4 系统实现

4.1 基础模块实现

4.1.1 认证功能实现

认证功能实现如图4.1。用户或管理员在登录界面填写登录表单，点击登录后提交登录表单进行认证，认证失败会显示认证失败的信息，成功则跳转到私家车位共享平台APP后端系统的首页。



图4.1 认证功能示例图

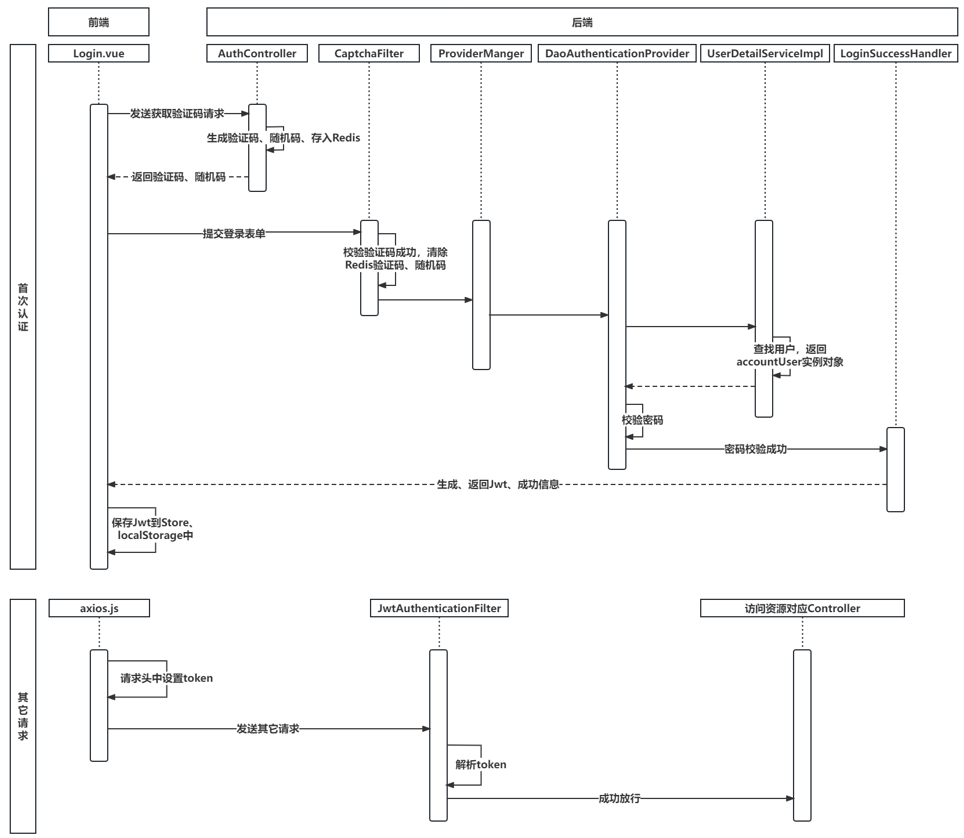
认证功能时序图如图4.2。进入登录界面时，由Login.vue发送获取验证码请求到后端，AuthController生成5位数验证码与32位随机码，存入缓存Redis中并返回Login.vue显示生成的验证码。填写登录表单后，Login.vue提交登录表单到Security过滤链中，首先经过自定义的验证码过滤器，校验验证码成功后清除Redis中的验证码、随机码，放行到过滤链下一环。经过Security自带的ProviderManger、DaoAuthenticationProvider后在实现的UserDetailServiceImpl中在数据库查找用户并返回accountUser实例对象到DaoAuthenticationProvider中。通过getUsername()、getPassword()、isAccountNonLocked()、isEnabled()方法对用户名、密码、用户状态、角色状态进行校验，校验成功则跳转到认证成功处理器LoginSuccessHandler。在LoginSuccessHandler中生成返回Jwt，并返回成功信息，最后由Login.vue将Jwt保存到Vuex.Store以及localStorage中。对其它请求，前端的axios.js会统一在请求头中设置token交由JwtAuthenticationFilter进行解析，校验成功则访问目标资源，失败则跳转到AuthenticationEntryPoint。

图4.2 认证功能时序图

其中，UserDetailServiceImpl在数据库查找用户并返回accountUser实例对象的代码如下，AccountUser是根据系统实现的Security自带的UserDetails接口类。限于篇幅仅展示代码核心部分，中间校验用户是否存在、状态等过程代码不过多展示。

public UserDetails loadUserByUsername(String username) {

User user = userService.getByUsername(username);

……//校验过程

AccountUser accountUser = new AccountUser(

user.getId(),user.getUsername(),user.getPassword(),

enabled,true,true,accountNonLocked,

getUserAuthority(user.getId()));

return accountUser;

}

4.1.2 授权功能实现

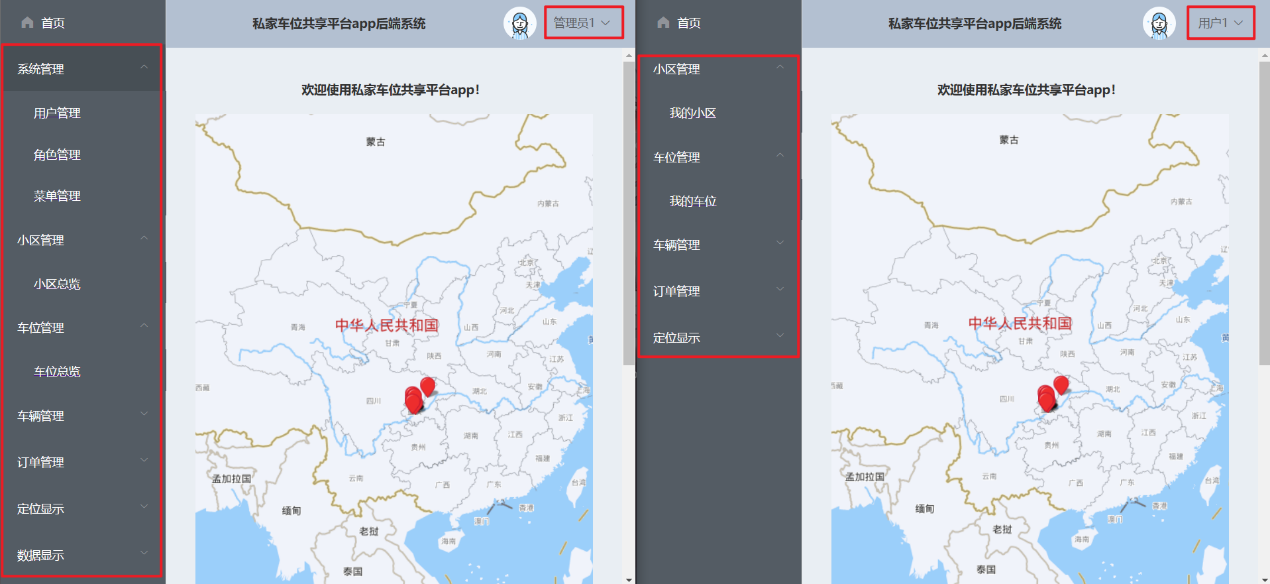
授权功能实现如图4.3。用户和管理员在首次授权后获得与权限相对应的菜单，对权限外的菜单没有访问权限。

