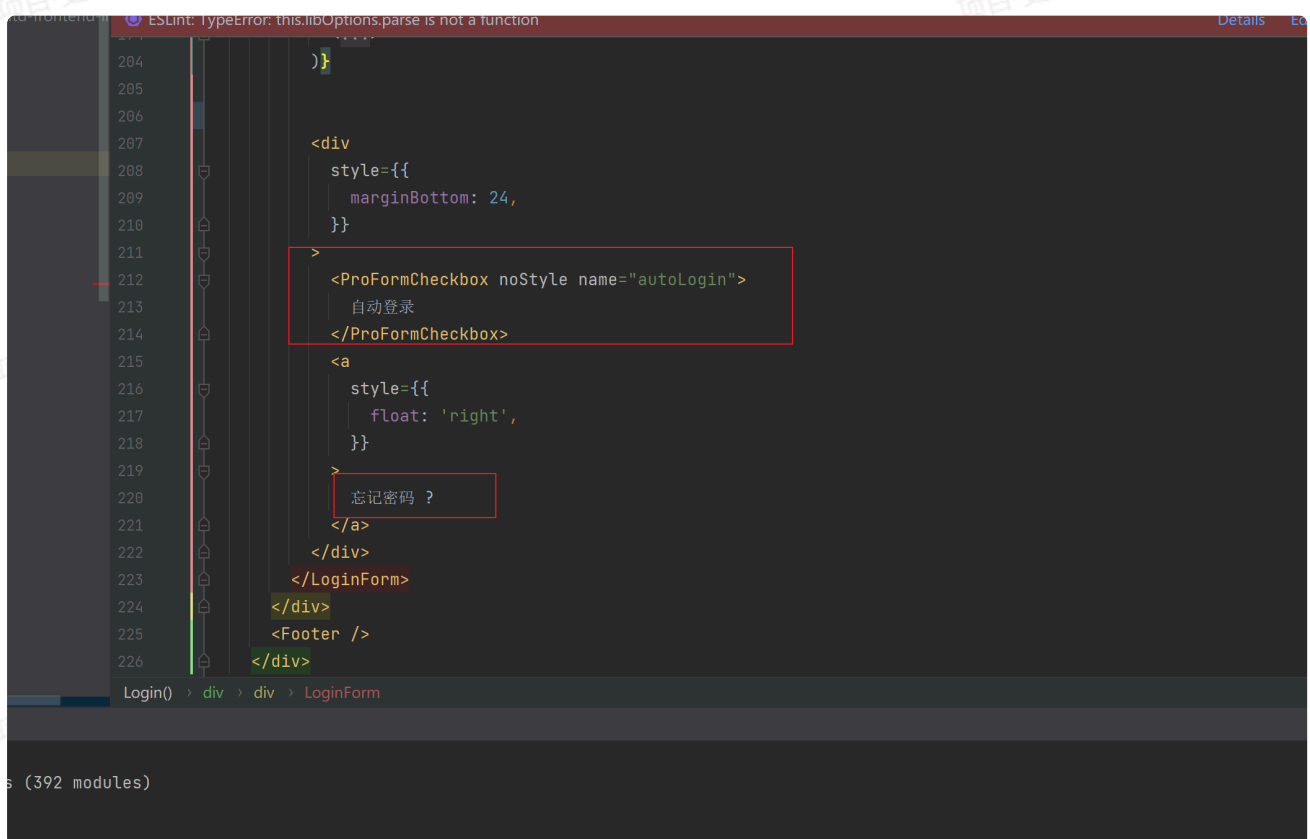


用户登录

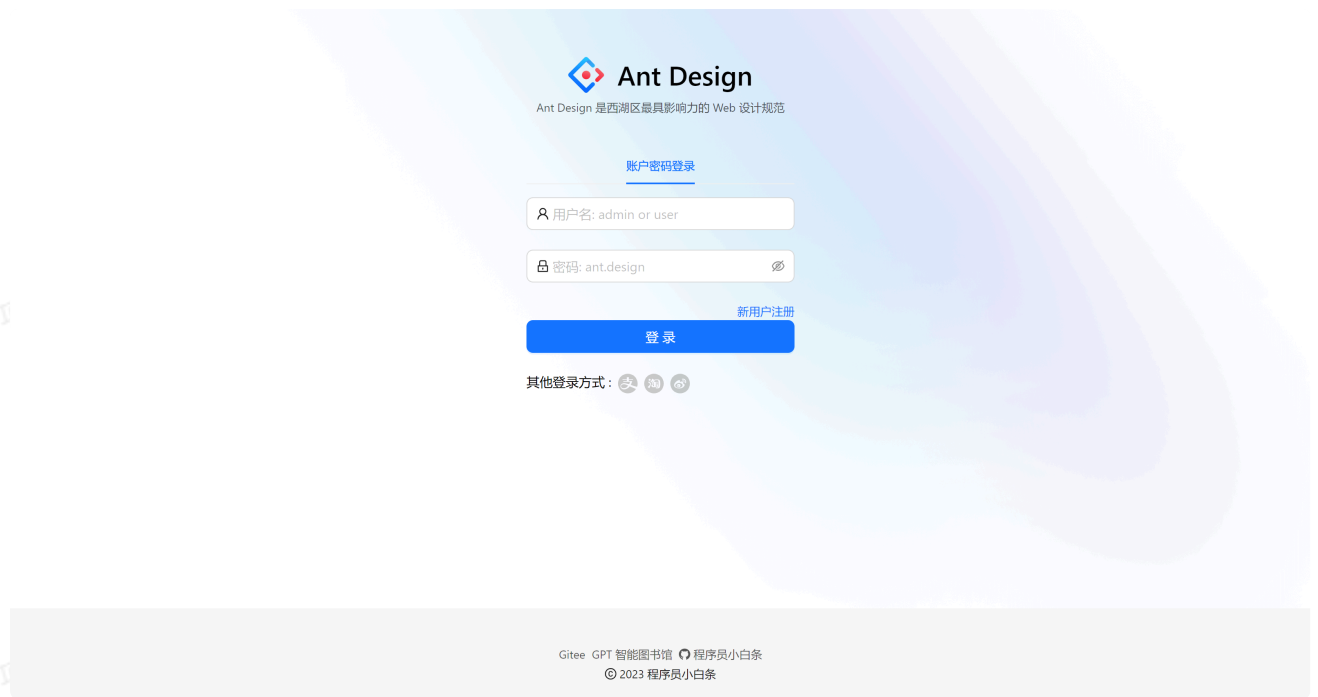
1) 移除手机号登录，验证码错误也可以一并删除



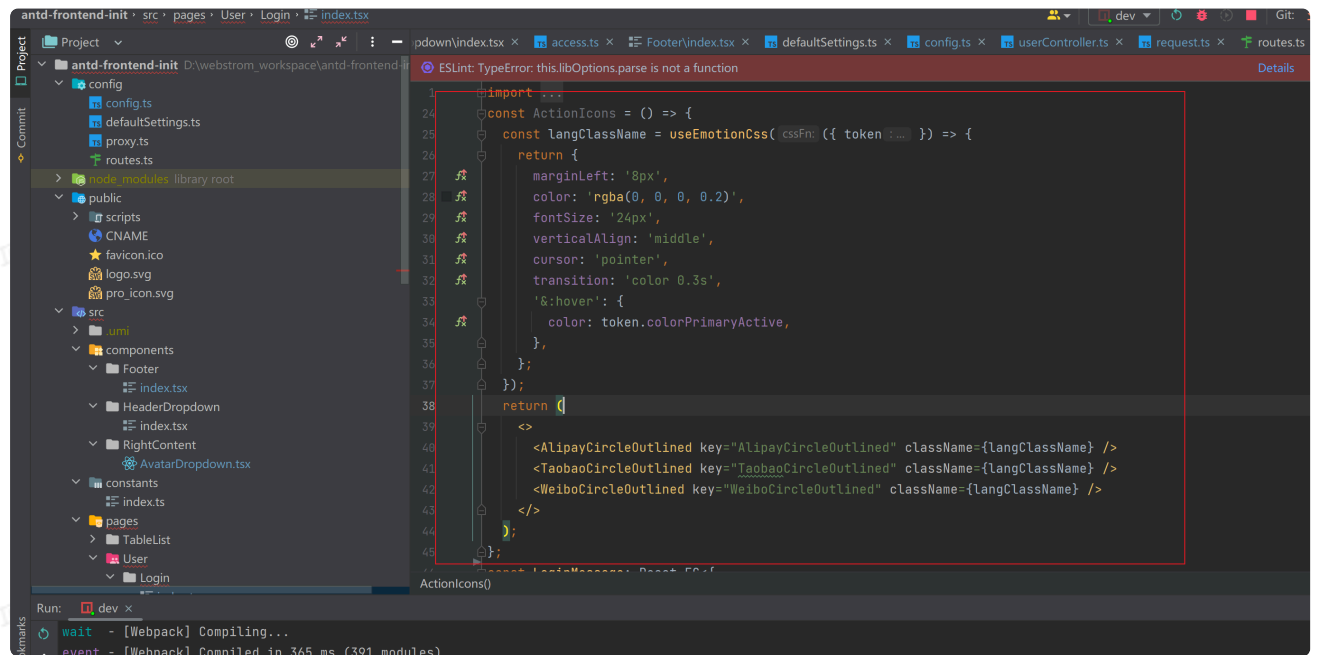
2) 移除自动登录，将忘记密码改为新用户注册



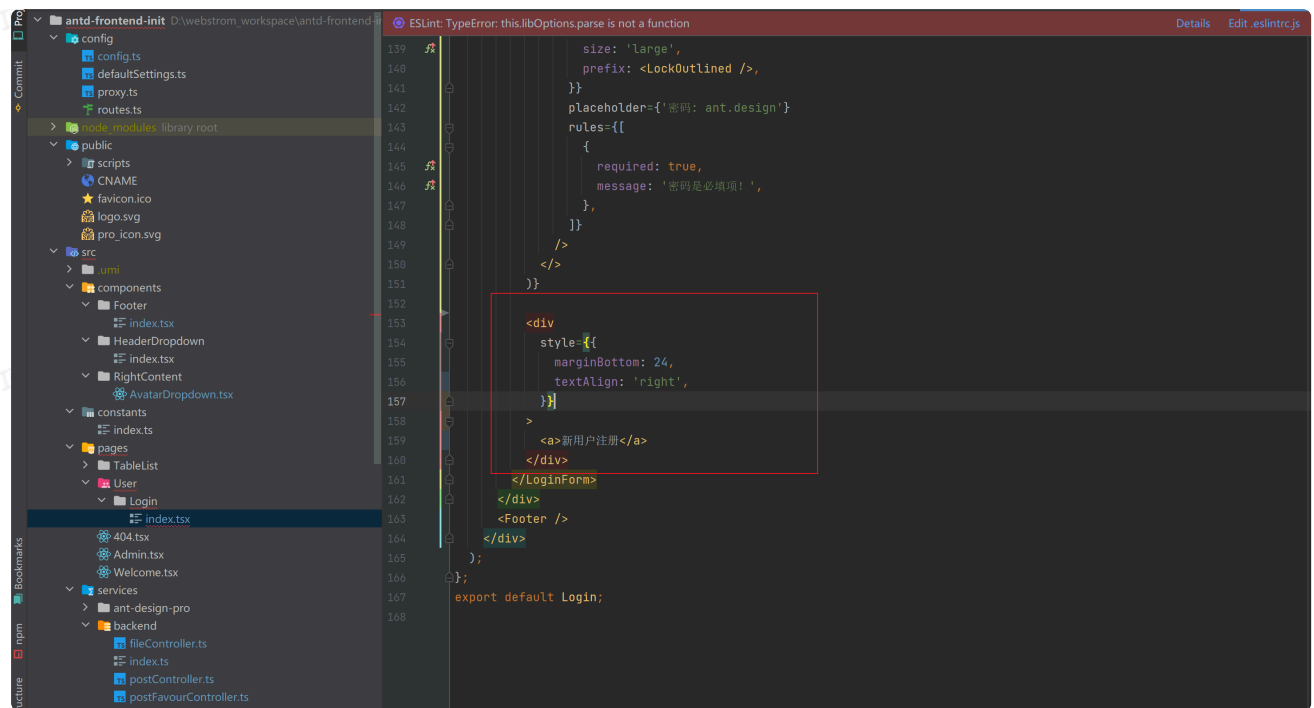
效果图：



3) 移除其他登录方式和一些无用的模块，清除不需要导入的模块和包，按 ctrl + alt + o



4) 调整新用户注册的位置，移除浮动，父标签加 textAlign: right



5) 修改 logo.svg，自己替换即可，将标题和副标题自己替换。

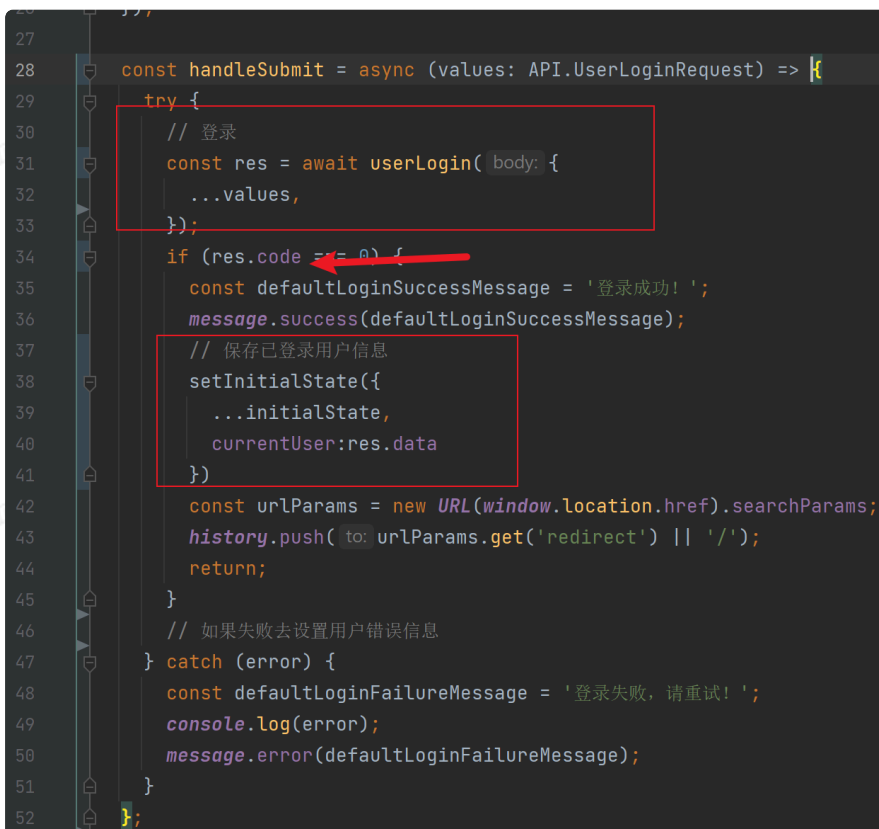


