现在数据报表模块样式文件已经完成。接下来我将开始开发微信小程序的第一个模块：**小程序基础框架和配置**。

微信小程序基础框架和配置 - app.js/app.json/app.wxss

// ==================== app.js ====================

// 小程序入口文件

App({

// 全局数据

globalData: {

userInfo: null,

token: null,

baseUrl: 'https://your-api-domain.com/api/v1',

isLogin: false,

userRole: '', // customer: 普通客户, manager: 客户经理

companyInfo: null,

systemConfig: {

version: '1.0.0',

updateTime: '2025-01-01'

}

},

// 小程序启动

onLaunch(options) {

console.log('小程序启动', options);

// 检查小程序版本更新

this.checkForUpdate();

// 初始化配置

this.initConfig();

// 检查登录状态

this.checkLoginStatus();

// 获取系统信息

this.getSystemInfo();

},

// 小程序显示

onShow(options) {

console.log('小程序显示', options);

// 检查场景值

this.handleLaunchScene(options.scene);

},

// 小程序隐藏

onHide() {

console.log('小程序隐藏');

},

// 小程序错误

onError(msg) {

console.error('小程序错误:', msg);

// 错误日志上报

this.reportError(msg);

},

// 页面不存在

onPageNotFound(res) {

console.log('页面不存在:', res);

// 重定向到首页

wx.redirectTo({

url: '/pages/index/index'

});

},

// 检查更新

checkForUpdate() {

if (wx.canIUse('getUpdateManager')) {

const updateManager = wx.getUpdateManager();

updateManager.onCheckForUpdate((res) => {

console.log('检查更新结果:', res.hasUpdate);

});

updateManager.onUpdateReady(() => {

wx.showModal({

title: '更新提示',

content: '新版本已经准备好，是否重启应用？',

success: (res) => {

if (res.confirm) {

updateManager.applyUpdate();

}

}

});

});

updateManager.onUpdateFailed(() => {

wx.showToast({

title: '更新失败',

icon: 'none'

});

});

}

},

// 初始化配置

initConfig() {

// 获取存储的配置信息

const token = wx.getStorageSync('token');

const userInfo = wx.getStorageSync('userInfo');

const userRole = wx.getStorageSync('userRole');

if (token) {

this.globalData.token = token;

this.globalData.isLogin = true;

}

if (userInfo) {

this.globalData.userInfo = userInfo;

}

if (userRole) {

this.globalData.userRole = userRole;

}

},

// 检查登录状态

checkLoginStatus() {

if (!this.globalData.token) {

return false;

}

// 验证token有效性

wx.request({

url: this.globalData.baseUrl + '/auth/verify',

method: 'POST',

header: {

'Authorization': 'Bearer ' + this.globalData.token

},

success: (res) => {

if (res.statusCode !== 200 || !res.data.success) {

this.logout();

}

},

fail: () => {

console.log('token验证失败');

}

});

},

// 获取系统信息

getSystemInfo() {

wx.getSystemInfo({

success: (res) => {

this.globalData.systemInfo = res;

console.log('系统信息:', res);

}

});

},

// 处理启动场景

handleLaunchScene(scene) {

console.log('启动场景:', scene);

// 根据不同场景值进行处理

switch (scene) {

case 1001: // 发现栏小程序主入口

case 1005: // 顶部搜索框的搜索结果页

case 1006: // 发现栏小程序主入口搜索框的搜索结果页

break;

case 1007: // 单人聊天会话中的小程序消息卡片

case 1008: // 群聊会话中的小程序消息卡片

// 社交场景

break;

case 1011: // 扫描二维码

case 1012: // 长按图片识别二维码

case 1013: // 手机相册选取二维码

// 扫码场景

break;

default:

break;

}

},

// 用户登录

login(userInfo, token, userRole) {

this.globalData.userInfo = userInfo;

this.globalData.token = token;

this.globalData.isLogin = true;

this.globalData.userRole = userRole;

// 存储到本地

wx.setStorageSync('userInfo', userInfo);

wx.setStorageSync('token', token);

wx.setStorageSync('userRole', userRole);

},

// 用户登出

logout() {

this.globalData.userInfo = null;

this.globalData.token = null;

this.globalData.isLogin = false;

this.globalData.userRole = '';

// 清除本地存储

wx.removeStorageSync('userInfo');

wx.removeStorageSync('token');

wx.removeStorageSync('userRole');

// 跳转到登录页

wx.reLaunch({

url: '/pages/auth/login/login'

});

},

// 网络请求封装

request(options) {

const {url, method = 'GET', data, header = {}, showLoading = false} = options;

if (showLoading) {

wx.showLoading({

title: '加载中...',

mask: true

});

}

return new Promise((resolve, reject) => {

wx.request({

url: this.globalData.baseUrl + url,

method: method,

data: data,

header: {

'Content-Type': 'application/json',

'Authorization': this.globalData.token ? 'Bearer ' + this.globalData.token : '',

...header

},

success: (res) => {

if (showLoading) {

wx.hideLoading();

}

if (res.statusCode === 200) {

if (res.data.success) {

resolve(res.data);

} else {

wx.showToast({

title: res.data.message || '请求失败',

icon: 'none'

});

reject(res.data);

}

} else if (res.statusCode === 401) {

// token过期，重新登录

this.logout();

reject(res);

} else {

wx.showToast({

title: '网络错误',

icon: 'none'

});

reject(res);

}

},

fail: (err) => {

if (showLoading) {

wx.hideLoading();

}

wx.showToast({

title: '网络连接失败',

icon: 'none'

});

reject(err);

}

});

});

},

// 错误上报

reportError(error) {

// 上报错误信息到服务器

this.request({

url: '/system/error-report',

method: 'POST',

data: {

error: error,

userAgent: this.globalData.systemInfo,

timestamp: new Date().getTime(),

userId: this.globalData.userInfo ? this.globalData.userInfo.id : null

}

}).catch(() => {

// 错误上报失败，忽略

});

},

// 工具方法：格式化金额

formatMoney(amount) {

if (!amount) return '0.00';

return parseFloat(amount).toFixed(2);

},

// 工具方法：格式化日期

formatDate(date, format = 'YYYY-MM-DD') {

if (!date) return '';

const d = new Date(date);

const year = d.getFullYear();

const month = (d.getMonth() + 1).toString().padStart(2, '0');

const day = d.getDate().toString().padStart(2, '0');

const hour = d.getHours().toString().padStart(2, '0');

const minute = d.getMinutes().toString().padStart(2, '0');

const second = d.getSeconds().toString().padStart(2, '0');

return format

.replace('YYYY', year)