图4.3 授权功能示例图

授权功能时序图如图4.4。首次授权在用户或管理员登录时完成，在上一节认证功能的基础上多增加了getUserAuthority()方法返回权限添加到accountUser实例对象中，并存储到Redis中。在调用接口时，前端后端对权限进行了双重防范，前端目录、菜单、按钮等通过绑定hasAuth()方法将功能与权限进行绑定，无权限功能将不会展示；后端则通过@PreAuthorize("hasAuthority('...')")注解将功能与权限进行绑定，无权限则无法访问对应的接口。在拥有对应权限的情况下，前端才会显示功能按钮、后端才会放行接口，最后返回成功信息，授权失败则返回失败信息。

其中getUserAuthority()方法的调用在上一节accountUser实例对象的创建返回中已有体现，该方法内部又调用了UserService下的getUserAuthorityInfo()方法，二者间仅进行了结构转换。下面展示getUserAuthorityInfo()的核心代码部分。

public String getUserAuthorityInfo(Long userId) {

String authority="";

if (redisUtil.hasKey("GrantedAuthority:" + user.getUsername())) {

……//判断是否缓存了权限信息，缓存过则直接取出

}else{

……//否则取出存入redis

}

return authority;

}

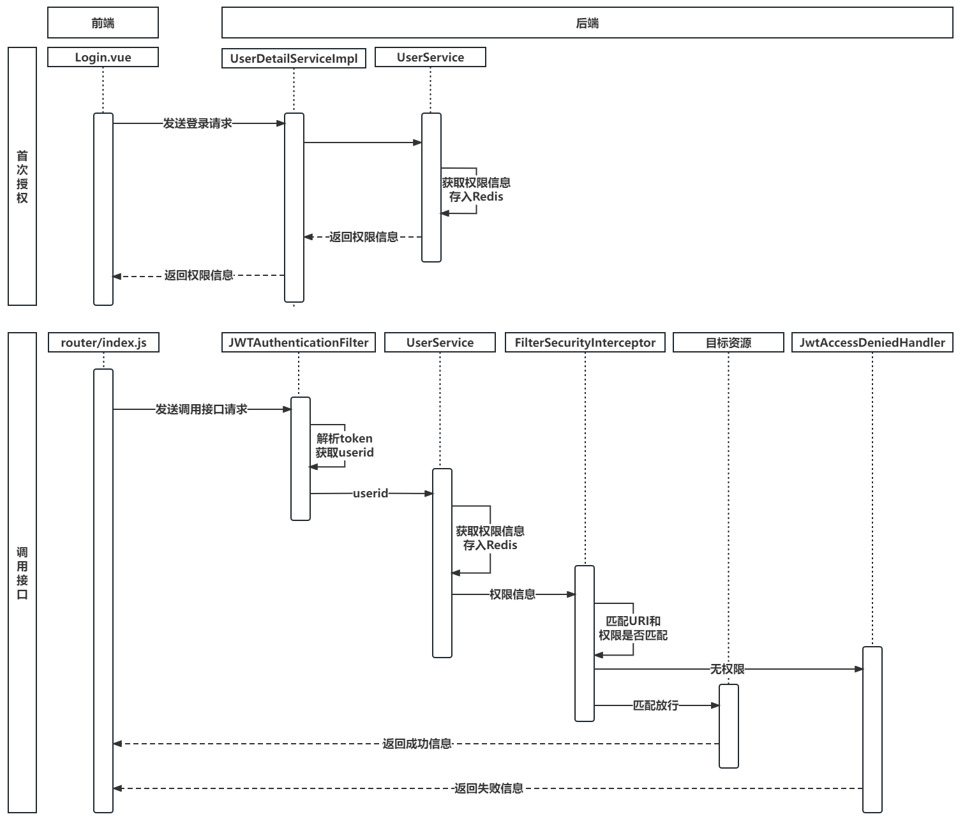


图4.4 授权功能时序图

4.1.3 注册功能实现

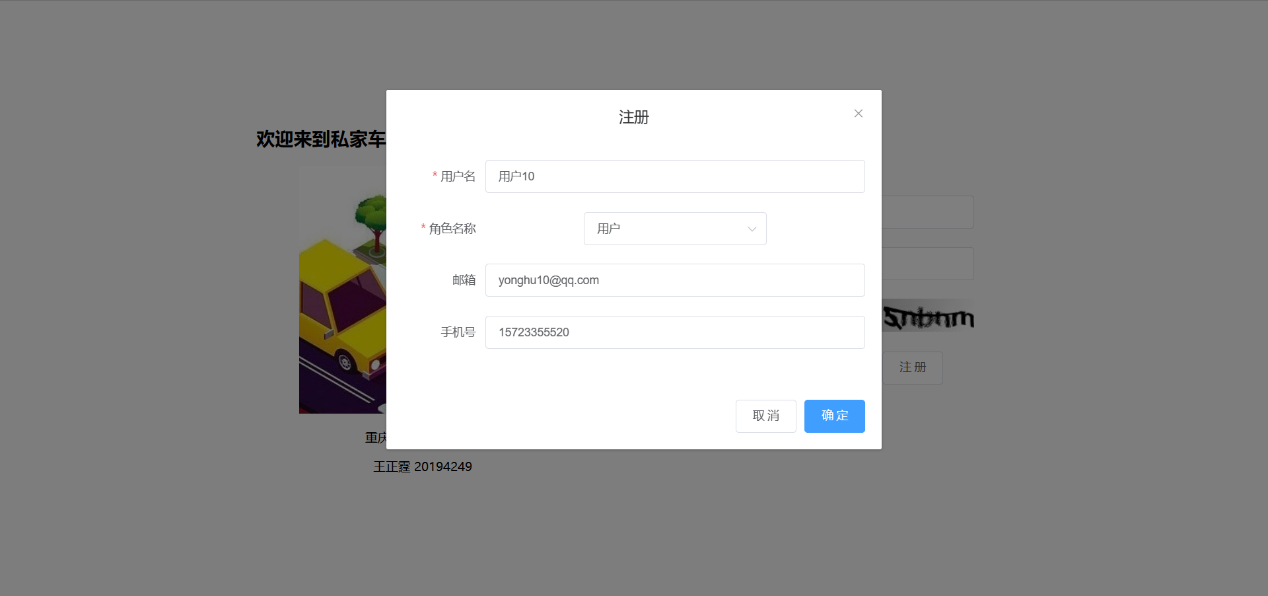
注册功能实现如图4.5。点击注册后填写注册表单校验确定后返回成功信息。

