1. 前端框架选择

目前主流前端框架为AngularJS、React和Vue.js。由于种种原因首先排除掉AngularJS。

React 和 vue.js 都是组件式ui开发框架，并且都具有完备的开发生态

React 全家桶：React + (Redux + Redux中间件) + React-Router

Vue 全家桶： Vue + vuex + vue-router

最终我们选择 Vue.js 原因如下：

上手难度

Column A	Column B
前端框架	React == vue
状态管理	Redux > Vue
路由跳转	react-router == vue-router

文档对比

在天朝，文档方面 vue 完虐 react...

其他

流行的代码检查工具和测试框架都有对应的版本。 公司并没有配置针对.vue 文件的静态代码审查工具，但是我们依然要根据规范来编写代码

2. 网络请求

网络请求工具决定使用axios.js, 原因如下：

1. 不使用jquery ajax 模块是因为没必要为了一个模块来引入整个jquery库；
2. 不适用 fetch 的原因是一些浏览器或者mock工具还不支持
3. axios.js 基于ajax，支持 promise 调用，上手不会有难度；
4. axios.js 针对 csrf 有优化，且提供请求拦截器功能，方便在请求前后做权限校验。

缺点：不支持jsonp的方式跨域。 解决方法：使用 CORS 方式跨域。

```
// 添加请求前的各种验证
Axios.interceptors.request.use(
    config => {
        if (
            config.method === 'post' ||
            config.method === 'put' ||
            config.method === 'delete'
        ) {
```

```
            // 序列化
            config.data = qs.stringify(config.data);
        }
         if (localStorage.token) {
             config.headers.Authorization = localStorage.token;
         }

        return config;
    },
    error => {
        return Promise.reject(error);
    }
);
//添加请求之后各种验证
Axios.interceptors.response.use(
    res => {
        if(res.status != 200) {
            throw...
        }
        //if (res.data.token) {
        //    console.log(res.data.token);
        //}
        return res;
    },
    error => {
        return Promise.reject(error);
    }
);
```

3. 权限校验

需要权限访问的页面和功能组件需要异步加载，在代码中编写异步组件，webpack打包时会自动使用代码分割功能将异步组件生成为单独的js文件。 3.1. 页面路由跳转权限： 1. 登录之后获取功能权限列表 2. 根据权限列表，动态生成导航菜单和Routers 3. 在跳转页面前判断本地是否有token以及页面权限，检查不通过这跳转到login

```
    router.beforeEach((to, from, next) => {
        if (to.meta.requireAuth) {  // 判断该路由是否需要登录权限
            if (token) {  // 判断token是否存在
                next();
            }
            else {
                next({
                    path: '/login',
                    query: {redirect: to.fullPath}  // 将跳转的路由path作为参数，登录成功后跳转到该路由
                })
```

```
            }
        }
        else {
            next();
        }
    })
```

3.2. 功能权限
    1. 登录之后获取功能权限
    2. 加载页面之后通过功能权限列表判断是否异步加载功能组件

异步组件示例：

```
    <template>
        ...
        <new-btn v-if='permissionList.newBtn'/>
        ...
    </template>
    <script>
        import { HandleContainer, AnchButton } from 'components/basic'; //导入同步组件
        import SearchBar from 'components/basic/SearchBar';
        export default {
            name: 'page1',
            components: {
                SearchBar,
                HandleContainer,
                AnchButton,
                newBtn(resolve) { //导入异步组件
                    require(['./buttons/newBtn'],resolve)
                }
            },
        }
    </script>
```

4. ui框架

选用 element-ui，提供了大多数需要的UI组件，可以尽量节省出编写 UI 代码的时间用于业务代码，提高效率。