**大连科技学院**

**帝可得管理系统开发**

**后端开发综合实训**

|  |  |
| --- | --- |
| **学 院** | 信息科学与技术学院 |
| **专业班级** | 计算机科学与技术(专升本)23-1班 |
| **学生姓名** | 王丹丹 |
| **学生学号** | 2306980112 |
| **指导教师** | 王小勇 |
| **职 称** | 讲师 |

**完成日期 2024 年 11月 16 日**

目 录

[1 帝可得管理系统概述 1](#_Toc27055)

[1.1 课题简介 1](#_Toc28490)

[1.2 设计目的 1](#_Toc19232)

[1.3 系统开发所采用的技术 1](#_Toc24540)

[1.3.1 开发工具 1](#_Toc7774)

[1.3.2 开发使用的数据库系统 1](#_Toc19563)

[1.3.3 所用技术 2](#_Toc9848)

[1.4 系统功能模块 2](#_Toc18657)

[2 帝可得管理系统的的设计与实现 5](#_Toc30824)

[2.1 区域模块设计 5](#_Toc27710)

[2.1.1 区域管理模块效果图 5](#_Toc30061)

[2.1.2 区域管理模块数据库 6](#_Toc10235)

[2.1.3 前端代码 6](#_Toc29036)

[2.1.4 后端代码 9](#_Toc20343)

[2.2 合作商模块设计 10](#_Toc11567)

[2.2.1 合作商管理模块效果图 10](#_Toc30637)

[2.2.2 合作商管理模块数据库 11](#_Toc19380)

[2.2.3 前端代码 12](#_Toc22987)

[2.2.4 后端代码 15](#_Toc25330)

[3 后端实训总结 18](#_Toc30568)

# 1 帝可得管理系统概述

## 1.1 课题简介

帝可得管理系统是一个基于物联网概念的智能售货机运营管理系统。具备完善的平台管理功能，它主要对分布广泛的售货机进行集中管控。在商品管理方面，能实时监控商品的库存信息，自动提醒补货，同时可灵活设置商品的价格、陈列方式等参数。设备管理上，可远程监测售货机的运行状态，包括设备的温度、湿度、故障报警等，以便及时维护，降低停机时间。在销售数据统计上，系统精确记录每一笔交易，通过数据分析为运营者提供销售趋势、热门商品等关键信息，指导商品的投放和运营策略调整。此外，还支持支付方式管理和用户交互功能优化，极大地提高了售货机运营的效率和效益，为售货机业务的拓展和精细化管理提供有力支撑。

## 1.2 设计目的

近年来，无接触式智能售货机逐渐成为年轻消费者的首选购物方式。这些设备不仅在公园、地铁站和商场等公共场所广泛分布，为顾客提供了极大的购物便利，而且由于其运营成本低廉和较高的利润空间，被众多电商企业视为新兴的蓝海市场。

## 1.3 系统开发所采用的技术

### 1.3.1 开发工具

前端使用 Vue.js 框架，它能高效构建用户界面，通过组件化的方式提高代码的复用性和可维护性。后端开发选择 IntelliJ IDEA ，利用 Java 语言和 Springboot等框架构建系统逻辑。数据库使用 Navicat for MySQL 等管理 MySQL 数据库，进行数据库的创建、表结构设计、数据查询和更新等操作。还使用 Git 进行版本控制和代码管理，通过 GitHub 或 GitLab 等平台实现团队协作，方便开发人员共同推进项目。同时利用 JIRA 等进行项目进度跟踪和问题管理。

### 1.3.2 开发使用的数据库系统

在开发中，它具有诸多优势。首先，MySQL 拥有丰富的数据类型，能满足多样化的数据存储需求，无论是简单的文本、数字，还是复杂的日期、二进制数据等都可妥善处理。其查询语言功能强大，开发人员可以通过灵活编写 SQL 语句实现高效的数据检索、更新和删除等操作。而且，它支持多用户并发访问，通过合理的配置可以在高负载环境下稳定运行。同时，MySQL 在多种操作系统上都能良好运行，具有很好的兼容性，还提供了可视化的管理工具，方便对数据库进行管理和维护。

### 1.3.3 所用技术

在技术层面，采用 Java 等高性能编程语言，结合 Spring Boot 等框架搭建稳定的后端服务，实现业务逻辑处理和数据交互。使用关系型数据库如 MySQL 存储核心数据，保证数据的完整性和一致性。利用 Redis 缓存技术提升系统性能，应对高并发场景。前端方面运用 Vue.js的流行的前端框架构建用户界面，实现良好的交互体验。通过 CSS3 和 HTML5 进行页面布局和样式设计，使界面美观大方且适配多种设备。采用 Ajax 等技术实现异步数据加载，增强页面响应速度和流畅度。