图4.5 注册功能示例图

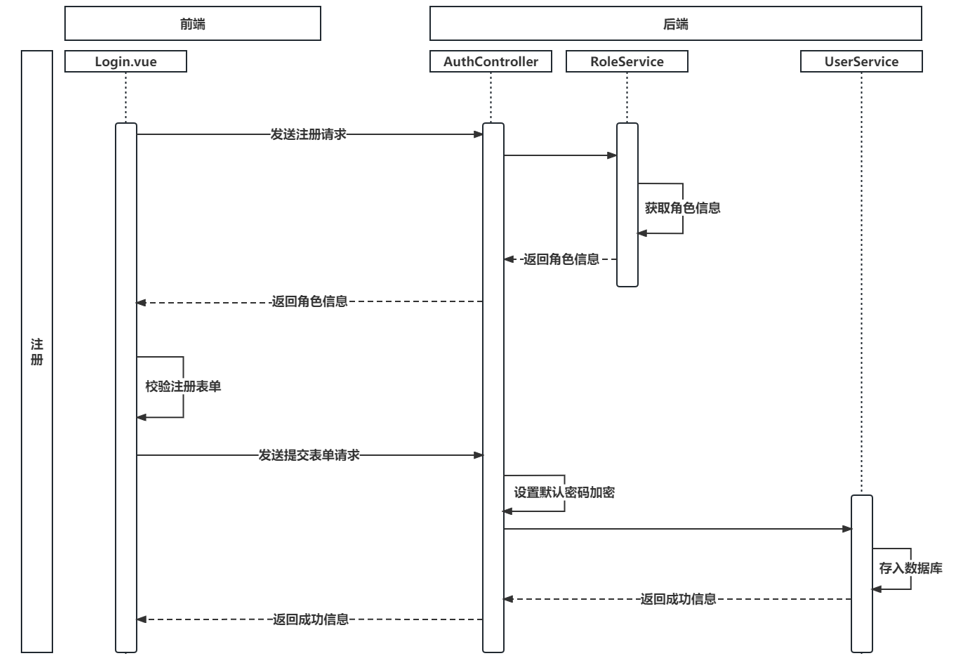
注册功能时序图如图4.6。由Login.vue发送注册请求到AuthController，调用RoleService获取角色信息返回到前端注册表单可供选择，填写注册表单校验后提交，由AuthController设置默认密码并加密，将注册的用户信息通过UserService创建为新的User对象并存储到数据库中，最后返回成功信息。

图4.6 注册功能时序图

其中，注册方法的核心代码如下所示。

public Result logon(@Validated @RequestBody User user) {

// 设置默认密码

String password = passwordEncoder.encode(Const.DEFULT\_PASSWORD);

user.setPassword(password);

userService.save(user);

return Result.succ(user);

}

4.1.4 修改密码功能实现

修改密码功能实现如图4.7。填写修改密表单后，前端会对新密码的长度、新密码与确认密码是否一致进行校验，提交修改表单后后端会对旧密码是否正确进行校验，正确则成功修改密码，否则返回失败信息。



图4.7 修改密码功能示例图

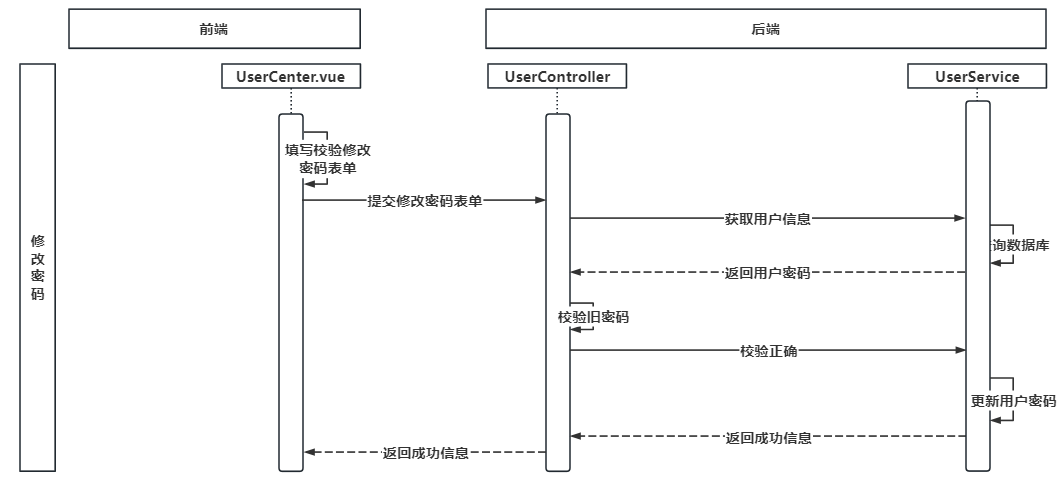
修改密码功能时序图如图4.8。填写修改密码表单后，由UserCenter.vue对表单进行校验，提交到UserController，UserController由当前的用户信息调用UserService到数据库中查询当前用户的旧密码，返回UserController中进行校验，正确则再次调用UserService更新数据库中用户密码，返回成功信息，否则返回失败信息。