6) 移除错误提示，修改官方的用户名 username 和 密码 password 为 userAccount 和 userPassword


```

1  const handleSubmit = async (values: API.UserLoginRequest) => {
2    try {
3      // 登录
4      const res = await userLogin({
5        ...values,
6      });
7      if (res.code === 0) {
8        const defaultLoginSuccessMessage = '登录成功! ';
9        message.success(defaultLoginSuccessMessage);
10       // 保存已登录用户信息
11       setInitialState({
12         ...initialState,
13         currentUser: res.data
14       })
15       const urlParams = new URL(window.location.href).searchParams;
16       history.push(urlParams.get('redirect') || '/');
17       return;
18     }
19     // 如果失败去设置用户错误信息
20   } catch (error) {
21     const defaultLoginFailureMessage = '登录失败, 请重试! ';
22     console.log(error);
23     message.error(defaultLoginFailureMessage);
24   }
25 };

```

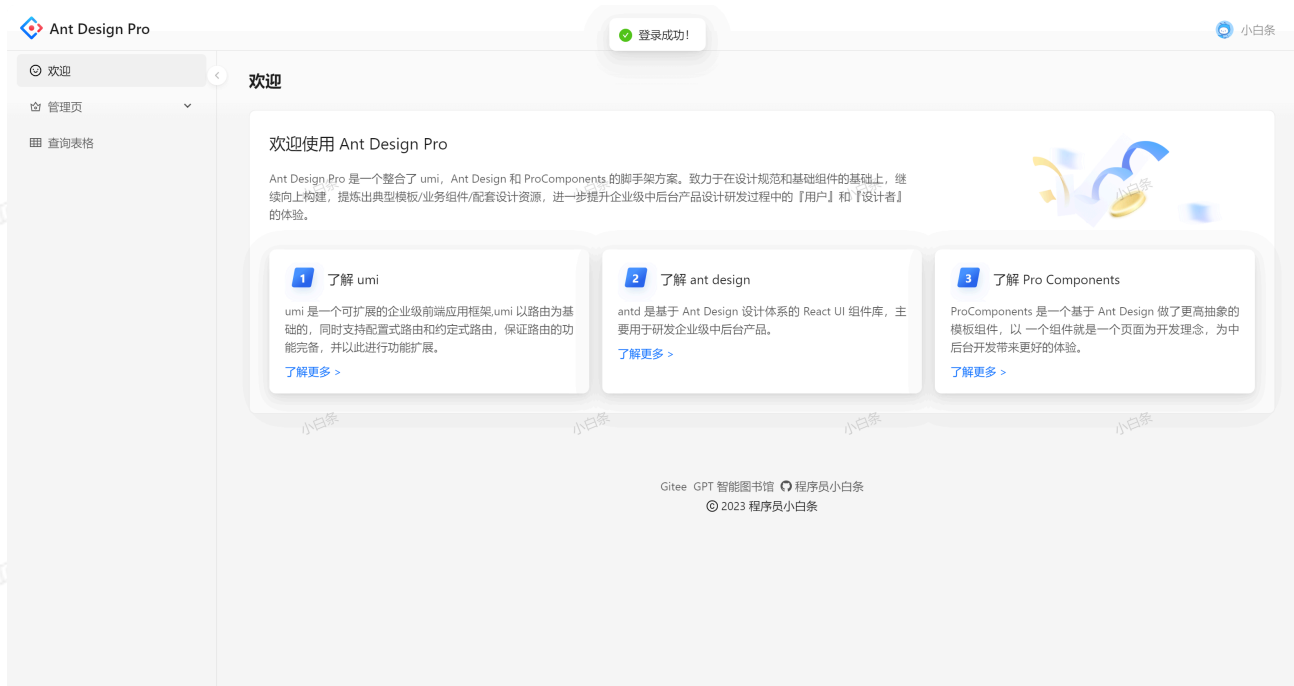