## 1.4 系统功能模块

****点位选择****：点位指的是智能售货机的具体放置位置。选择点位时，我们会考虑人流量、目标顾客群体、可见度以及便利性等因素，以最大化售货机的使用效率和顾客的购买体验。以下是点位管理的流程图如下图所示：



****区域管理****：为了更高效地进行经营管理，公司将运营范围划分为若干个逻辑区域。这些区域的划分基于业务需求，可能与地理上的行政区域有所区别，以确保更合理的资源分配和更高效的运营管理，以下是区域管理的流程图如下图所示：



****售货机功能****：智能售货机就像是一个自动的小店，里面摆满了各种商品。顾客想要什么，直接在机器上选，然后机器就会把商品送到他们手中，就像是一个自动化的仓库。以下是售货机样式图如下图所示：



****货道设计：****售货机里面的货道，你可以想象成超市里的那种货架。每一层都有好几个位置可以放商品，这样就能放很多种不同的商品，而且每一层都能放很多，这样顾客的选择就会更多。以下是货道样式图如下图所示：



# 2 帝可得管理系统的的设计与实现

## 2.1 区域模块设计

### 2.1.1 区域管理模块效果图

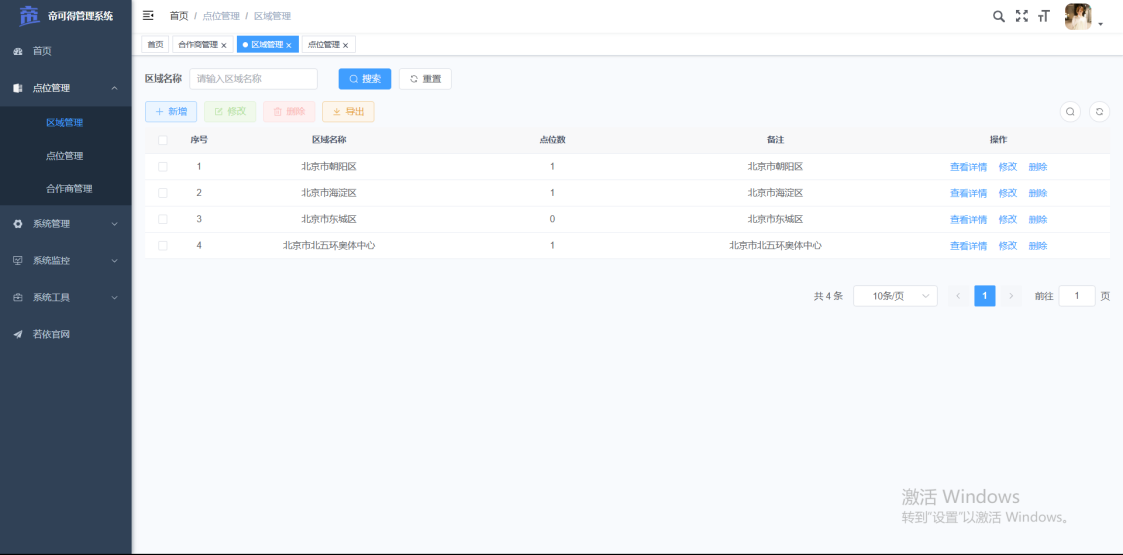


图2.1 区域管理模块效果图



图2.2 区域管理查看详情效果图

帝可得区域管理是整个管理系统中的一个关键模块，主要负责对售货机运营的地理区域进行统筹规划、监控和优化。它将运营范围划分为不同的区域单元，通过对每个区域的精细化管理，实现资源的合理配置、销售业绩的提升以及服务质量的优化。区域管理包括区域名称、点位数、备注、功能包括查看详情、修改、删除、增加等，如图2.1，图2.2所示。

### 2.1.2 区域管理模块数据库

区域表包括主键id，区域名称region\_name，创建时间create\_time，修改时间update\_time，创建人create\_by，修改人update\_by，备注remark。如表2.1所示。