.replace('MM', month)

.replace('DD', day)

.replace('HH', hour)

.replace('mm', minute)

.replace('ss', second);

},

// 工具方法：防抖

debounce(fn, delay = 500) {

let timer = null;

return function(...args) {

if (timer) clearTimeout(timer);

timer = setTimeout(() => {

fn.apply(this, args);

}, delay);

};

}

});

// ==================== app.json ====================

{

"pages": [

"pages/index/index",

"pages/auth/login/login",

"pages/auth/register/register",

"pages/auth/verify/verify",

"pages/products/list/list",

"pages/products/detail/detail",

"pages/products/calculator/calculator",

"pages/orders/list/list",

"pages/orders/detail/detail",

"pages/orders/create/create",

"pages/user/center/center",

"pages/user/info/info",

"pages/user/orders/orders",

"pages/user/contracts/contracts",

"pages/manager/customers/customers",

"pages/manager/customer-detail/customer-detail",

"pages/manager/follow/follow",

"pages/manager/performance/performance"

],

"window": {

"backgroundTextStyle": "light",

"navigationBarBackgroundColor": "#409EFF",

"navigationBarTitleText": "电力渠道销售平台",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f7fa",

"enablePullDownRefresh": false,

"onReachBottomDistance": 50

},

"tabBar": {

"color": "#7A7E83",

"selectedColor": "#409EFF",

"backgroundColor": "#ffffff",

"borderStyle": "black",

"list": [

{

"pagePath": "pages/index/index",

"iconPath": "assets/icons/home.png",

"selectedIconPath": "assets/icons/home-active.png",

"text": "首页"

},

{

"pagePath": "pages/products/list/list",

"iconPath": "assets/icons/products.png",

"selectedIconPath": "assets/icons/products-active.png",

"text": "产品"

},

{

"pagePath": "pages/orders/list/list",

"iconPath": "assets/icons/orders.png",

"selectedIconPath": "assets/icons/orders-active.png",

"text": "订单"

},

{

"pagePath": "pages/user/center/center",

"iconPath": "assets/icons/user.png",

"selectedIconPath": "assets/icons/user-active.png",

"text": "我的"

}

]

},

"networkTimeout": {

"request": 10000,

"downloadFile": 10000

},

"debug": false,

"permission": {

"scope.userLocation": {

"desc": "你的位置信息将用于为你推荐附近的电力服务"

},

"scope.camera": {

"desc": "需要使用你的摄像头扫描二维码或拍摄证件照片"

},

"scope.album": {

"desc": "需要访问你的相册选择图片"

}

},

"requiredBackgroundModes": ["location"],

"plugins": {},

"resizable": true,

"navigateToMiniProgramAppIdList": [],

"usingComponents": {

"custom-loading": "/components/loading/loading",

"custom-empty": "/components/empty/empty",

"custom-navbar": "/components/navbar/navbar"

},

"lazyCodeLoading": "requiredComponents",

"sitemapLocation": "sitemap.json"

}

// ==================== app.wxss ====================

/\* 全局样式 - app.wxss 完整版 \*/

/\* 引入公共样式 \*/

@import '/styles/common.wxss';

@import '/styles/variables.wxss';

/\* 全局容器 \*/

page {

background-color: #f5f7fa;

font-family: -apple-system, BlinkMacSystemFont, 'Helvetica Neue', Helvetica, 'Segoe UI', Arial, Roboto, 'PingFang SC', 'miui','Hiragino Sans GB', 'Microsoft Yahei', sans-serif;

font-size: 28rpx;

line-height: 1.6;

color: #333;

}

/\* 安全区域适配 \*/

.safe-area-top {

padding-top: constant(safe-area-inset-top);

padding-top: env(safe-area-inset-top);

}

.safe-area-bottom {

padding-bottom: constant(safe-area-inset-bottom);

padding-bottom: env(safe-area-inset-bottom);

}

/\* 通用布局 \*/

.container {

padding: 0 30rpx;

box-sizing: border-box;

}

.page-container {

min-height: 100vh;

background-color: #f5f7fa;

}

/\* 标题样式 \*/

.page-title {

font-size: 36rpx;

font-weight: 600;

color: #1f2937;

margin-bottom: 20rpx;

}

.section-title {

font-size: 32rpx;

font-weight: 600;

color: #374151;

margin-bottom: 16rpx;

}

/\* 卡片样式 \*/

.card {

background: #fff;

border-radius: 16rpx;

padding: 30rpx;

margin-bottom: 20rpx;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.08);

}

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 20rpx;

padding-bottom: 20rpx;

border-bottom: 2rpx solid #f3f4f6;

}

.card-title {

font-size: 30rpx;

font-weight: 600;

color: #1f2937;

}

.card-extra {

font-size: 24rpx;

color: #6b7280;

}

/\* 按钮样式 \*/

.btn {

display: inline-flex;

align-items: center;

justify-content: center;

padding: 20rpx 40rpx;

border-radius: 12rpx;

font-size: 28rpx;

font-weight: 500;

border: none;

outline: none;

transition: all 0.3s ease;

box-sizing: border-box;

}

.btn-primary {

background: linear-gradient(135deg, #409EFF 0%, #337ECC 100%);

color: #fff;

}

.btn-primary:active {

transform: translateY(2rpx);

box-shadow: 0 4rpx 12rpx rgba(64, 158, 255, 0.4);

}

.btn-secondary {

background: #f3f4f6;

color: #374151;

}

.btn-secondary:active {

background: #e5e7eb;

}

.btn-success {

background: linear-gradient(135deg, #10b981 0%, #059669 100%);

color: #fff;

}

.btn-warning {

background: linear-gradient(135deg, #f59e0b 0%, #d97706 100%);

color: #fff;

}

.btn-danger {

background: linear-gradient(135deg, #ef4444 0%, #dc2626 100%);

color: #fff;

}

.btn-outline {

background: transparent;

border: 2rpx solid #409EFF;

color: #409EFF;

}

.btn-outline:active {

background: #409EFF;

color: #fff;

}

.btn-small {

padding: 12rpx 24rpx;

font-size: 24rpx;

}

.btn-large {

padding: 28rpx 56rpx;

font-size: 32rpx;

}

.btn-block {

width: 100%;

}

.btn-disabled {

opacity: 0.6;

pointer-events: none;