图4.8 修改密码功能时序图

其中，修改密码的核心代码如下所示。

public Result updatePass(@Validated @RequestBody PassDto passDto, Principal principal) {

User user = userService.getByUsername(principal.getName());

boolean matches = passwordEncoder.matches

(passDto.getCurrentPass(), user.getPassword());

if (!matches) {

return Result.fail("旧密码不正确");

}

……//更新密码

return Result.succ("");

}

}

4.1.5 退出功能实现

退出功能实现如图4.9。点击退出后返回登录页面并清除用户信息。



图4.9 退出功能示例图

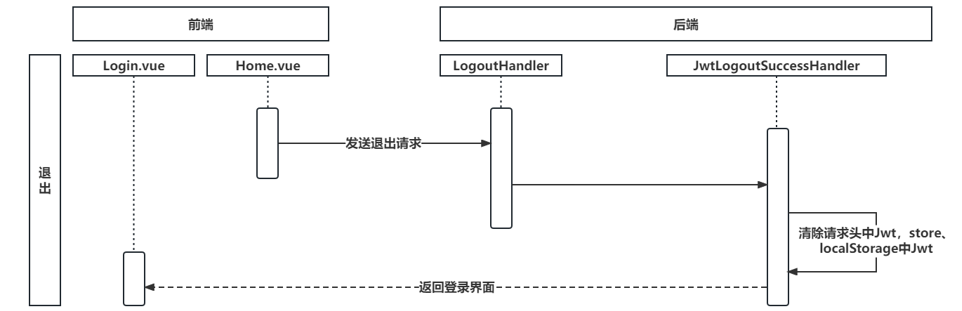
退出功能时序图如图4.10。在Home.vue中点击退出发送退出请求到LogoutHandler，再到自定义的JwtLogoutSuccessHandler中清除请求头中Jwt、store、localStorage中的用户信息，最后返回登录页面。

图4.10 退出功能时序图

其中，清除请求头中Jwt的代码如下。

response.setHeader(jwtUtils.getHeader(), "");

4.2 核心模块实现

4.2.1 表单更新与查询功能实现

表单更新与查询功能是所有主体都拥有的功能，以小区主体为例，实现如图4.11。在输入关键字点击搜索或改变分页参数后实现表单的更新与查询，将新的结果展示给用户。

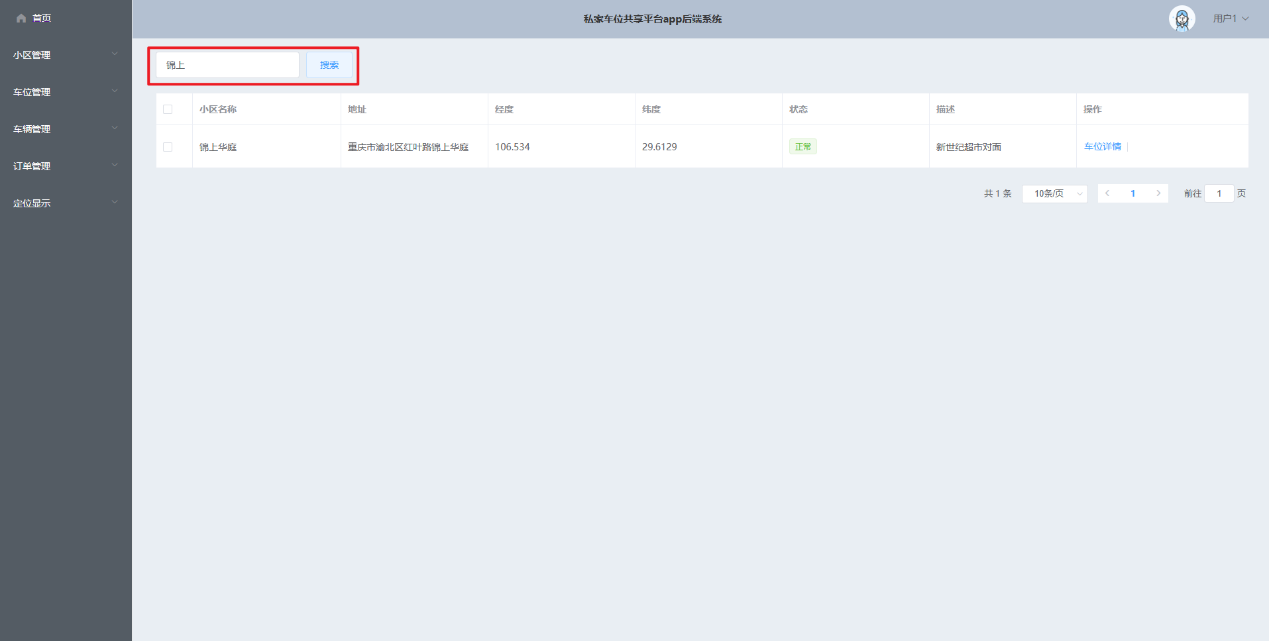


图4.11 表单更新与查询功能示例图

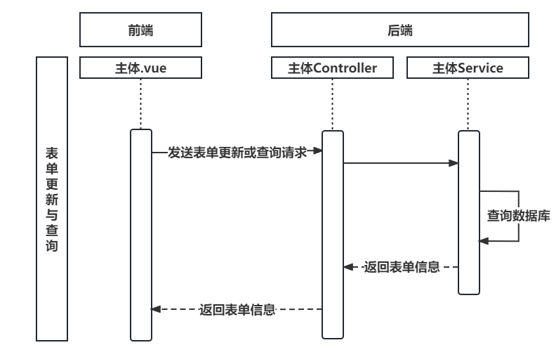
表单更新与查询功能时序图如图4.12。不同主体的表单更新与查询由不同主体的前端Vue与后端Controller、Service共同控制。

图4.12 表单更新与查询功能时序图

以小区主体为例，前端中控制表单更新与查询功能的代码如下，params参数中villagename为搜索关键字，current为分页参数中当前页，size为分页参数中每页的数据容量。