表2.1 区域表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名** | **说 明** | **类 型** | **长 度** | **可否为空** | **主键** |
| id | 主键 | int | 11 | 否 | 是 |
| region\_name | 区域名称 | varchar | 255 | 否 | 否 |
| create\_time | 创建时间 | timestamp | 64 | 是 | 否 |
| update\_time | 修改时间 | timestamp | 64 | 是 | 否 |
| create\_by | 创建人 | varchar | 64 | 是 | 否 |
| update\_by | 修改人 | varchar | 64 | 是 | 否 |
| remark | 备注 | varchar | 255 | 是 | 否 |

### 2.1.3 前端代码

以下代码主要展示核心功能、样式以及逻辑等，是区域管理的前端vue核心代码，代码如下展示：

<!-- 添加或修改区域管理对话框 -->

<el-dialog :title="title" v-model="open" width="500px" append-to-body>

<el-form ref="regionRef" :model="form" :rules="rules" label-width="80px">

<el-form-item label="区域名称" prop="regionName">

<el-input v-model="form.regionName" placeholder="请输入区域名称" />

</el-form-item>

<el-form-item label="备注" prop="remark">

<el-input v-model="form.remark" type="textarea" placeholder="请输入内容" />

</el-form-item>

</el-form>

<template #footer>

<div class="dialog-footer">

<el-button type="primary" @click="submitForm">确 定</el-button>

<el-button @click="cancel">取 消</el-button>

</div>

</template>

</el-dialog>

<!-- 查看详情对话框 -->

<el-dialog title="区域详情" v-model="regionInfoOpen" width="500px" append-to-body>

<el-form-item label="区域名称" prop="regionName">

<el-input v-model="form.regionName" disabled />

</el-form-item>

<label>包含点位：</label>

<el-table :data="nodeList">

<el-table-column label="序号" type="index" width="50" align="center" />

<el-table-column label="点位名称" align="center" prop="nodeName" />

<el-table-column label="设备数量" align="center" prop="vmCount" />

</el-table>

</el-dialog>

</div>

</template>

<script setup name="Region">

import { listRegion, getRegion, delRegion, addRegion, updateRegion } from "@/api/manage/region";

import { listNode } from "@/api/manage/node";

import { loadAllParams } from "@/api/page";

const { proxy } = getCurrentInstance();

const regionList = ref([]);

const open = ref(false);

const loading = ref(true);

const showSearch = ref(true);

const ids = ref([]);

const single = ref(true);

const multiple = ref(true);

const total = ref(0);

const title = ref("");

const data = reactive({

form: {},

queryParams: {

pageNum: 1,

pageSize: 10,

regionName: null,

},

rules: {

regionName: [

{ required: true, message: "区域名称不能为空", trigger: "blur" }

],

remark: [

{ required: true, message: "备注不能为空", trigger: "blur" }

]

}

});

const { queryParams, form, rules } = toRefs(data);

/\*\* 查询区域管理列表 \*/

function getList() {

loading.value = true;

listRegion(queryParams.value).then(response => {

regionList.value = response.rows;

total.value = response.total;

loading.value = false;

});

}

// 取消按钮

function cancel() {

open.value = false;

reset();

}

// 表单重置

function reset() {

form.value = {

id: null,

regionName: null,

createTime: null,

updateTime: null,

createBy: null,

updateBy: null,

remark: null

};

proxy.resetForm("regionRef");

}

/\*\* 搜索按钮操作 \*/

function handleQuery() {

queryParams.value.pageNum = 1;

getList();

}

/\*\* 重置按钮操作 \*/

function resetQuery() {

proxy.resetForm("queryRef");

handleQuery();

}

// 多选框选中数据

function handleSelectionChange(selection) {

ids.value = selection.map(item => item.id);

single.value = selection.length != 1;

multiple.value = !selection.length;

}

/\*\* 新增按钮操作 \*/

function handleAdd() {

reset();

open.value = true;

title.value = "添加区域管理";

}

/\*\* 修改按钮操作 \*/

function handleUpdate(row) {

reset();

const \_id = row.id || ids.value

getRegion(\_id).then(response => {

form.value = response.data;

open.value = true;

title.value = "修改区域管理";

});

}

/\* 查看详情按钮操作 \*/

const nodeList = ref([]);

const regionInfoOpen = ref(false);

function getRegionInfo(row) {

// 查询区域信息

reset();

const \_id = row.id

getRegion(\_id).then(response => {

form.value = response.data;

});

// 查询点位列表

loadAllParams.regionId = row.id;

listNode(loadAllParams).then(response => {

nodeList.value = response.rows;

});

regionInfoOpen.value = true;

}

/\*\* 提交按钮 \*/

function submitForm() {

proxy.$refs["regionRef"].validate(valid => {

if (valid) {

if (form.value.id != null) {

updateRegion(form.value).then(response => {

proxy.$modal.msgSuccess("修改成功");

open.value = false;

getList();