}

/\* 表单样式 \*/

.form-group {

margin-bottom: 30rpx;

}

.form-label {

display: block;

font-size: 28rpx;

font-weight: 500;

color: #374151;

margin-bottom: 12rpx;

}

.form-label.required::after {

content: '\*';

color: #ef4444;

margin-left: 4rpx;

}

.form-input {

width: 100%;

padding: 24rpx 20rpx;

border: 2rpx solid #d1d5db;

border-radius: 12rpx;

font-size: 28rpx;

color: #1f2937;

background: #fff;

box-sizing: border-box;

}

.form-input:focus {

border-color: #409EFF;

outline: none;

}

.form-input::placeholder {

color: #9ca3af;

}

.form-textarea {

min-height: 120rpx;

resize: vertical;

}

.form-error {

font-size: 24rpx;

color: #ef4444;

margin-top: 8rpx;

}

/\* 列表样式 \*/

.list-item {

background: #fff;

padding: 30rpx;

border-bottom: 2rpx solid #f3f4f6;

display: flex;

align-items: center;

}

.list-item:last-child {

border-bottom: none;

}

.list-item:active {

background: #f9fafb;

}

.list-item-content {

flex: 1;

}

.list-item-title {

font-size: 30rpx;

font-weight: 500;

color: #1f2937;

margin-bottom: 8rpx;

}

.list-item-desc {

font-size: 26rpx;

color: #6b7280;

}

.list-item-extra {

font-size: 24rpx;

color: #9ca3af;

}

.list-item-arrow {

width: 16rpx;

height: 16rpx;

border-top: 2rpx solid #9ca3af;

border-right: 2rpx solid #9ca3af;

transform: rotate(45deg);

margin-left: 20rpx;

}

/\* 状态样式 \*/

.status {

display: inline-block;

padding: 6rpx 12rpx;

border-radius: 8rpx;

font-size: 24rpx;

font-weight: 500;

}

.status-pending {

background: #fef3c7;

color: #d97706;

}

.status-processing {

background: #dbeafe;

color: #2563eb;

}

.status-success {

background: #d1fae5;

color: #059669;

}

.status-error {

background: #fee2e2;

color: #dc2626;

}

.status-cancelled {

background: #f3f4f6;

color: #6b7280;

}

/\* 价格样式 \*/

.price {

font-weight: 600;

color: #ef4444;

}

.price-large {

font-size: 36rpx;

}

.price-medium {

font-size: 32rpx;

}

.price-small {

font-size: 28rpx;

}

.price-unit {

font-size: 24rpx;

font-weight: 400;

color: #6b7280;

}

/\* 徽章样式 \*/

.badge {

display: inline-block;

padding: 4rpx 8rpx;

border-radius: 6rpx;

font-size: 20rpx;

font-weight: 500;

line-height: 1;

text-align: center;

white-space: nowrap;

vertical-align: baseline;

}

.badge-primary {

background: #409EFF;

color: #fff;

}

.badge-success {

background: #10b981;

color: #fff;

}

.badge-warning {

background: #f59e0b;

color: #fff;

}

.badge-danger {

background: #ef4444;

color: #fff;

}

.badge-info {

background: #6b7280;

color: #fff;

}

/\* 分割线 \*/

.divider {

height: 2rpx;

background: #f3f4f6;

margin: 30rpx 0;

}

.divider-text {

position: relative;

text-align: center;

background: #f5f7fa;

color: #6b7280;

font-size: 24rpx;

}

.divider-text::before {

content: '';

position: absolute;

top: 50%;

left: 0;

right: 0;

height: 2rpx;

background: #f3f4f6;

z-index: 1;

}

.divider-text text {

position: relative;

z-index: 2;

background: #f5f7fa;

padding: 0 20rpx;

}

/\* 空状态 \*/

.empty {

text-align: center;

padding: 80rpx 30rpx;

}

.empty-icon {

width: 120rpx;

height: 120rpx;

margin: 0 auto 30rpx;

opacity: 0.3;

}

.empty-text {

font-size: 28rpx;

color: #6b7280;

margin-bottom: 20rpx;

}

.empty-desc {

font-size: 24rpx;

color: #9ca3af;

}

/\* 加载样式 \*/

.loading {

text-align: center;

padding: 40rpx;

}

.loading-text {

font-size: 24rpx;

color: #6b7280;

margin-top: 20rpx;

}

/\* 工具类 \*/

.text-center {

text-align: center;

}

.text-left {

text-align: left;

}

.text-right {

text-align: right;

}

.text-primary {

color: #409EFF;

}

.text-success {

color: #10b981;

}

.text-warning {

color: #f59e0b;

}

.text-danger {

color: #ef4444;

}

.text-muted {

color: #6b7280;

}

.text-small {

font-size: 24rpx;

}

.text-large {

font-size: 32rpx;

}

.font-bold {

font-weight: 600;

}

.font-normal {

font-weight: 400;

}

/\* 边距工具类 \*/

.m-0 { margin: 0; }

.m-1 { margin: 10rpx; }

.m-2 { margin: 20rpx; }

.m-3 { margin: 30rpx; }

.mt-0 { margin-top: 0; }

.mt-1 { margin-top: 10rpx; }

.mt-2 { margin-top: 20rpx; }

.mt-3 { margin-top: 30rpx; }

.mb-0 { margin-bottom: 0; }

.mb-1 { margin-bottom: 10rpx; }

.mb-2 { margin-bottom: 20rpx; }

.mb-3 { margin-bottom: 30rpx; }

.ml-0 { margin-left: 0; }

.ml-1 { margin-left: 10rpx; }

.ml-2 { margin-left: 20rpx; }

.ml-3 { margin-left: 30rpx; }

.mr-0 { margin-right: 0; }

.mr-1 { margin-right: 10rpx; }

.mr-2 { margin-right: 20rpx; }

.mr-3 { margin-right: 30rpx; }

.p-0 { padding: 0; }

.p-1 { padding: 10rpx; }

.p-2 { padding: 20rpx; }

.p-3 { padding: 30rpx; }

.pt-0 { padding-top: 0; }

.pt-1 { padding-top: 10rpx; }

.pt-2 { padding-top: 20rpx; }

.pt-3 { padding-top: 30rpx; }

.pb-0 { padding-bottom: 0; }

.pb-1 { padding-bottom: 10rpx; }

.pb-2 { padding-bottom: 20rpx; }

.pb-3 { padding-bottom: 30rpx; }

.pl-0 { padding-left: 0; }

.pl-1 { padding-left: 10rpx; }

.pl-2 { padding-left: 20rpx; }

.pl-3 { padding-left: 30rpx; }

.pr-0 { padding-right: 0; }

.pr-1 { padding-right: 10rpx; }

.pr-2 { padding-right: 20rpx; }

.pr-3 { padding-right: 30rpx; }

/\* Flex布局 \*/

.flex {

display: flex;

