**步骤：**

1. Vue-cli 3.0 创建项目 vue create 项目名称
2. 安装iview依赖
3. 修改main.js文件和app.vue文件
4. 配置路由
5. 创建store库
6. 创建登录页面，实现跳转
7. 创建main.vue进行布局，按区域划分组件（顶部、左侧导航、标签）
8. 通过路由获取菜单列表，通过组件显示在页面上
9. 顶部导航，通过退出跳转到登录页，点击修改密码跳转到弹出修改密码框
10. 标签页，存储标签数据，每次新增标签时，需要判断是否标签列表中已存在，如果存在该标签页高亮显示，不新增标签，删除标签，通过判断删除的当前标签与标签列表中的相等项进行删除，并且通过删除项找到标签列表下一路由进行高亮显示，删除其他和全部，获取标签列表中与当前项或者首页相同的项，通过处理重新赋值给标签列表数据
11. 路由拦截

**界面规划：**

1. 登录页
2. 顶部通栏：顶部为通栏，左侧为导航，右侧为用户名称及用户操作（退出和修改密码），并且顶部是固定的。
3. 左侧导航栏：左侧导航为二级导航菜单栏
4. 内容块：内容块分为了两个小部分，标签导航（上）和内容页（下）

**菜单实现：**

1. 创建vue文件（项目中的页面），并在router.js中进行配置，记得export default导出
2. 通过路由列表获取菜单列表（通过判断路由里的meta、meta.hideInMenu和meta.children来获取），在store中getter内通过方法获取到菜单列表并存储在getter。
3. This.$store.state.getter.menuList获取到存储在store内的共享数据menuList
4. 根据菜单列表中的children来判断是否显示在左侧导航，通过判断children里的个数（当children里的个数大于1时，为二级导航，当个数等于1时，为一级导航）判断是否为二级导航或一级导航。

**标签页：**

1. 新增标签前要判断当前路由是否已经在标签列表中了，是否为新增标签，如果当前标签列表中没有与当前路由相同的项，则当前标签列表新增当前路由并标签生成新的标签高亮显示为当前页，有则不新增，页面跳转到当前页，当前路由与标签列表各项进行对比，相同的一项及为当前标签并高亮显示。
2. 删除标签页，通过删除页与标签列表页对比进行删除。

**路由拦截：**

1. 登录成功后接口会返回token值，返回的token值需要存储到cookie中
2. 获取存储在cookie中的token，路由跳转前，如果token没有值，并且跳转的页面不是登录页，则跳转到登录页；如果token没有值，并且跳转的页面是登陆页，则跳转到登录页，如果token有值并且跳转的页面是登录页，则跳转到首页
3. 判断是否获取到用户信息，如果获取到用户信息判断是否有权限并跳转，如果没有获取到用户信息重新调用获取用户信息接口重新获取用户信息，如果调用接口失败，则返回登录页，重新登录。

**知识点：**

1. push() 方法可以在数组的末属添加一个或多个元素
2. pop() 方法把数组中的最后一个元素删除
3. shift() 方法把数组中的第一个元素删除
4. unshift() 方法可以在数组的前端添加一个或多个元素
5. some() 方法是用于检测数组中的元素是否有满足指定条件，如果有一个元素满足条件，则表达式返回true，剩余的元素不会再执行检测；如果没有妈祖的条件的原色，则返回false
6. filter() 创建一个新数组，新数组中的元素是通过检测原数组中符合条件的所有元素

filter() 不会对空数组进行检测，也不会改变原数组

1. map() 返回一个新数组，数组中的元素为原数组元素条用函数处理后的值

map()不会对空数组进行检测，也不会改变原数组

1. findIndex() 获取元素的在数组中的第一个索引位置
2. 本地存储localStorage(长期存储)：

存值localStorage.setItem(“name”, value)

取值localStorage.getItem(“name”, value)

本地存储sessionStorage（临时存储）过期时间就是关闭浏览器，是个临时会话窗口

Cookie空间太小

1. Vuex: state getter mutations actions modules

a. State: 存储数据，提供唯一的公共数据源

b. Getters: 存储数据，可以通过调用函数方法获取值并存储在getter中

c. Mutations: 这里边的方法是唯一可以改变state里的数据的方法，对state里数据赋值之类的操作，都是在这里进行的，mutations里的代码都是同步的，mutations类似于自定义事件

d. Actions: actions与mutations类似涉及到的是异步操作

11. 导航守卫：router.beforeEach((to, from, next) => {})

a. to:即将要进入的目标路由

b. from:当前导航正要离开的路由

c. next: Function: 一定要调用该方法来 **resolve** 这个钩子。执行效果依赖 next 方法的调用参数。