```

27
28  const handleSubmit = async (values: API.UserLoginRequest) => {
29    try {
30      // 登录
31      const res = await userLogin( body: {
32        ...values,
33      });
34      if (res.code === 0) {
35        const defaultLoginSuccessMessage = '登录成功! ';
36        message.success(defaultLoginSuccessMessage);
37        // 保存已登录用户信息
38        setInitialState({
39          ...initialState,
40          currentUser: res.data
41        })
42        const urlParams = new URL(window.location.href).searchParams;
43        history.push( to: urlParams.get('redirect') || '/');
44        return;
45      }
46      // 如果失败去设置用户错误信息
47    } catch (error) {
48      const defaultLoginFailureMessage = '登录失败, 请重试! ';
49      console.log(error);
50      message.error(defaultLoginFailureMessage);
51    }
52  };

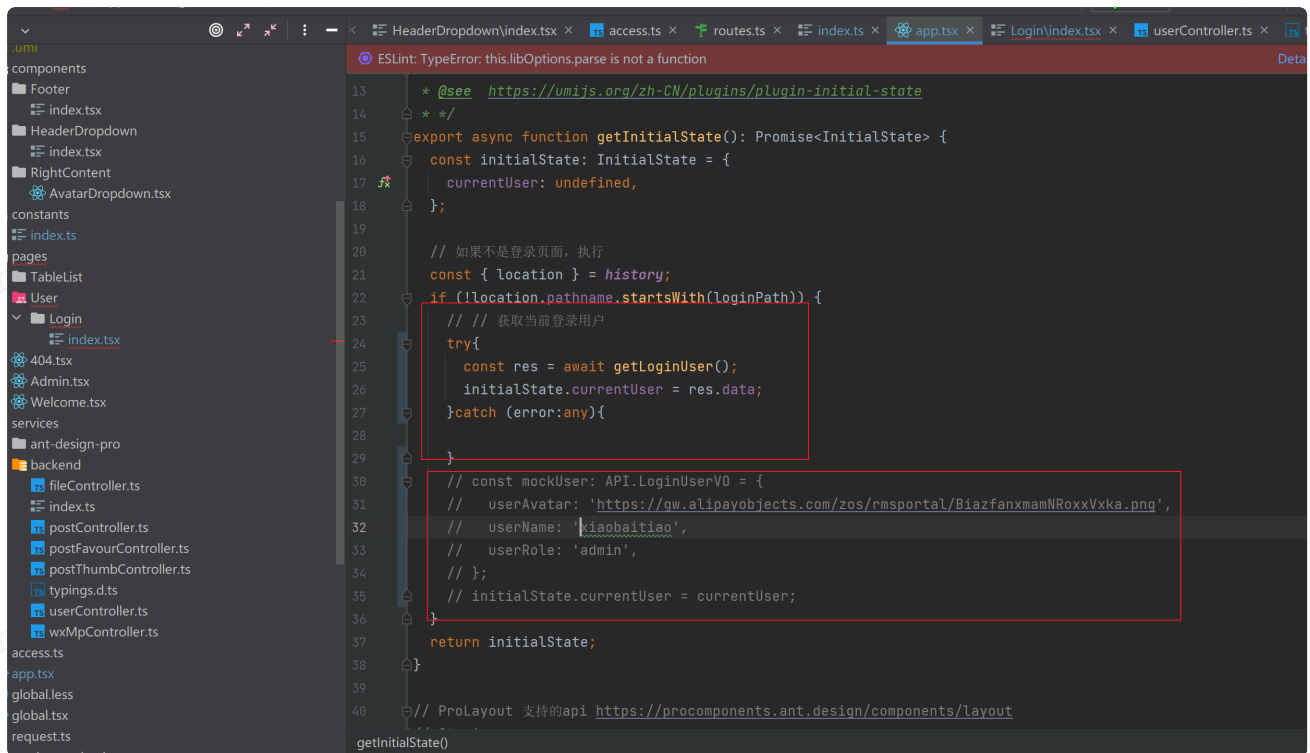
```