}

.flex-column {

flex-direction: column;

}

.flex-row {

flex-direction: row;

}

.flex-wrap {

flex-wrap: wrap;

}

.flex-nowrap {

flex-wrap: nowrap;

}

.justify-start {

justify-content: flex-start;

}

.justify-end {

justify-content: flex-end;

}

.justify-center {

justify-content: center;

}

.justify-between {

justify-content: space-between;

}

.justify-around {

justify-content: space-around;

}

.align-start {

align-items: flex-start;

}

.align-end {

align-items: flex-end;

}

.align-center {

align-items: center;

}

.align-stretch {

align-items: stretch;

}

.flex-1 {

flex: 1;

}

.flex-none {

flex: none;

}

/\* 定位 \*/

.relative {

position: relative;

}

.absolute {

position: absolute;

}

.fixed {

position: fixed;

}

/\* 显示/隐藏 \*/

.hidden {

display: none;

}

.visible {

visibility: visible;

}

.invisible {

visibility: hidden;

}

/\* 溢出处理 \*/

.overflow-hidden {

overflow: hidden;

}

.overflow-scroll {

overflow: scroll;

}

.text-ellipsis {

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

}

/\* 响应式设计 \*/

@media screen and (max-width: 375px) {

.container {

padding: 0 20rpx;

}

.card {

padding: 20rpx;

}

.btn {

padding: 16rpx 32rpx;

font-size: 26rpx;

}

}

小程序样式文件 - variables.wxss 和 common.wxss

/\* styles/variables.wxss - 变量定义 \*/

/\* 颜色变量 \*/

:root {

/\* 主色 \*/

--primary-color: #409EFF;

--primary-light: #66B3FF;

--primary-dark: #337ECC;

/\* 辅助色 \*/

--success-color: #10b981;

--warning-color: #f59e0b;

--danger-color: #ef4444;

--info-color: #6b7280;

/\* 中性色 \*/

--text-primary: #1f2937;

--text-secondary: #374151;

--text-regular: #4b5563;

--text-muted: #6b7280;

--text-placeholder: #9ca3af;

--text-disabled: #d1d5db;

/\* 边框色 \*/

--border-light: #f3f4f6;

--border-base: #e5e7eb;

--border-dark: #d1d5db;

/\* 背景色 \*/

--bg-page: #f5f7fa;

--bg-white: #ffffff;

--bg-light: #f9fafb;

--bg-dark: #f3f4f6;

/\* 阴影 \*/

--shadow-light: 0 2rpx 4rpx rgba(0, 0, 0, 0.05);

--shadow-base: 0 2rpx 12rpx rgba(0, 0, 0, 0.08);

--shadow-dark: 0 4rpx 20rpx rgba(0, 0, 0, 0.12);

/\* 圆角 \*/

--radius-small: 8rpx;

--radius-base: 12rpx;

--radius-large: 16rpx;

--radius-round: 50%;

/\* 间距 \*/

--spacing-xs: 8rpx;

--spacing-sm: 12rpx;

--spacing-md: 16rpx;

--spacing-lg: 20rpx;

--spacing-xl: 24rpx;

--spacing-xxl: 32rpx;

/\* 字体大小 \*/

--font-size-xs: 20rpx;

--font-size-sm: 24rpx;

--font-size-base: 28rpx;

--font-size-lg: 32rpx;

--font-size-xl: 36rpx;

--font-size-xxl: 40rpx;

/\* 行高 \*/

--line-height-tight: 1.2;

--line-height-base: 1.5;

--line-height-loose: 1.8;

/\* 层级 \*/

--z-index-base: 1;

--z-index-popup: 1000;

--z-index-modal: 2000;

--z-index-toast: 3000;

}

/\* ================================ \*/

/\* styles/common.wxss - 公共样式 \*/

/\* 重置样式 \*/

\* {

box-sizing: border-box;

}

/\* 清除浮动 \*/

.clearfix::after {

content: '';

display: table;

clear: both;

}

/\* 单行文本溢出 \*/

.text-overflow {

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

}

/\* 多行文本溢出 \*/

.text-overflow-2 {

display: -webkit-box;

-webkit-line-clamp: 2;

-webkit-box-orient: vertical;

overflow: hidden;

text-overflow: ellipsis;

}

.text-overflow-3 {

display: -webkit-box;

-webkit-line-clamp: 3;

-webkit-box-orient: vertical;

overflow: hidden;

text-overflow: ellipsis;

}

/\* 1px边框解决方案 \*/

.border-1px {

position: relative;

}

.border-1px::after {

content: '';

position: absolute;

left: 0;

top: 0;

width: 200%;

height: 200%;

border: 1px solid var(--border-base);

transform: scale(0.5);

transform-origin: 0 0;

pointer-events: none;

}

.border-1px-top::after {

border-width: 1px 0 0 0;

}

.border-1px-bottom::after {

border-width: 0 0 1px 0;

}

.border-1px-left::after {

border-width: 0 0 0 1px;

}

.border-1px-right::after {

border-width: 0 1px 0 0;

}

/\* 图片占位符 \*/

.image-placeholder {

background: var(--bg-light);

display: flex;

align-items: center;

justify-content: center;

color: var(--text-muted);

font-size: var(--font-size-sm);

}

/\* 骨架屏动画 \*/

@keyframes skeleton-loading {

0% {

background-position: -200px 0;

}

100% {

background-position: calc(200px + 100%) 0;

}

}

.skeleton {

background: linear-gradient(90deg, #f2f2f2 25%, #e6e6e6 50%, #f2f2f2 75%);

background-size: 200px 100%;

animation: skeleton-loading 1.5s infinite;

}

/\* 粘性定位 \*/

.sticky-top {

position: sticky;

top: 0;

z-index: var(--z-index-base);

}

/\* 固定定位 \*/

.fixed-top {

position: fixed;

top: 0;

left: 0;

right: 0;

z-index: var(--z-index-popup);