});

} else {

addRegion(form.value).then(response => {

proxy.$modal.msgSuccess("新增成功");

open.value = false;

getList();

});

}

}

});

}

/\*\* 删除按钮操作 \*/

function handleDelete(row) {

const \_ids = row.id || ids.value;

proxy.$modal.confirm('是否确认删除区域管理编号为"' + \_ids + '"的数据项？').then(function() {

return delRegion(\_ids);

}).then(() => {

getList();

proxy.$modal.msgSuccess("删除成功");

}).catch(() => {});

}

/\*\* 导出按钮操作 \*/

function handleExport() {

proxy.download('manage/region/export', {

...queryParams.value

}, `region\_${new Date().getTime()}.xlsx`)

}

getList();

</script>

### 2.1.4 后端代码

Regioncontroller在与前端交互中扮演着关键的接口角色。它接收来自前端的 HTTP 请求，这些请求可能是通过浏览器中的 JavaScript 代码发送的，以下是regioncontroller的代码实现。以下是regioncontroller的核心代码：

**Regioncontroller模块:**

/\*\*

\* 区域管理Controller

\*

\* @author ruoyi1

\* @date 2024-11-06

\*/

@RestController

@RequestMapping("/manage/region")

public class RegionController extends BaseController

{

@Autowired

private IRegionService regionService;

/\*\*

\* 查询区域管理列表

\*/

@PreAuthorize("@ss.hasPermi('manage:region:list')")

@GetMapping("/list")

public TableDataInfo list(Region region)

{

startPage();

List<RegionVo> voList =regionService.selectRegionVoList(region);

return getDataTable(voList);

}

/\*\*

\* 导出区域管理列表

\*/

@PreAuthorize("@ss.hasPermi('manage:region:export')")

@Log(title = "区域管理", businessType = BusinessType.EXPORT)

@PostMapping("/export")

public void export(HttpServletResponse response, Region region)

{

List<Region> list = regionService.selectRegionList(region);

ExcelUtil<Region> util = new ExcelUtil<Region>(Region.class);

util.exportExcel(response, list, "区域管理数据");

}

/\*\*

\* 获取区域管理详细信息

\*/

@PreAuthorize("@ss.hasPermi('manage:region:query')")

@GetMapping(value = "/{id}")

public AjaxResult getInfo(@PathVariable("id") Long id)

{

return success(regionService.selectRegionById(id));

}

/\*\*

\* 新增区域管理

\*/

@PreAuthorize("@ss.hasPermi('manage:region:add')")

@Log(title = "区域管理", businessType = BusinessType.INSERT)

@PostMapping

public AjaxResult add(@RequestBody Region region)

{

return toAjax(regionService.insertRegion(region));

}

/\*\*

\* 修改区域管理

\*/

@PreAuthorize("@ss.hasPermi('manage:region:edit')")

@Log(title = "区域管理", businessType = BusinessType.UPDATE)

@PutMapping

public AjaxResult edit(@RequestBody Region region)

{

return toAjax(regionService.updateRegion(region));

}

/\*\*

\* 删除区域管理

\*/

@PreAuthorize("@ss.hasPermi('manage:region:remove')")

@Log(title = "区域管理", businessType = BusinessType.DELETE)

@DeleteMapping("/{ids}")

public AjaxResult remove(@PathVariable Long[] ids)

{

return toAjax(regionService.deleteRegionByIds(ids));