9) 输入账号和密码, 与后端数据库的一致即可, 登录成功效果图展示如下



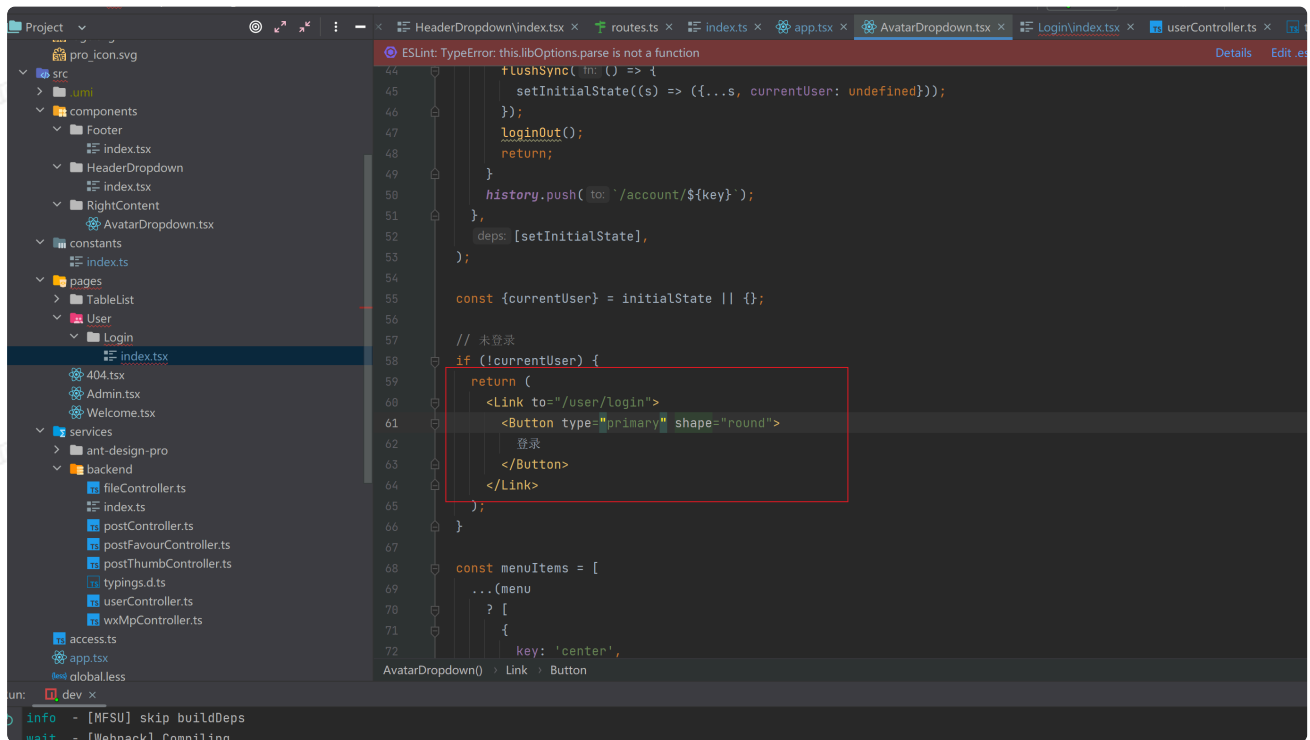
10) 之前由于初始化的时候获取的还是模拟的状态，因此刷新后，用户状态会失效。将 app.tsx 的代码进行修改。

```
1 export async function getInitialState(): Promise<InitialState> {
2   const initialState: InitialState = {
3     currentUser: undefined,
4   };
5
6   // 如果不是登录页面，执行
7   const { location } = history;
8   if (!location.pathname.startsWith(loginPath)) {
9     // // 获取当前登录用户
10    try{
11      const res = await getLoginUser();
12      initialState.currentUser = res.data;
13    }catch (error:any){
14
15    }
16    // const mockUser: API.LoginUserVO = {
17    //   userAvatar: 'https://gw.alipayobjects.com/zos/rmsportal/BiazfanxmamMN
18    //   userName: 'xiaobaitiao',
19    //   userRole: 'admin',
20    // };
21    // initialState.currentUser = currentUser;
22  }
23  return initialState;
24 }
```



11) 修改官方的退出登录接口，修改为我们自己后端的退出登录方法。

12) 给登录按钮添加跳转链接



13) 根据后端的接口返回相对应的错误信息，再次修改 index.tsx 的 handleSubmit 方法

```

1  const handleSubmit = async (values: API.UserLoginRequest) => {
2    try {
3      // 登录
4      const res = await userLogin({
5        ...values,
6      });
7      const defaultLoginSuccessMessage = '登录成功!';
8      message.success(defaultLoginSuccessMessage);
9      // 保存已登录用户信息
10     setInitialState({
11       ...initialState,
12       currentUser: res.data,
13     });
14     const urlParams = new URL(window.location.href).searchParams;
15     history.push(urlParams.get('redirect') || '/');
16     return;
17     // 如果失败去设置用户错误信息
18   } catch (error: any) {
19     const defaultLoginFailureMessage = `登录失败，请重试! ${error.message}`;
20     message.error(defaultLoginFailureMessage);
21   }
22 };

```

错误提示效果图：

14) 修改 ts 类型

用户管理

1) 调整添加节点的代码

```

1  /**
2   * 添加节点
3   * @param fields
4   */
5  const handleAdd = async (fields: API.RuleListItem) => {
6      const hide = message.loading('正在添加');
7      try {
8          await addRule({
9              ...fields,
10             });
11             hide();
12             message.success('创建成功');
13             return true;
14         } catch (error:any) {
15             hide();
16             message.error('创建失败, '+error.message);
17             return false;
18         }
19     };

```

2) 调整更新节点的方法

```

1  /**
2   * 更新节点
3   * @param fields
4   */
5  const handleUpdate = async (fields: FormValueType) => {
6      const hide = message.loading('正在更新');
7      try {
8          await updateRule({
9              name: fields.name,
10             desc: fields.desc,
11             key: fields.key,
12             });
13             hide();
14             message.success('更新成功');
15             return true;
16         } catch (error:any) {
17             hide();
18             message.error('更新失败, '+error.message);
19             return false;
20         }
21     };

```

3) 调整删除节点的代码

```

1  /**
2   * 删除节点
3   * @param selectedRows
4   */
5  const handleRemove = async (selectedRows: API.RuleListItem[]) => {
6      const hide = message.loading('正在删除');
7      if (!selectedRows) return true;
8      try {
9          await removeRule({
10              key: selectedRows.map((row) => row.key),
11          });
12          hide();
13          message.success('删除成功');
14          return true;
15      } catch (error: any) {
16          hide();
17          message.error('删除失败, '+error.message);
18          return false;
19      }
20  };