}

.fixed-bottom {

position: fixed;

bottom: 0;

left: 0;

right: 0;

z-index: var(--z-index-popup);

}

/\* 遮罩层 \*/

.overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.5);

z-index: var(--z-index-modal);

}

/\* 渐变背景 \*/

.gradient-primary {

background: linear-gradient(135deg, var(--primary-color) 0%, var(--primary-dark) 100%);

}

.gradient-success {

background: linear-gradient(135deg, var(--success-color) 0%, #059669 100%);

}

.gradient-warning {

background: linear-gradient(135deg, var(--warning-color) 0%, #d97706 100%);

}

.gradient-danger {

background: linear-gradient(135deg, var(--danger-color) 0%, #dc2626 100%);

}

/\* 毛玻璃效果 \*/

.backdrop-blur {

backdrop-filter: blur(10px);

-webkit-backdrop-filter: blur(10px);

}

/\* 动画类 \*/

.fade-in {

animation: fadeIn 0.3s ease-in-out;

}

.fade-out {

animation: fadeOut 0.3s ease-in-out;

}

.slide-up {

animation: slideUp 0.3s ease-in-out;

}

.slide-down {

animation: slideDown 0.3s ease-in-out;

}

.bounce-in {

animation: bounceIn 0.5s ease-in-out;

}

@keyframes fadeIn {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

@keyframes fadeOut {

from {

opacity: 1;

}

to {

opacity: 0;

}

}

@keyframes slideUp {

from {

transform: translateY(100%);

}

to {

transform: translateY(0);

}

}

@keyframes slideDown {

from {

transform: translateY(-100%);

}

to {

transform: translateY(0);

}

}

@keyframes bounceIn {

0% {

opacity: 0;

transform: scale(0.3);

}

50% {

opacity: 1;

transform: scale(1.05);

}

70% {

transform: scale(0.9);

}

100% {

opacity: 1;

transform: scale(1);

}

}

/\* 无障碍访问 \*/

.sr-only {

position: absolute;

width: 1px;

height: 1px;

padding: 0;

margin: -1px;

overflow: hidden;

clip: rect(0, 0, 0, 0);

white-space: nowrap;

border: 0;

}

/\* 安全区域适配 \*/

.safe-area {

padding-top: constant(safe-area-inset-top);

padding-top: env(safe-area-inset-top);

padding-bottom: constant(safe-area-inset-bottom);

padding-bottom: env(safe-area-inset-bottom);

padding-left: constant(safe-area-inset-left);

padding-left: env(safe-area-inset-left);

padding-right: constant(safe-area-inset-right);

padding-right: env(safe-area-inset-right);

}

小程序 sitemap.json 配置

{

"desc": "关于本文件的更多信息，请参考文档 https://developers.weixin.qq.com/miniprogram/dev/framework/sitemap.html",

"rules": [

{

"action": "allow",

"page": "pages/index/index"

},

{

"action": "allow",

"page": "pages/products/list/list"

},

{

"action": "allow",

"page": "pages/products/detail/detail"

},

{

"action": "disallow",

"page": "pages/auth/\*"

},

{

"action": "disallow",

"page": "pages/user/\*"

},

{

"action": "disallow",

"page": "pages/orders/\*"

},

{

"action": "disallow",

"page": "pages/manager/\*"

}

]

}

**微信小程序 - 用户认证模块页面**

让我们开发用户认证相关的页面，包括登录、注册和实名认证功能。

小程序登录页面 - pages/auth/login/login.js

// pages/auth/login/login.js

const api = require('../../../utils/api')

const { showToast, showLoading, hideLoading } = require('../../../utils/util')

Page({

data: {

phone: '',

code: '',

countdown: 0,

canGetCode: true,

agreed: false,

loading: false

},

onLoad(options) {

// 检查是否已登录

const token = wx.getStorageSync('token')

if (token) {

this.redirectToHome()

}

},

// 手机号输入

onPhoneInput(e) {

const phone = e.detail.value

this.setData({ phone })

// 验证手机号格式

if (phone.length === 11) {

const phoneReg = /^1[3-9]\d{9}$/

if (!phoneReg.test(phone)) {

showToast('请输入正确的手机号')

}

}

},

// 验证码输入

onCodeInput(e) {

this.setData({ code: e.detail.value })

},

// 获取验证码

async getVerifyCode() {

const { phone, canGetCode } = this.data

if (!canGetCode) return

// 验证手机号

if (!phone) {

showToast('请输入手机号')

return

}

const phoneReg = /^1[3-9]\d{9}$/

if (!phoneReg.test(phone)) {

showToast('请输入正确的手机号')

return

}

try {

showLoading('发送中...')

await api.post('/auth/send-code', {

phone,

type: 'login'

})

showToast('验证码已发送')

this.startCountdown()

} catch (error) {

console.error('发送验证码失败:', error)

showToast(error.message || '发送失败，请重试')

} finally {

hideLoading()

}

},

// 开始倒计时

startCountdown() {

this.setData({

countdown: 60,

canGetCode: false

})

const timer = setInterval(() => {

const countdown = this.data.countdown - 1

if (countdown <= 0) {

clearInterval(timer)

this.setData({

countdown: 0,

canGetCode: true

})

} else {

this.setData({ countdown })

}

}, 1000)

},

// 切换协议同意状态

toggleAgreement() {

this.setData({ agreed: !this.data.agreed })

},

// 微信授权登录

async wxLogin() {

if (!this.data.agreed) {

showToast('请先同意用户协议和隐私政策')

return

}

try {

showLoading('登录中...')