//更新以及查询

getVillageList() {

this.$axios.get("/vilman/village/list", {

params: {

villagename: this.searchForm.villagename,

current: this.current,

size: this.size

}

}).then(res => {……//传入前端})}

后端中控制表单更新与查询功能的代码如下。后端通过.isNotBlank()方法判断用户是否输入关键字进行搜索，并用Page类型存储查询结构。

public Result list(String villagename) {

Page<Village> pageData = villageService.page(getPage(),

new QueryWrapper<Village>()

.like(StrUtil.isNotBlank(villagename),"villagename",villagename)

);

return Result.succ(pageData);

}

4.2.2 新增与编辑功能实现

新增与编辑功能是所有主体都拥有的功能，以车位主体为例，实现如图4.13。

图4.13 新增与编辑功能示例图

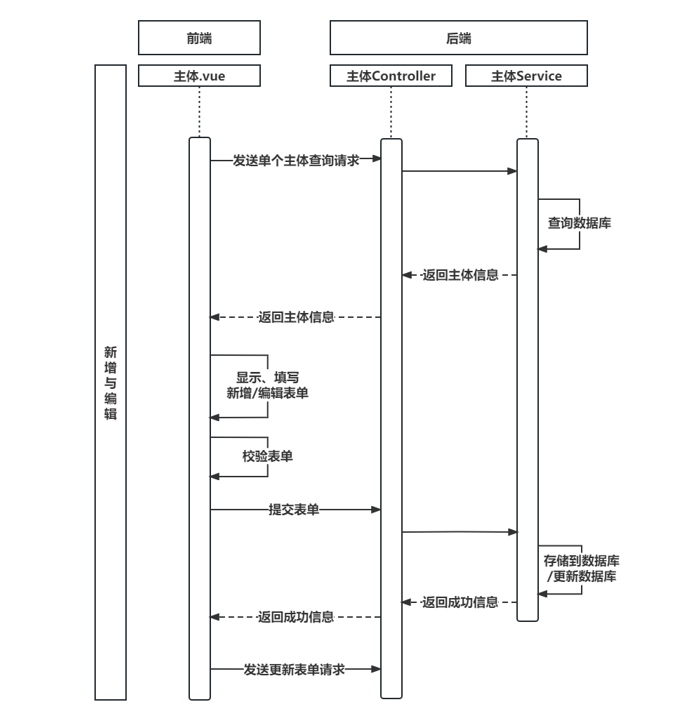
新增与编辑功能时序图如图4.14。不同主体的表单新增与编辑由不同主体的前端Vue与后端Controller、Service共同控制。编辑会在新增的流程前先对选中主体进行查询，并将返回信息显示到对话框中，新增则直接显示空白的对话框。填写表单后，前端的主体Vue会校验表格，提交表单后，由主体Controller调用主体Service新建或更新主体，保存到数据库中，返回成功信息。最后前端主体Vue还会自动调用一次更新表单的请求更新前端的表格数据。

图4.14 新增与编辑功能时序图

以车位主体为例，前端中控制新增与编辑功能的代码如下，通过editForm.id是否存在判断是编辑还是新增，在onClose:()方法中调用更新表单方法，如果取消则调用resetForm()方法清空表单。后端代码即简单的操作数据库，不再展示。

submitForm(formName) {

this.$refs[formName].validate((valid) => {

if (valid) {

this.$axios.post('/parkman/park/'

+ (this.editForm.id?'update' : 'save'), this.editForm)

.then(res => {

……

onClose:() => {

this.getParkList()

}

});

this.resetForm(formName)

})

} else {

return false;

}

});

}

4.2.3 角色管理-分配权限功能实现

分配权限功能实现如图4.15。

图4.15 分配权限功能示例图

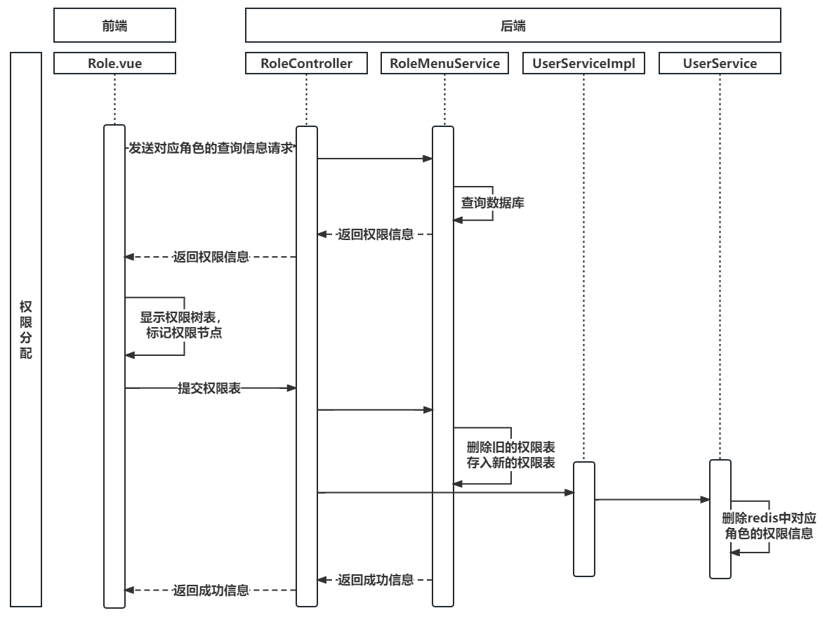
分配权限功能时序图如图4.16。在前端Role.vue中发送分配权限的请求，会通过RoleMenuService到RoleMenu中间表中查询选中的角色对应的权限信息。返回前端后将权限数据展示到树状表中，对拥有的权限节点进行标记，分配新的权限后提交权限表，会由RoleController调用RoleMenuService删除旧的权限表加入新的权限表，并调用UserService中的clearUserAuthorityInfoByRoleId()方法清除相应权限，最后返回成功信息。

图4.16 分配权限功能时序图

分配权限功能的核心实现代码如下。

public Result perm(@PathVariable("roleId") Long roleId,@RequestBody Long[] menuIds) {

List<RoleMenu> roleMenus = new ArrayList<>();

Arrays.stream(menuIds).forEach(menuId -> {

RoleMenu roleMenu = new RoleMenu();

roleMenu.setMenuId(menuId);

roleMenu.setRoleId(roleId);

roleMenus.add(roleMenu);

});

……// 先删除原来的记录，再保存新的

……// 删除缓存

return Result.succ(menuIds);