```

4) 移除获取详情和批量删除的代码

5) 删除其余批量选择和显示详情的代码

6) 修改路由配置，将用户管理移动到管理员的二级路由，将原有的 TableList 文件夹下的 index.tsx 和 components 组件一并移动到 Admin/User 文件夹下。

```

1 export default [
2   {
3     path: '/user',
4     layout: false,
5     routes: [{ name: '登录', path: '/user/login', component: './User/Login' }],
6   },
7   { path: '/welcome', name: '欢迎', icon: 'smile', component: './Welcome' },
8   {
9     path: '/admin',
10    name: '管理页',
11    icon: 'crown',
12    access: 'canAdmin',
13    routes: [
14      { path: '/admin', redirect: '/admin/user' },
15      { name: '用户管理', icon: 'table', path: '/admin/user', component: './Ac
16    ],
17  },
18
19  { path: '/', redirect: '/welcome' },
20  { path: '*', layout: false, component: './404' },
21 ];

```

7) 修改 ProTable 表格，请求我们自己的后台真实接口，修改 TS 类型。

高级组件的官方地址：[ProTable - 高级表格 - ProComponents \(ant.design\)](https://procomponents.ant.design/)

<https://procomponents.ant.design/components/table>

```

1 request={
2   async (params, sort, filter)=>{
3     const sortField = Object.keys(sort)?.[0];
4     const sortOrder = sort?.[sortField]?? undefined;
5     const {data,code} = await listUserByPage({
6       ...params,
7       sortField,
8       sortOrder,
9       ...filter
10    } as API.UserQueryRequest);
11    return {
12      success:code===0,
13      data:data?.records||[],
14      total:Number(data?.total)||0,
15    }
16  }
17 }

```

8) 修改 columns 的配置，让前端的属性值和后端传递的对应即可，调整一下修改和删除的样式。

```
1  const columns: ProColumns<API.RuleListItem>[] = [
2    {
3      title: 'id',
4      dataIndex: 'id',
5      valueType: 'text',
6    },
7    {
8      title: '账号',
9      dataIndex: 'userAccount',
10     valueType: 'text',
11   },
12   {
13     title: '用户名',
14     dataIndex: 'userName',
15     valueType: 'text',
16   },
17   {
18     title: '头像',
19     dataIndex: 'userAvatar',
20     valueType: 'image',
21     fieldProps: {
22       width: 64,
23     },
24     hideInSearch: true,
25   },
26   {
27     title: '简介',
28     dataIndex: 'userProfile',
29     valueType: 'textarea',
30   },
31   {
32     title: '权限',
33     dataIndex: 'userRole',
34     valueEnum: {
35       user: {
36         text: '用户',
37       },
38       admin: {
39         text: '管理员',
40       },
41     },
42   },
43   {
44     title: '创建时间',
45     sorter: true,
46     dataIndex: 'createTime',
47     valueType: 'dateTime',
48     hideInSearch: true,
49   },
50   {
```

```

51     title: '更新时间',
52     sorter: true,
53     dataIndex: 'updateTime',
54     valueType: 'dateTime',
55     hideInSearch: true,
56   },
57   {
58     title: '操作',
59     dataIndex: 'option',
60     valueType: 'option',
61     render: (_, record) => (
62       <Space size={"middle"}>
63         <Typography.Link
64           key="config"
65           onClick={() => {
66             handleUpdateModalOpen(true);
67             setCurrentRow(record);
68           }}
69         >
70           修改
71         </Typography.Link>
72
73         <Typography.Link type="danger" key="subscribeAlert" href="@/pages/Adm
74           删除
75         </Typography.Link>
76
77       </Space>
78     ),
79   },
80 },
81 1.

```

数据展示效果：

9) 添加删除用户的功能

先给链接添加触发事件，传入对应的数据行，方法名称修改一下，handleRemove -> handleDelete

编写对应的删除代码，注意这边 handleDelete 从外面移到了里面，为了使用 actionRef 删除成功后自动页面加载。


```
1  const handleDelete = async (row: API.User) => {
2    const hide = message.loading('正在删除');
3    if (!row) return true;
4    try {
5      await deleteUser({
6        id: row.id,
7      });
8      hide();
9      message.success('删除成功');
10     actionRef?.current?.reload();
11     return true;
12   } catch (error: any) {
13     hide();
14     message.error('删除失败, ' + error.message);
15     return false;
16   }
17   };
```