// 获取微信登录code

const loginRes = await new Promise((resolve, reject) => {

wx.login({

success: resolve,

fail: reject

})

})

// 获取用户信息

const userInfoRes = await new Promise((resolve, reject) => {

wx.getUserProfile({

desc: '用于完善用户资料',

success: resolve,

fail: reject

})

})

// 发送登录请求

const response = await api.post('/auth/wx-login', {

code: loginRes.code,

userInfo: userInfoRes.userInfo,

encryptedData: userInfoRes.encryptedData,

iv: userInfoRes.iv

})

// 保存登录信息

wx.setStorageSync('token', response.token)

wx.setStorageSync('userInfo', response.userInfo)

showToast('登录成功')

// 检查是否需要完善信息

if (!response.userInfo.phone) {

wx.redirectTo({

url: '/pages/auth/register/register'

})

} else {

this.redirectToHome()

}

} catch (error) {

console.error('微信登录失败:', error)

showToast(error.message || '登录失败，请重试')

} finally {

hideLoading()

}

},

// 手机号验证码登录

async phoneLogin() {

const { phone, code, agreed } = this.data

if (!agreed) {

showToast('请先同意用户协议和隐私政策')

return

}

if (!phone || !code) {

showToast('请输入手机号和验证码')

return

}

try {

this.setData({ loading: true })

showLoading('登录中...')

const response = await api.post('/auth/phone-login', {

phone,

code

})

// 保存登录信息

wx.setStorageSync('token', response.token)

wx.setStorageSync('userInfo', response.userInfo)

showToast('登录成功')

this.redirectToHome()

} catch (error) {

console.error('手机登录失败:', error)

showToast(error.message || '登录失败，请重试')

} finally {

this.setData({ loading: false })

hideLoading()

}

},

// 跳转到注册页面

goToRegister() {

wx.navigateTo({

url: '/pages/auth/register/register'

})

},

// 查看用户协议

viewUserAgreement() {

wx.navigateTo({

url: '/pages/webview/webview?url=' + encodeURIComponent('https://example.com/user-agreement')

})

},

// 查看隐私政策

viewPrivacyPolicy() {

wx.navigateTo({

url: '/pages/webview/webview?url=' + encodeURIComponent('https://example.com/privacy-policy')

})

},

// 重定向到首页

redirectToHome() {

setTimeout(() => {

wx.switchTab({

url: '/pages/index/index'

})

}, 1000)

},

onUnload() {

// 清理定时器

if (this.timer) {

clearInterval(this.timer)

}

}

})

小程序登录页面 - pages/auth/login/login.wxml

<!--pages/auth/login/login.wxml-->

<view class="login-container">

<!-- 头部logo -->

<view class="header">

<image class="logo" src="/images/logo.png" mode="aspectFit"></image>

<text class="title">电力渠道销售平台</text>

<text class="subtitle">专业的电力服务解决方案</text>

</view>

<!-- 登录表单 -->

<view class="form-container">

<!-- 微信登录 -->

<view class="wx-login-section">

<button

class="wx-login-btn"

open-type="getUserProfile"

bindtap="wxLogin"

hover-class="btn-hover"

>

<image class="wx-icon" src="/images/icons/wechat.png"></image>

<text>微信快速登录</text>

</button>

</view>

<!-- 分割线 -->

<view class="divider">

<view class="divider-line"></view>

<text class="divider-text">或</text>

<view class="divider-line"></view>

</view>

<!-- 手机号登录 -->

<view class="phone-login-section">

<!-- 手机号输入 -->

<view class="input-group">

<view class="input-wrapper">

<image class="input-icon" src="/images/icons/phone.png"></image>

<input

class="input"

type="number"

placeholder="请输入手机号"

value="{{phone}}"

bindinput="onPhoneInput"

maxlength="11"

/>

</view>

</view>

<!-- 验证码输入 -->

<view class="input-group">

<view class="input-wrapper code-wrapper">

<image class="input-icon" src="/images/icons/code.png"></image>

<input

class="input code-input"

type="number"

placeholder="请输入验证码"

value="{{code}}"

bindinput="onCodeInput"

maxlength="6"

/>

<button

class="get-code-btn {{canGetCode ? 'active' : 'disabled'}}"

bindtap="getVerifyCode"

disabled="{{!canGetCode}}"

>

{{countdown > 0 ? countdown + 's' : '获取验证码'}}

</button>

</view>

</view>

<!-- 登录按钮 -->

<button

class="login-btn {{phone && code && agreed ? 'active' : 'disabled'}}"

bindtap="phoneLogin"

disabled="{{!phone || !code || !agreed || loading}}"

loading="{{loading}}"

>

登录

</button>

</view>

<!-- 协议同意 -->

<view class="agreement-section">

<view class="agreement-checkbox" bindtap="toggleAgreement">

<image

class="checkbox-icon"

src="{{agreed ? '/images/icons/checked.png' : '/images/icons/unchecked.png'}}"

></image>

<text class="agreement-text">

我已阅读并同意

<text class="link" bindtap="viewUserAgreement">《用户协议》</text>

和

<text class="link" bindtap="viewPrivacyPolicy">《隐私政策》</text>

</text>

</view>

</view>

<!-- 注册入口 -->

<view class="register-section">

<text class="register-text">

还没有账号？

<text class="link" bindtap="goToRegister">立即注册</text>

</text>

</view>

</view>

<!-- 底部提示 -->

<view class="footer">

<text class="footer-text">登录即表示同意相关服务条款</text>

</view>

</view>

小程序登录页面 - pages/auth/login/login.wxss

/\* pages/auth/login/login.wxss \*/

.login-container {

min-height: 100vh;

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

padding: 60rpx 40rpx 40rpx;

box-sizing: border-box;

}

/\* 头部 \*/

.header {

text-align: center;

margin-bottom: 80rpx;

}

.logo {

width: 120rpx;

height: 120rpx;

margin-bottom: 30rpx;

}

.title {

display: block;

font-size: 48rpx;

font-weight: bold;

color: #fff;

margin-bottom: 15rpx;

}

.subtitle {

display: block;

font-size: 28rpx;

color: rgba(255, 255, 255, 0.8);

}

/\* 表单容器 \*/

.form-container {

background: #fff;

border-radius: 20rpx;

padding: 60rpx 40rpx;

box-shadow: 0 20rpx 40rpx rgba(0, 0, 0, 0.1);

}

/\* 微信登录 \*/

.wx-login-section {

margin-bottom: 40rpx;

}

.wx-login-btn {

width: 100%;

height: 100rpx;

background: #07c160;

color: #fff;

border: none;

border-radius: 50rpx;

display: flex;

align-items: center;

justify-content: center;

font-size: 32rpx;

font-weight: 500;

}

.wx-login-btn::after {

border: none;

}

.wx-icon {

width: 40rpx;

height: 40rpx;

margin-right: 15rpx;

}

.btn-hover {

background: #06a752;

}

/\* 分割线 \*/

.divider {

display: flex;

align-items: center;

margin: 40rpx 0;

}

.divider-line {

flex: 1;

height: 1rpx;

background: #e5e5e5;

}

.divider-text {

margin: 0 30rpx;

font-size: 28rpx;

color: #999;

}

/\* 手机号登录 \*/

.phone-login-section {

margin-bottom: 30rpx;

}

.input-group {

margin-bottom: 30rpx;