}

4.2.4 订单管理-编辑订单功能实现

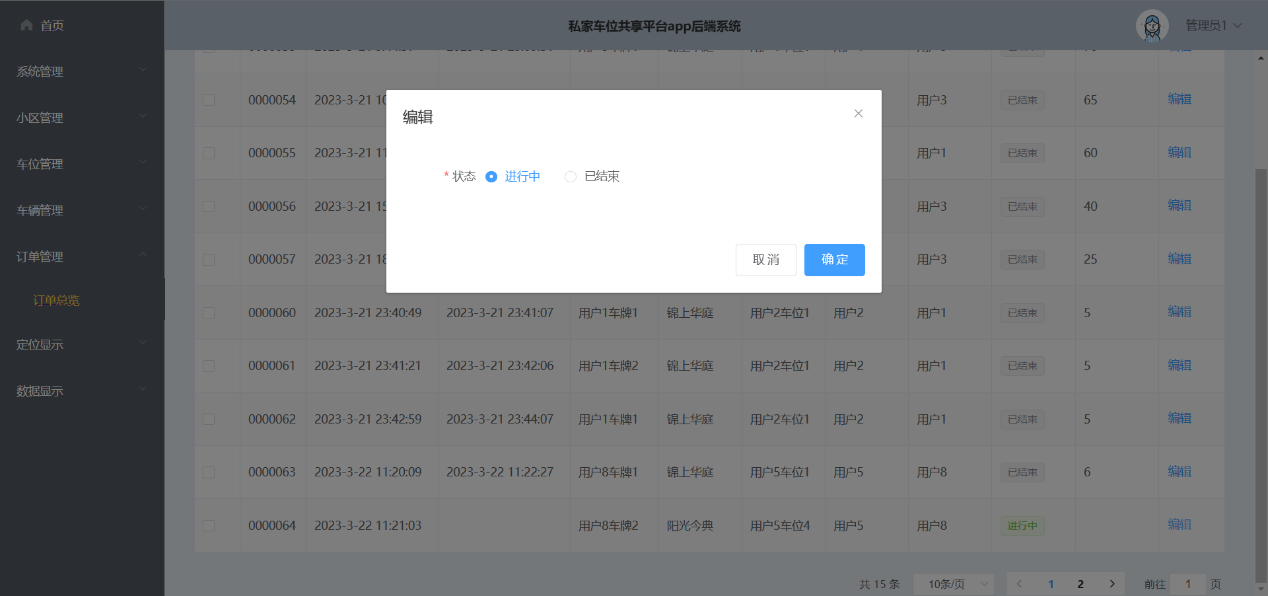
编辑订单功能实现如图4.17。对已结束的订单进行编辑会返回已结束订单不可再编辑的信息，对进行中的订单可以进行编辑并返回成功信息。

图4.17 编辑订单功能示例图

编辑订单功能时序图如图4.18。由前端Order.vue发送编辑请求，首先查询并返回选中订单的状态信息，若状态为已结束则前端显示订单已结束不可再编辑信息，若进行中则显示编辑订单对话框。编辑订单时保持订单状态为进行中则直接返回成功信息，若将状态置为已结束则由Order.vue获取当前时间作为订单结束时间，得到订单总时间，与结束订单请求一并发送至IndentController。IndentController通过调用ParkService与CarService将订单中对应的车位、车辆状态置为空闲，同时返回车位价格到IndentController中计算订单总价。IndentController再次调用IndentService更新订单状态、结束时间、总价信息，保存到数据库。最后，返回成功信息，前端Order.vue还会自动调用一次更新表单的请求更新前端的表格数据。

前端中提交编辑订单表单前的核心代码如下，通过getCurrentTime()方法与getHour()方法获取订单结束时间与订单总时间，交由后端去计算总价。

if(this.editForm.statu == 0){

this.editForm.orderend = this.getCurrentTime()

let hour = this.getHour(this.editForm.orderstart,this.editForm.orderend)

this.editForm.cost = hour

}

后端计算订单总价的核心代码如下，后端接收前端订单总时间参数时无需新建其它参数，直接使用Indent下cost参数接收，最后乘以车位的小时单价即可得到订单的总价。

if(indent.getStatu()==0)

{

indent.setCost(indent.getCost() \* park.getPrice());

}

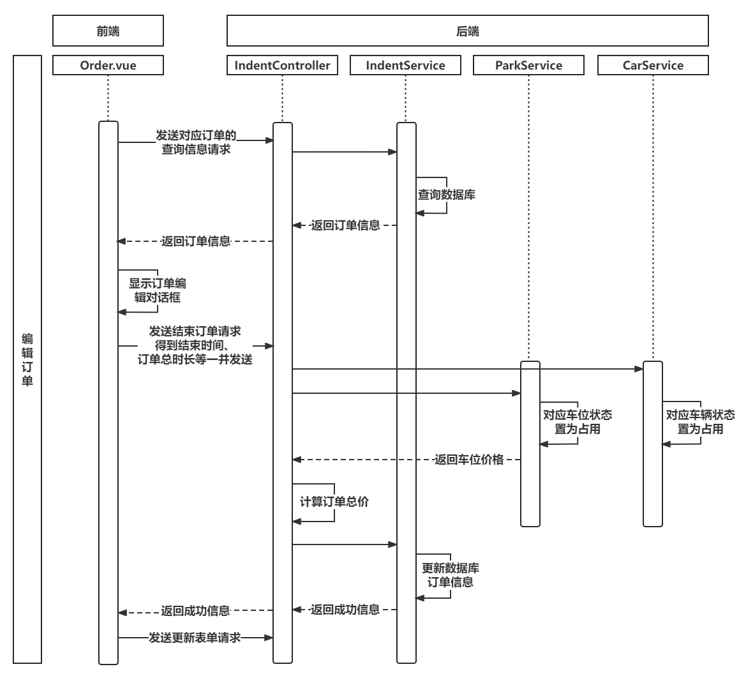
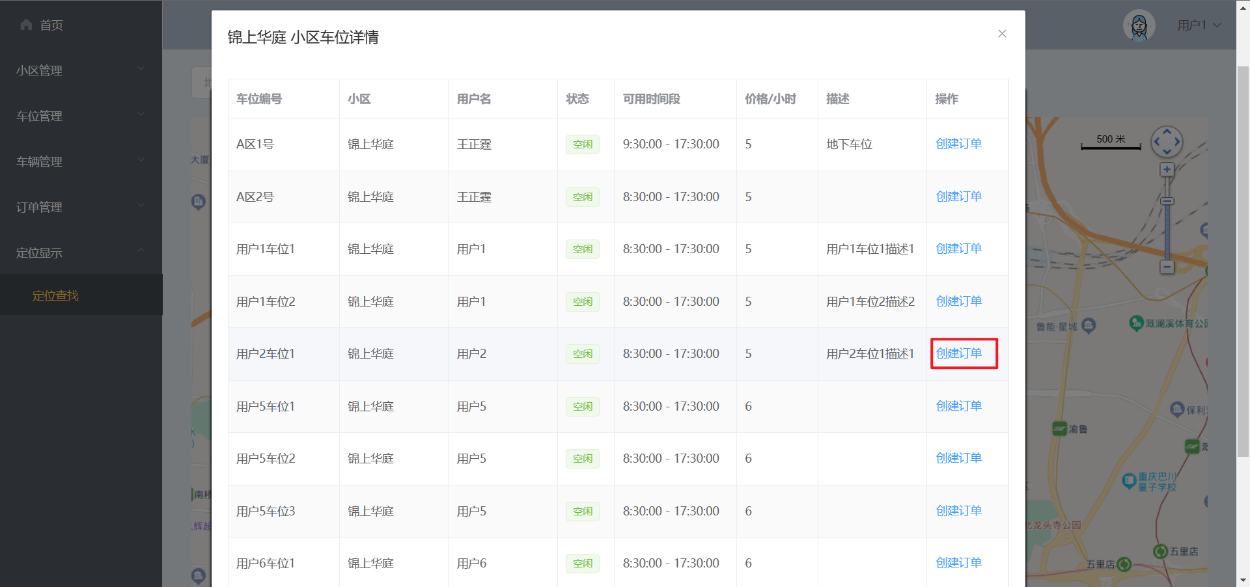