}

}

## 2.2 合作商模块设计

### 2.2.1 合作商管理模块效果图

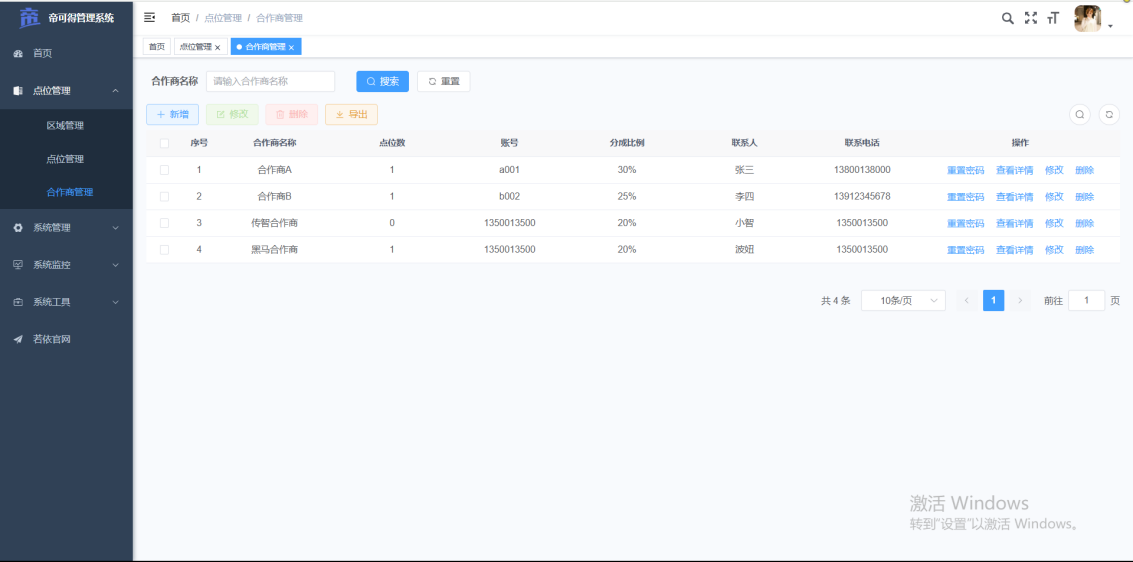


图2.3 合作商管理模块效果图

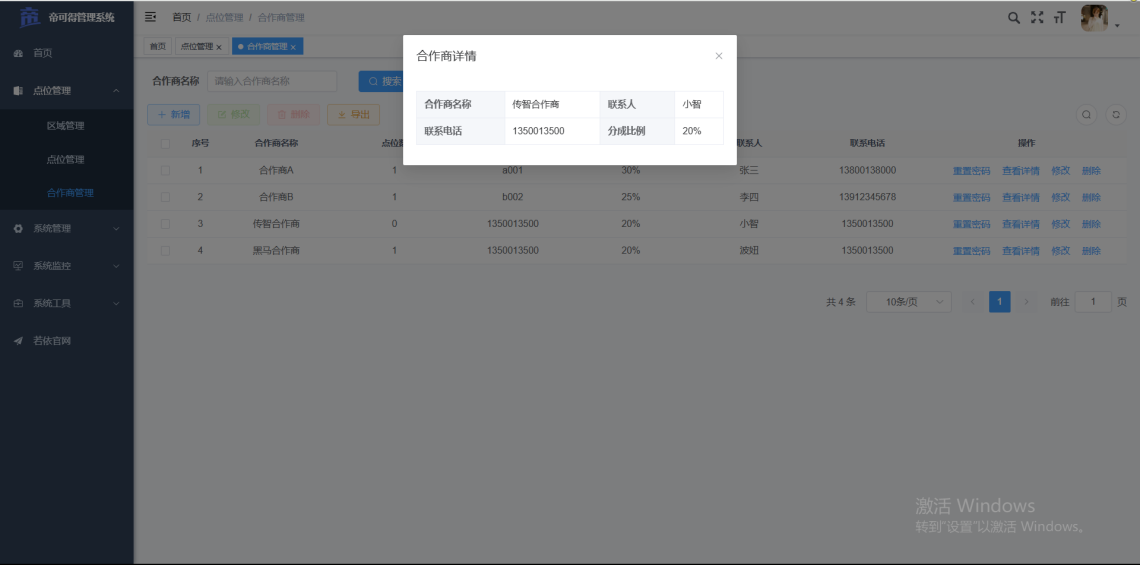


图2.4 合作商模块查看详情效果图

帝可得合作商管理可以记录合作商的名称、联系人姓名、联系电话等，以便进行沟通和业务联系。这些信息是合作商的基本标识，对于建立合作关系和日常沟通非常重要。合作商管理包括合作商名称、点位数、账号、分成比例、联系电话、功能包括查看详情、重置密码、修改、删除、增加等，如图2.3，图2.4所示。

### 2.2.2 合作商管理模块数据库

合作商表包括主键id，合作商名称partner\_name，联系人contact\_person，联系电话contact\_phone，分成比例profit\_ratio，账号account，密码password,创建时间create\_time，修改时间updata\_time。如表2.2所示。