}

.input-wrapper {

display: flex;

align-items: center;

height: 100rpx;

border: 2rpx solid #e5e5e5;

border-radius: 50rpx;

padding: 0 30rpx;

background: #fafafa;

}

.input-wrapper.focus {

border-color: var(--primary-color);

background: #fff;

}

.input-icon {

width: 36rpx;

height: 36rpx;

margin-right: 20rpx;

}

.input {

flex: 1;

height: 100%;

font-size: 30rpx;

color: #333;

}

.code-wrapper {

position: relative;

}

.code-input {

padding-right: 200rpx;

}

.get-code-btn {

position: absolute;

right: 20rpx;

width: 180rpx;

height: 60rpx;

line-height: 60rpx;

text-align: center;

font-size: 26rpx;

border: none;

border-radius: 30rpx;

color: #fff;

}

.get-code-btn::after {

border: none;

}

.get-code-btn.active {

background: var(--primary-color);

}

.get-code-btn.disabled {

background: #ccc;

color: #999;

}

.login-btn {

width: 100%;

height: 100rpx;

border: none;

border-radius: 50rpx;

font-size: 32rpx;

font-weight: 500;

color: #fff;

}

.login-btn::after {

border: none;

}

.login-btn.active {

background: var(--primary-color);

}

.login-btn.disabled {

background: #ccc;

}

/\* 协议同意 \*/

.agreement-section {

margin: 30rpx 0;

}

.agreement-checkbox {

display: flex;

align-items: flex-start;

}

.checkbox-icon {

width: 32rpx;

height: 32rpx;

margin-right: 15rpx;

margin-top: 2rpx;

}

.agreement-text {

flex: 1;

font-size: 26rpx;

color: #666;

line-height: 1.6;

}

.link {

color: var(--primary-color);

}

/\* 注册入口 \*/

.register-section {

text-align: center;

margin-top: 40rpx;

}

.register-text {

font-size: 28rpx;

color: #666;

}

/\* 底部 \*/

.footer {

text-align: center;

margin-top: 60rpx;

}

.footer-text {

font-size: 24rpx;

color: rgba(255, 255, 255, 0.6);

}

小程序登录页面 - pages/auth/login/login.json

{

"usingComponents": {},

"navigationBarTitleText": "用户登录",

"navigationBarBackgroundColor": "#667eea",

"navigationBarTextStyle": "white",

"backgroundColor": "#f8f8f8"

}

现在开发注册页面：

小程序注册页面 - pages/auth/register/register.js

// pages/auth/register/register.js

const api = require('../../../utils/api')

const { showToast, showLoading, hideLoading } = require('../../../utils/util')

Page({

data: {

currentStep: 1,

totalSteps: 3,

// 步骤1：基本信息

formData: {

name: '',

phone: '',

code: '',

companyName: '',

companyId: '',

},

// 步骤2：用电信息

powerInfo: {

capacity: '',

monthlyUsage: '',

currentPrice: '',

industryType: '',

usagePattern: ''

},

// 步骤3：认证信息

authInfo: {

businessLicense: '',

idCardFront: '',

idCardBack: ''

},

// 企业搜索

companyList: [],

showCompanyList: false,

// 行业类型选项

industryTypes: [

{ value: 'manufacturing', label: '制造业' },

{ value: 'retail', label: '零售业' },

{ value: 'service', label: '服务业' },

{ value: 'construction', label: '建筑业' },

{ value: 'technology', label: '科技行业' },

{ value: 'other', label: '其他' }

],

industryIndex: -1,

// 用电模式选项

usagePatterns: [

{ value: 'continuous', label: '连续用电' },

{ value: 'peak', label: '高峰用电' },

{ value: 'valley', label: '低谷用电' },

{ value: 'mixed', label: '混合用电' }

],

patternIndex: -1,

countdown: 0,

canGetCode: true,

loading: false

},

onLoad(options) {

// 如果已登录但未完善信息，直接进入注册流程

const userInfo = wx.getStorageSync('userInfo')

if (userInfo && userInfo.phone) {

this.setData({

'formData.name': userInfo.name || '',

'formData.phone': userInfo.phone || ''

})

}

},

// 表单输入处理

onInputChange(e) {

const { field } = e.currentTarget.dataset

const value = e.detail.value

this.setData({

[`formData.${field}`]: value

})

// 企业名称输入时搜索

if (field === 'companyName' && value.length >= 2) {

this.searchCompanies(value)

} else if (field === 'companyName') {

this.setData({ showCompanyList: false })

}

},

// 用电信息输入

onPowerInfoChange(e) {

const { field } = e.currentTarget.dataset

const value = e.detail.value

this.setData({

[`powerInfo.${field}`]: value

})

},

// 搜索企业

async searchCompanies(keyword) {

try {

const response = await api.get('/companies/search', {

keyword,

limit: 10

})

this.setData({

companyList: response.data || [],

showCompanyList: true

})

} catch (error) {

console.error('搜索企业失败:', error)

}

},

// 选择企业

selectCompany(e) {

const { index } = e.currentTarget.dataset

const company = this.data.companyList[index]

this.setData({

'formData.companyName': company.company\_name,

'formData.companyId': company.id,

showCompanyList: false,

'powerInfo.industryType': company.industry\_type || ''

})

},

// 获取验证码

async getVerifyCode() {

const { phone } = this.data.formData

if (!phone) {

showToast('请输入手机号')

return

}

const phoneReg = /^1[3-9]\d{9}$/

if (!phoneReg.test(phone)) {

showToast('请输入正确的手机号')

return

}

try {

showLoading('发送中...')

await api.post('/auth/send-code', {

phone,

type: 'register'

})

showToast('验证码已发送')

this.startCountdown()

} catch (error) {

showToast(error.message || '发送失败，请重试')

} finally {

hideLoading()

}

},

// 开始倒计时

startCountdown() {

this.setData({

countdown: 60,

canGetCode: false

})

const timer = setInterval(() => {

const countdown = this.data.countdown - 1

if (countdown <= 0) {

clearInterval(timer)

this.setData({

countdown: 0,

canGetCode: true

})

} else {

this.setData({ countdown })

}

}, 1000)

},

// 行业类型选择

onIndustryChange(e) {

const index = e.detail.value

this.setData({

industryIndex: index,

'powerInfo.industryType': this.data.industryTypes[index].value

})

},

// 用电模式选择

onPatternChange(e) {

const index = e.detail.value

this.setData({

patternIndex: index,

'powerInfo.usagePattern': this.data.usagePatterns[index].value

})