图4.18 编辑订单功能时序图

4.2.5 定位查找-创建订单功能实现

创建订单功能实现如图4.19。

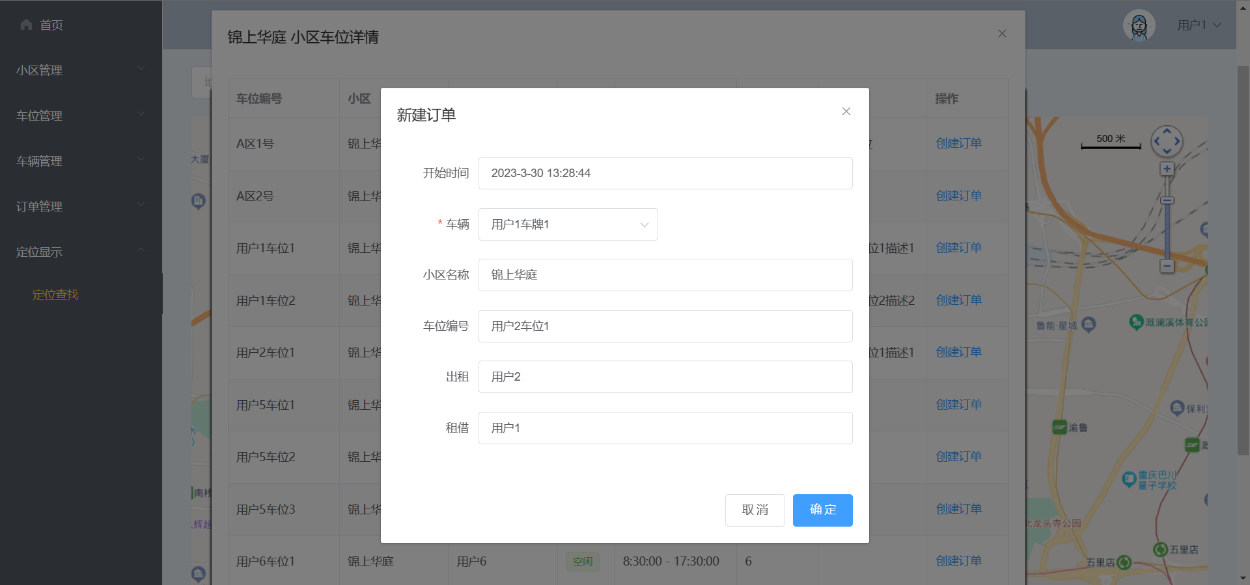


图4.19 创建订单功能示例图

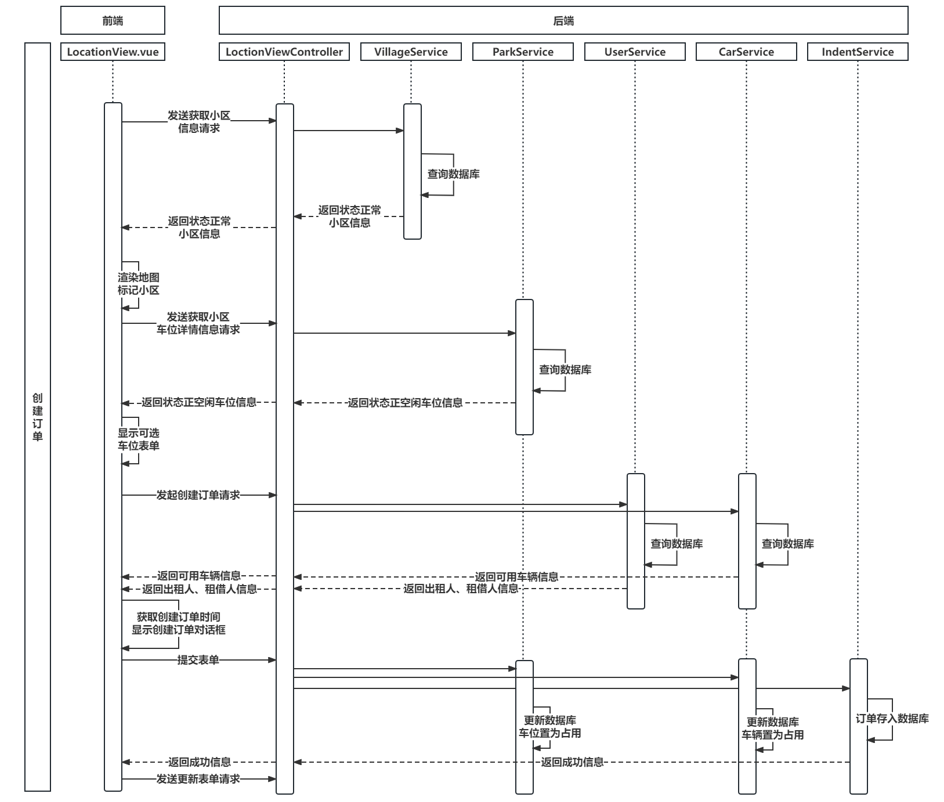
创建订单功能时序图如图4.20。首先由前端LocationView.vue发送获取小区信息请求，VillageService查询返回状态正常的小区信息，LocationView.vue收到信息后渲染地图并对小区进行标记展示在前端。点击小区标记后LocationView.vue发送获取选中小区车位信息请求，ParkService查询返回状态空闲的车位信息，展示到前端表格中。选中需求的车位，点击创建订单，LocationView.vue发送创建订单请求，LocationViewController调用CarService、UserService查询返回可选车辆信息、出租人和租借人信息，再由LocationView.vue自身获取当前时间作为订单开始时间，将上述信息全部显示在新建订单对话框中。点击确定，发送提交表单请求，LocationViewController再次调用几个主体Service，将对应车位、车辆状态置为占用，订单置为进行中，并存储到数据库，返回成功信息。最后，前端LocationView.vue还会自动调用一次更新表单的请求更新前端的表格数据。