表2.2 合作商表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名** | **说 明** | **类 型** | **长 度** | **可否为空** | **主键** |
| id | 主键 | int | 11 | 否 | 是 |
| partner\_name | 合作商名称 | varchar | 255 | 否 | 否 |
| contact\_person | 联系人 | varchar | 64 | 是 | 否 |
| contact\_phone | 联系电话 | varchar | 15 | 是 | 否 |
| profit\_ratio | 分成比例 | int | 64 | 是 | 否 |
| account | 账号 | varchar | 64 | 是 | 否 |
| password | 密码 | varchar | 64 | 是 | 否 |
| create\_time | 创建时间 | timestamp | 64 | 是 | 否 |
| updata\_time | 修改时间 | timestamp | 64 | 是 | 否 |

### 2.2.3 前端代码

以下代码主要展示核心功能、样式以及逻辑等，是合作商管理的前端vue核心代码，代码如下展示：

<!-- 添加或修改合作商管理对话框 -->

<el-dialog :title="title" v-model="open" width="500px" append-to-body>

<el-form ref="partnerRef" :model="form" :rules="rules" label-width="100px">

<el-form-item label="合作商名称" prop="partnerName">

<el-input v-model="form.partnerName" placeholder="请输入合作商名称" />

</el-form-item>

<el-form-item label="联系人" prop="contactPerson">

<el-input v-model="form.contactPerson" placeholder="请输入联系人" />

</el-form-item>

<el-form-item label="联系电话" prop="contactPhone">

<el-input v-model="form.contactPhone" placeholder="请输入联系电话" />

</el-form-item>

<el-form-item label="创建时间" prop="createTime" v-if="form.id != null">

{{form.createTime}}

</el-form-item>

<el-form-item label="分成比例" prop="profitRatio">

<el-input v-model="form.profitRatio" placeholder="请输入分成比例" />

</el-form-item>

<el-form-item label="账号" prop="account" v-if="form.id == null">

<el-input v-model="form.account" placeholder="请输入账号" />

</el-form-item>

<el-form-item label="密码" prop="password" v-if="form.id == null">

<el-input v-model="form.password" type="password" placeholder="请输入密码" />

</el-form-item>

</el-form>

<template #footer>

<div class="dialog-footer">

<el-button type="primary" @click="submitForm">确 定</el-button>

<el-button @click="cancel">取 消</el-button>

</div>

</template>

</el-dialog>

<!-- 查看合作商详情对话框 -->

<el-dialog title="合作商详情" v-model="partnerInfoOpen" width="500px" append-to-body>

<!-- 使用el-descriptions组件以卡片形式展示信息，更加整洁 -->

<el-descriptions :column="2" border>

<el-descriptions-item label="合作商名称">{{ form.partnerName }}</el-descriptions-item>

<el-descriptions-item label="联系人">{{ form.contactPerson }}</el-descriptions-item>

<el-descriptions-item label="联系电话">{{ form.contactPhone }}</el-descriptions-item>

<el-descriptions-item label="分成比例">{{ form.profitRatio }}%</el-descriptions-item>

</el-descriptions>

</el-dialog>

</div>

</template>

<script setup name="Partner">

import { listPartner, getPartner, delPartner, addPartner, updatePartner,resetPartnerPwd } from "@/api/manage/partner";

import { ref } from "vue";

const { proxy } = getCurrentInstance();

const partnerList = ref([]);

const open = ref(false);

const loading = ref(true);

const showSearch = ref(true);

const ids = ref([]);

const single = ref(true);

const multiple = ref(true);

const total = ref(0);

const title = ref("");

const data = reactive({

form: {},

queryParams: {

pageNum: 1,

pageSize: 10,

partnerName: null,

},

rules: {

partnerName: [

{ required: true, message: "合作商名称不能为空", trigger: "blur" }

],

contactPerson: [

{ required: true, message: "联系人不能为空", trigger: "blur" }

],

contactPhone: [

{ required: true, message: "联系电话不能为空", trigger: "blur" }

],

profitRatio: [

{ required: true, message: "分成比例不能为空", trigger: "blur" }

],

account: [

{ required: true, message: "账号不能为空", trigger: "blur" }

],

password: [

{ required: true, message: "密码不能为空", trigger: "blur" }

],

}

});

const { queryParams, form, rules } = toRefs(data);

/\*\* 查询合作商管理列表 \*/

function getList() {

loading.value = true;

listPartner(queryParams.value).then(response => {

partnerList.value = response.rows;

total.value = response.total;

loading.value = false;

});

}

// 取消按钮

function cancel() {

open.value = false;

reset();

}

// 表单重置

function reset() {

form.value = {

id: null,

partnerName: null,

contactPerson: null,

contactPhone: null,

profitRatio: null,

account: null,

password: null,

createTime: null,

updateTime: null,

createBy: null,

updateBy: null,

remark: null

};

proxy.resetForm("partnerRef");

