总体的思路：

1.把原先写死的路由放到数据库

2.通过后台方法包括递归等返回对应的基本上和之前写死一样的数据和结构（可能会有细微的差别）

3.前端获取通过改造router下面的index.js文件，获取后端路由，并且拼接到现在的路由上，核心的代码：

```
// 路由加载方法
export async function loadRoutes(role) {
  // 获取菜单列表
  const
    menuList
   = await queryRolePermission({
        role: role[0]})
  console.log("menuList", menuList)
  const routes = formatRoutes(menuList.Entity.treeList);
  console.log("routes", routes);
  return routes;
}
```

```
const formatRoutes = (data) =>{
  const res = [];
  data.forEach(item => {
    const tmp = {
      ...item
    };
    if (tmp.Children) {
      tmp.children = formatRoutes(tmp.Children);
    }
    let route = {
      path: tmp.Path,
      //这个是一个关键点，因为webpack编译的时候，动态载入的路由通过import的方式会不能识别，所以一定要用
      //  这种写法  resolve => require(['@/' + tmp.Component.replace('@/', '').replace('.vue', '') + '.vue'], resolve)
```

```
      component: tmp.Component === 'Layout' ? Layout : resolve => require(['@/' + tmp.Component.replace('@/', '').replace('.vue', '') + '.vue'], resolve),

      name: tmp.Name,
      meta: {
        title: tmp.Meta_Title,
        icon: tmp.Meta_Icon
      },
      children: tmp.children || [],
      hidden: tmp.Hidden
    };
    // 如果tmp.ParentId为null，添加redirect属性
    if (tmp.ParentId === null) {
      route.redirect = tmp.Path;
    }

    res.push(route);
  });
  return res;
};
```

4.测试，测试是软件开发永恒的话题，也是最基本上的构成，如果没有测试，软件就是不可靠的，是不能称之为真正的软件的

重要关键点如下：

1.vue里面登录会内置角色，我们的角色一定要是固定的，然后登录系统的角色来通过其他的字段来赋值即可

```
{
    url: '/vue-element-admin/user/info\.*',
    type: 'get',
    response: config => {
      const { token } = config.query
      const userinfo = JSON.parse(token)
      console.log('userinfo', userinfo)
      const info = users[userinfo.role + userinfo.EnterprisesType]

```

```
      console.log('政府主管部门', info, users, userinfo)
      // 如果是主管部门,加上自定义角色角色
      if (userinfo.role === 'ZG') {
        if (!info.roles.includes(userinfo.usertype)) {
          info.roles.unshift(userinfo.usertype);
        }
      }

      // mock error
      if (!info) {
        return {
          code: 50008,
          message: 'Login failed, unable to get user details.'
        }
      } else {
        info.userid = userinfo.userid
        info.fullname = userinfo.fullname
        info.unitcode = userinfo.unitcode
        info.usertype = userinfo.usertype
        info.avatar = userinfo?.avatar
      }
      return {
        code: 20000,
        data: info
      }
    }
```

2.后端的主要代码如下

```
public Task<MenuDto> QueryAllTreeListForRole()
        {
            try
            {
                var rep = this.commonRepositoryFactory.BaseRepository();
                var menus = rep.FindList<MenuModel>().Result.ToList();
                // 可能需要排序或者过滤 转换为树形结构（此处应按需实现）
                var treeList = TransformToTreeList(menus);
                var ret = new MenuDto { treeList = treeList, ids = menus.Select(x => x.
Id).ToList() };
```

```
                return Task.FromResult(ret);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                return default;
            }


        }

        public Task<MenuDto> QueryRolePermission(string role, string userName = "")
        {
            try
            {
                var rep = this.commonRepositoryFactory.BaseRepository();
                var all = rep.FindList<MenuModel>().Result.ToList();
                var menus = all.Where(o => o.RoleCode.Split(',')
                          .Select(x => x.Trim().ToUpper())
                          .Contains(role.Trim().ToUpper()))
                .ToList();


                // 可能需要排序或者过滤 转换为树形结构（此处应按需实现）
                var treeList = TransformToTreeList(menus);
                var ret = new MenuDto { treeList = treeList, ids = menus.Select(x => x.Id).ToList() };
                return Task.FromResult(ret);
            }catch(Exception ex)
            {
                Console.WriteLine(ex.Message);
                return default;
            }


        }

        private List<MenuModel> TransformToTreeList(List<MenuModel> menus, string parentId = null)
```

```
        {
            var treeList = new List<MenuModel>();
            foreach (var menu in menus.Where(m => m.ParentId == parentId).OrderBy(m => m.Sort))
            {
                menu.Children = TransformToTreeList(menus, menu.Id);
                treeList.Add(menu);
            }
            return treeList;
        }

        public async Task<bool> SaveRolePermission(string role, string permissionIds)
        {
            try
            {
                var rep = this.commonRepositoryFactory.BaseRepository();

                // 获取所有有效的RoleCode
                //var allValidRoleCodes = (await rep.FindList<RoleModel>()).Select(r => r.RoleCode).ToList();

                var menus = (await rep.FindList<MenuModel>()).ToList();
                string[] permissionIdArray = permissionIds.Split(",");

                foreach (var menu in menus)
                {
                    // 分割角色代码，判断是否为null或空字符串
                    string[] roles = (menu.RoleCode ?? "").Split(new[] { ',' }, StringSplitOptions.RemoveEmptyEntries);

                    // 删除无效的RoleCode
                    //roles = roles.Where(r => allValidRoleCodes.Contains(r)).ToArray();


                    // 判断是否包含指定角色
                    bool containsRole = roles.Contains(role);

                    // 判断是否在权限ID列表中
                    bool inPermissionList = permissionIdArray.Contains(menu.Id);
```

```
83
84                      if (inPermissionList)
85                      {
86                          if (!containsRole)
87                          {
88                              // 添加角色
89                              roles = roles.Append(role).ToArray();
90                          }
91                      }
92                      else
93                      {
94                          if (containsRole)
95                          {
96                              // 移除角色
97                              roles = roles.Where(r => r != role).ToArray();
98                          }
99                      }
100
101                     // 重新拼接角色代码
102                     menu.RoleCode = String.Join(",", roles);
103                 }
104
105                 // 批量更新数据库
106                 await rep.UpdateEx(menus);
107
108                 return true;
109             }
110             catch (Exception ex)
111             {
112                 Console.WriteLine(ex.Message);
113                 return false;
114             }
115         }
```