图4.20 创建订单功能时序图

前端加入Vue-Baidumap插件后渲染地图时对小区进行标记的核心代码如下。主要通过<bm-marker>标签进行标记，lng、lat分别表示小区的经度、纬度，通过绑定@click事件与infoWindowOpen()方法控制小区车位详情的显示。

<template v-for="item in tableDataVillage">

<bm-marker :position="{lng: item.lng, lat: item.lat}"

@click="infoWindowOpen(item)">

</bm-marker>

……//小区车位详情表单

</template>

4.3 辅助模块-数据显示功能实现

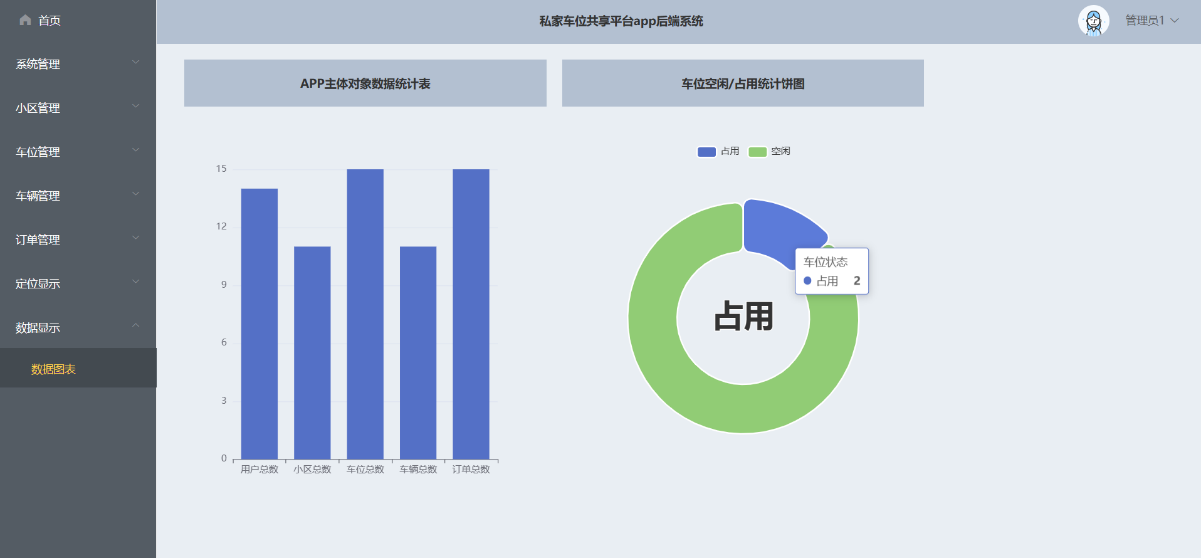
数据显示功能实现如图4.21，仅管理员可视。

图4.21 数据显示功能示例图

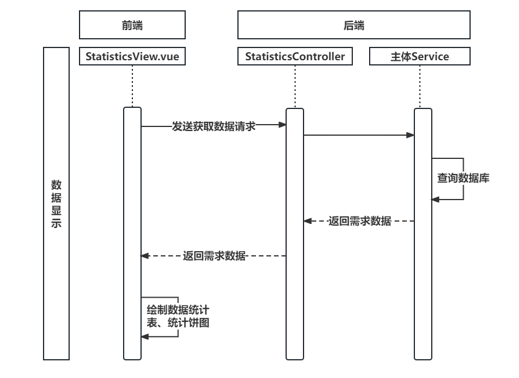
数据显示功能时序图如图4.22。由前端StatisticsView.vue发送获取数据请求，由后端StatisticsController调用各个主体Service到数据库中查询数据并返回，交由前端进行绘制统计数、饼状图。

图4.22 数据显示功能时序图

其中，前端加入Vue-Echart组件后绘制统计图、饼状图的核心代码如下。主要通过<e-charts>标签进行图像绘制，通过实现option()方法制定绘制的统计图类型、参数、数据等。

<e-charts class="chart1" :option="option1"></e-charts>

……

option1(){

return{

xAxis: {

type: 'category',

data: ['用户总数', '小区总数', '车位总数', '车辆总数', '订单总数']

},

yAxis: {type: 'value'},

series: [……//数据]}}

4.4 本章小结

本章节详细介绍了私家车位共享平台APP后端系统的总体实现。依照第三章系统功能设计的分类，分别介绍了基础模块的认证、授权、注册、退出、修改密码功能的实现；核心功能模块的表单更新与查询、新增与编辑、分配权限、编辑订单、创建订单功能的实现；辅助模块的数据显示功能的实现。对每个功能展示实际系统的示例图、应用UML统一建模工具绘制相应的时序图进一步阐述、并展示实现功能的核心代码。