}

/\*\* 搜索按钮操作 \*/

function handleQuery() {

queryParams.value.pageNum = 1;

getList();

}

/\*\* 重置按钮操作 \*/

function resetQuery() {

proxy.resetForm("queryRef");

handleQuery();

}

// 多选框选中数据

function handleSelectionChange(selection) {

ids.value = selection.map(item => item.id);

single.value = selection.length != 1;

multiple.value = !selection.length;

}

/\*\* 新增按钮操作 \*/

function handleAdd() {

reset();

open.value = true;

title.value = "添加合作商管理";

}

/\*\* 修改按钮操作 \*/

function handleUpdate(row) {

reset();

const \_id = row.id || ids.value

getPartner(\_id).then(response => {

form.value = response.data;

open.value = true;

title.value = "修改合作商管理";

});

}

/\*\* 查看详情按钮操作 \*/

const partnerInfoOpen = ref(false)

function getPartnerInfo(row) {

reset();

const \_id = row.id

getPartner(\_id).then(response => {

form.value = response.data;

partnerInfoOpen.value = true;

});

}

/\*\* 提交按钮 \*/

function submitForm() {

proxy.$refs["partnerRef"].validate(valid => {

if (valid) {

if (form.value.id != null) {

updatePartner(form.value).then(response => {

proxy.$modal.msgSuccess("修改成功");

open.value = false;

getList();

});

} else {

addPartner(form.value).then(response => {

proxy.$modal.msgSuccess("新增成功");

open.value = false;

getList();

});

}

}

});

}

/\*\* 删除按钮操作 \*/

function handleDelete(row) {

const \_ids = row.id || ids.value;

proxy.$modal.confirm('是否确认删除合作商管理编号为"' + \_ids + '"的数据项？').then(function() {

return delPartner(\_ids);

}).then(() => {

getList();

proxy.$modal.msgSuccess("删除成功");

}).catch(() => {});

}

/\*\* 重置合作商密码按钮操作 \*/

function resetPwd(row) {

const \_id = row.id

proxy.$modal.confirm('你确定要重置该合作商密码吗？').then(function() {

return resetPartnerPwd(\_id);

}).then(() => {

proxy.$modal.msgSuccess("重置密码成功");

}).catch(() => {});

}

/\*\* 导出按钮操作 \*/

function handleExport() {

proxy.download('manage/partner/export', {

...queryParams.value

}, `partner\_${new Date().getTime()}.xlsx`)

}

getList();

</script>

### 2.2.4 后端代码

Partnercontroller负责从后端获取合作商的各类详细信息，如合作商名称、联系人、联系电话等，并在前端页面以清晰、易读的格式展示给用户。例如，可能会在一个专门的合作商列表页面，将各个合作商的关键信息以表格形式罗列出来，方便用户快速浏览和了解整体情况。以下是partnercontroller的核心代码：

**Partnercontroller模块**

/\*\*

\* 合作商管理Controller

\*

\* @author ruoyi1

\* @date 2024-11-06

\*/

@RestController

@RequestMapping("/manage/partner")

public class PartnerController extends BaseController

{

@Autowired

private IPartnerService partnerService;

/\*\*

\* 查询合作商管理列表

\*/

@PreAuthorize("@ss.hasPermi('manage:partner:list')")

@GetMapping("/list")

public TableDataInfo list(Partner partner)

{

startPage();

List<PartnerVo> voList =partnerService.selectPartnerVoList(partner);

return getDataTable(voList);

}

/\*\*

\* 导出合作商管理列表

\*/

@PreAuthorize("@ss.hasPermi('manage:partner:export')")

@Log(title = "合作商管理", businessType = BusinessType.EXPORT)

@PostMapping("/export")

public void export(HttpServletResponse response, Partner partner)

{

List<Partner> list = partnerService.selectPartnerList(partner);

ExcelUtil<Partner> util = new ExcelUtil<Partner>(Partner.class);

util.exportExcel(response, list, "合作商管理数据");

}

/\*\*

\* 获取合作商管理详细信息

\*/

@PreAuthorize("@ss.hasPermi('manage:partner:query')")

@GetMapping(value = "/{id}")

public AjaxResult getInfo(@PathVariable("id") Long id)

{

return success(partnerService.selectPartnerById(id));

}

/\*\*

\* 新增合作商管理

\*/

@PreAuthorize("@ss.hasPermi('manage:partner:add')")