10) 新增添加用户的功能

首先创建一个 createModal.tsx

编写代码，把 index.tsx 的 handleAdd 移动到此组件即可。

```

1  import { addUser } from '@services/backend/userController';
2  import { ProColumns, ProTable } from '@ant-design/pro-components';
3  import { message, Modal } from 'antd';
4  import React from 'react';
5
6  interface Props {
7    modalVisible: boolean;
8    columns: ProColumns<API.User>[];
9    onSubmit: () => void;
10   onCancel: () => void;
11 }
12
13 /**
14  * 添加节点
15  * @param fields
16  */
17 const handleAdd = async (fields: API.UserAddRequest) => {
18   const hide = message.loading('正在添加');
19   try {
20     await addUser({
21       ...fields,
22     });
23     hide();
24     message.success('创建成功');
25     return true;
26   } catch (error) {
27     hide();
28     message.error('创建失败');
29     return false;
30   }
31 };
32
33 /**
34  * 创建数据弹窗
35  * @param props
36  * @constructor
37  */
38 const CreateModal: React.FC<Props> = (props) => {
39   const { columns, modalVisible, onCancel, onSubmit } = props;
40
41   return (
42     <Modal title={'新建'} open={modalVisible} destroyOnClose footer={null} onC
43     <ProTable<API.UserAddRequest>
44       columns={columns}
45       type="form"
46       onSubmit={async (value) => {
47         const success = await handleAdd(value);
48         if (success) {
49           onSubmit?.();
50         }

```

```

51     }}
52     />
53   </Modal>
54
55   );
56   };
57
58   export default CreateModal;

```

编写是否显示的布尔值

```

1  // 是否显示新建窗口
2  const [createModalVisible, setCreateModalVisible] = useState<boolean>(false);

```

添加 createModal 组件

```

1  <CreateModal
2    modalVisible={createModalVisible}
3    columns={columns}
4    onSubmit={() => {
5      setCreateModalVisible(false);
6      actionRef.current?.reload();
7    }}
8    onCancel={() => {
9      setCreateModalVisible(false);
10   }}
11  />

```

修改点击按钮的事件，让弹窗显示即可

利用 hideInForm 去除不必要显示的属性，比如 id、更新时间、创建时间

新建用户效果图

用户简介由于后端的 UserAddRequest 没有 UserProfile 字段，因此不能新增，但用户可以自己修改，管理员不能去改用户简介。

11) 新增更新用户的功能，先将 index.tsx 的 handleUpdate 的方法剪切到 UpdateModal.tsx 然后做适当修改即可。

```

1  import { updateUser } from '@services/backend/userController';
2  import { ProColumns, ProTable } from '@ant-design/pro-components';
3  import { message, Modal } from 'antd';
4  import React from 'react';
5
6  interface Props {
7    oldData?: API.User;
8    modalVisible: boolean;
9    columns: ProColumns<API.User>[];
10   onSubmit: () => void;
11   onCancel: () => void;
12 }
13
14 /**
15  * 更新节点
16  *
17  * @param fields
18  */
19 const handleUpdate = async (fields: API.UserUpdateRequest) => {
20   const hide = message.loading('更新中');
21   try {
22     await updateUser(fields);
23     hide();
24     message.success('更新成功');
25     return true;
26   } catch (error: any) {
27     hide();
28     message.error('更新失败, ' + error.message);
29     return false;
30   }
31 };
32
33 /**
34  * 更新数据弹窗
35  * @param props
36  * @constructor
37  */
38 const UpdateModal: React.FC<Props> = (props) => {
39   const { oldData, columns, modalVisible, onCancel, onSubmit } = props;
40
41   if (!oldData) {
42     return <></>;
43   }
44
45   return (
46     <Modal title={'更新'} open={modalVisible} destroyOnClose footer={null} onCancel={onCancel}>
47       <ProTable<API.UserUpdateRequest>
48         columns={columns}
49         form={{
50           initialValues: oldData,

```

```

51   }}
52   type="form"
53   onSubmit={async (values) => {
54     const success = await handleUpdate({
55       ...values,
56       id: oldData.id,
57     });
58     if (success) {
59       onSubmit?.();
60     }
61   }}
62   />
63 </Modal>
64
65 );
66 };
67
68 export default UpdateModal;

```

12) 添加显示更新窗口的一个布尔值

```

1  // 是否显示更新窗口
2  const [updateModalVisible, setUpdateModalVisible] = useState<boolean>(false);

```

13) 更新用户行 TS 类型

修改对应的 Click 事件

添加组件

```

1    <UpdateModal
2      modalVisible={updateModalVisible}
3      columns={columns}
4      oldData={currentRow}
5      onSubmit={() => {
6        setUpdateModalVisible(false);
7        setCurrentRow(undefined);
8        actionRef.current?.reload();
9      }}
10     onCancel={() => {
11       setUpdateModalVisible(false);
12     }}
13   />

```

更新效果图：

其他

1) 修改 title 和 logo

2) 修改 footer 的背景色为 none

3) BUG 修复：如果当前用户为管理员，路径为 admin/user，注销后再次登录，会发现无权限。将重定向取消后即可修复 BUG，这段代码在 AvatarDropdown.tsx 中。

📄 <https://se

url=https%3A%2F%2Fwww.yuque.com%2Fu37765561%2Fak85bt%2Fb184b8fa98ecac1b50f

%E5%89%8D%E7%AB%AF%E4%B8%87%E7%94%A8%E6%A8%A1%E6%9D%BF%E5%BC%80%E5%