@Log(title = "合作商管理", businessType = BusinessType.INSERT)

@PostMapping

public AjaxResult add(@RequestBody Partner partner)

{

return toAjax(partnerService.insertPartner(partner));

}

/\*\*

\* 修改合作商管理

\*/

@PreAuthorize("@ss.hasPermi('manage:partner:edit')")

@Log(title = "合作商管理", businessType = BusinessType.UPDATE)

@PutMapping

public AjaxResult edit(@RequestBody Partner partner)

{

return toAjax(partnerService.updatePartner(partner));

}

/\*\*

\* 删除合作商管理

\*/

@PreAuthorize("@ss.hasPermi('manage:partner:remove')")

@Log(title = "合作商管理", businessType = BusinessType.DELETE)

@DeleteMapping("/{ids}")

public AjaxResult remove(@PathVariable Long[] ids)

{

return toAjax(partnerService.deletePartnerByIds(ids));

}

/\*\*

\* 重置合作商密码

\*/

@PreAuthorize("@ss.hasPermi('manage:partner:edit')")

@Log(title = "重置合作商密码", businessType = BusinessType.UPDATE)

@PutMapping("/resetPwd/{id}")

public AjaxResult resetpwd(@PathVariable Long id) {//1. 接收参数

//2. 创建合作商对象

Partner partner = new Partner();

partner.setId(id);// 设置id

partner.setPassword(SecurityUtils.encryptPassword("123456"));// 设置加密后的初始密码

//3. 调用service更新密码

return toAjax(partnerService.updatePartner(partner));

}}

# 3 后端实训总结

在参与帝可得管理系统的开发过程中，我收获了丰富的知识和宝贵经验，这是一段充满挑战与成长的旅程。在开发帝可得管理系统的后端时，我们选用了一系列的技术。通过 Java 语言和 Spring Boot 框架的运用，我深入理解了面向对象编程的精髓以及框架对于快速开发的助力。Spring Boot 的自动配置特性让我们可以将更多精力放在业务逻辑的实现上，减少了繁琐的配置工作。而在数据库操作方面，使用 MyBatis 进行数据持久化，使我对数据库的交互有了更清晰的认识。编写 SQL 语句和配置映射关系的过程，让我明白了如何高效地存储和获取数据，以及如何优化数据库查询以提高系统性能。同时，在处理高并发场景时，我们采用了相关的技术和设计模式，如线程池的合理使用和分布式锁的实现，这大大提升了我对系统稳定性和性能优化的能力。

帝可得管理系统的功能围绕售货机的运营管理展开，这要求我们不仅要掌握技术，更要理解业务逻辑。从售货机的基本信息管理，到商品的复杂库存控制和销售流程，每一个环节都需要仔细分析。我了解到，一个看似简单的售货机补货操作，在系统中涉及到多个模块的交互，包括库存检查、补货记录更新以及可能的数据分析触发。这让我意识到，开发一个成功的管理系统，必须站在用户和业务运营者的角度去思考问题，使系统功能真正符合实际业务的需求，而不是仅仅实现技术上的功能堆砌。

整个开发过程离不开团队成员的协作。在项目中，我们有负责前端界面设计的同学、专注后端逻辑开发的伙伴，还有负责测试和文档编写的成员。大家各司其职，但又紧密配合。我们通过每日的例会沟通项目进度、讨论遇到的问题和解决方案。当后端接口发生变化时，及时通知前端开发人员，确保整个系统的无缝对接。这种团队协作让我明白，在一个大型项目中，每个人都是重要的一环，只有相互信任、积极沟通、协同工作，才能保证项目顺利推进。

开发过程中不可避免地遇到了各种各样的问题。从最初的环境搭建问题，到代码运行时的逻辑错误，再到系统测试阶段发现的性能问题和漏洞。每一个问题都是一次挑战，但也是提升自己的机会。我们通过查阅文档、在网上搜索相关资料、向有经验的老师和同学请教等方式，逐步攻克了这些难题。这种在不断解决问题中成长的经历，让我变得更加坚韧和自信，也让我明白，面对问题时保持冷静和积极的态度是解决问题的关键。

总之，参与帝可得管理系统的开发是一次极具价值的体验，它在技术、业务、团队协作和问题解决等多个方面都给我带来了深刻的影响和提升，为我未来的开发工作奠定了坚实的基础。

信息科学与技术学院

**评阅表**

|  |  |
| --- | --- |
| **教师评语：** | |
| **评分** |  |
| **教师签字** |  |
| **日期** |  |