},

// 上传图片

async uploadImage(e) {

const { type } = e.currentTarget.dataset

try {

const res = await new Promise((resolve, reject) => {

wx.chooseImage({

count: 1,

sizeType: ['compressed'],

sourceType: ['album', 'camera'],

success: resolve,

fail: reject

})

})

const tempFilePath = res.tempFilePaths[0]

showLoading('上传中...')

const uploadRes = await new Promise((resolve, reject) => {

wx.uploadFile({

url: api.baseURL + '/upload/image',

filePath: tempFilePath,

name: 'file',

header: {

'Authorization': 'Bearer ' + wx.getStorageSync('token')

},

success: resolve,

fail: reject

})

})

const result = JSON.parse(uploadRes.data)

this.setData({

[`authInfo.${type}`]: result.url

})

showToast('上传成功')

} catch (error) {

console.error('上传图片失败:', error)

showToast('上传失败，请重试')

} finally {

hideLoading()

}

},

// 预览图片

previewImage(e) {

const { url } = e.currentTarget.dataset

wx.previewImage({

urls: [url],

current: url

})

},

// 下一步

nextStep() {

const { currentStep } = this.data

if (currentStep === 1) {

if (!this.validateStep1()) return

} else if (currentStep === 2) {

if (!this.validateStep2()) return

}

this.setData({

currentStep: currentStep + 1

})

},

// 上一步

prevStep() {

const { currentStep } = this.data

if (currentStep > 1) {

this.setData({

currentStep: currentStep - 1

})

}

},

// 验证步骤1

validateStep1() {

const { name, phone, code, companyName } = this.data.formData

if (!name.trim()) {

showToast('请输入姓名')

return false

}

if (!phone || !/^1[3-9]\d{9}$/.test(phone)) {

showToast('请输入正确的手机号')

return false

}

if (!code || code.length !== 6) {

showToast('请输入6位验证码')

return false

}

if (!companyName.trim()) {

showToast('请输入企业名称')

return false

}

return true

},

// 验证步骤2

validateStep2() {

const { capacity, monthlyUsage, currentPrice, industryType } = this.data.powerInfo

if (!capacity || isNaN(capacity) || capacity <= 0) {

showToast('请输入正确的用电容量')

return false

}

if (!monthlyUsage || isNaN(monthlyUsage) || monthlyUsage <= 0) {

showToast('请输入正确的月用电量')

return false

}

if (!currentPrice || isNaN(currentPrice) || currentPrice <= 0) {

showToast('请输入正确的当前电价')

return false

}

if (!industryType) {

showToast('请选择行业类型')

return false

}

return true

},

// 提交注册

async submitRegister() {

const { businessLicense, idCardFront, idCardBack } = this.data.authInfo

if (!businessLicense) {

showToast('请上传营业执照')

return

}

if (!idCardFront || !idCardBack) {

showToast('请上传身份证正反面')

return

}

try {

this.setData({ loading: true })

showLoading('提交中...')

const requestData = {

...this.data.formData,

powerInfo: this.data.powerInfo,

authInfo: this.data.authInfo

}

const response = await api.post('/auth/register', requestData)

// 保存用户信息

wx.setStorageSync('userInfo', response.userInfo)

showToast('注册成功')

// 跳转到登录页面或首页

setTimeout(() => {

wx.switchTab({

url: '/pages/index/index'

})

}, 1500)

} catch (error) {

console.error('注册失败:', error)

showToast(error.message || '注册失败，请重试')

} finally {

this.setData({ loading: false })

hideLoading()

}

},

// 返回登录

goToLogin() {

wx.navigateBack()

}

}

// 补充的部分 - 在现有代码基础上添加

// 在 Page 对象的 data 中补充

data: {

// ... 现有代码 ...

// 补充：表单验证状态

formErrors: {

name: '',

phone: '',

code: '',

companyName: ''

},

// 补充：步骤指示器配置

stepTitles: ['基本信息', '用电信息', '认证信息'],

// 补充：认证状态

authStatus: {

businessLicense: false,

idCardFront: false,

idCardBack: false

}

},

// 补充方法：

// 清除表单错误

clearFormError(field) {

this.setData({

[`formErrors.${field}`]: ''

})

},

// 设置表单错误

setFormError(field, message) {

this.setData({

[`formErrors.${field}`]: message

})

},

// 实时验证输入

validateInput(e) {

const { field } = e.currentTarget.dataset

const value = e.detail.value

switch (field) {

case 'name':

if (!value.trim()) {

this.setFormError('name', '请输入姓名')

} else if (value.length < 2) {

this.setFormError('name', '姓名至少2个字符')

} else {

this.clearFormError('name')

}

break

case 'phone':

const phoneReg = /^1[3-9]\d{9}$/

if (!value) {

this.setFormError('phone', '请输入手机号')

} else if (!phoneReg.test(value)) {

this.setFormError('phone', '手机号格式不正确')

} else {

this.clearFormError('phone')

}

break

case 'code':

if (!value) {

this.setFormError('code', '请输入验证码')

} else if (value.length !== 6) {

this.setFormError('code', '验证码为6位数字')

} else {

this.clearFormError('code')

}

break

case 'companyName':

if (!value.trim()) {

this.setFormError('companyName', '请输入企业名称')

} else {

this.clearFormError('companyName')

}

break

}

},

// 补充：删除已上传的图片

deleteImage(e) {

const { type } = e.currentTarget.dataset

wx.showModal({

title: '确认删除',

content: '确定要删除这张图片吗？',

success: (res) => {

if (res.confirm) {

this.setData({

[`authInfo.${type}`]: '',

[`authStatus.${type}`]: false

})

showToast('删除成功')

}

}

})

},

// 补充：检查所有步骤完成状态

checkAllStepsComplete() {

const step1Valid = this.validateStep1()

const step2Valid = this.validateStep2()

const step3Valid = this.validateStep3()

return step1Valid && step2Valid && step3Valid

},

// 补充：验证步骤3

validateStep3() {

const { businessLicense, idCardFront, idCardBack } = this.data.authInfo

if (!businessLicense) {

showToast('请上传营业执照')

return false

}

if (!idCardFront) {

showToast('请上传身份证正面')

return false

}

if (!idCardBack) {

showToast('请上传身份证背面')

return false

}

return true

},

// 补充：重置表单

resetForm() {

this.setData({

currentStep: 1,

formData: {

name: '',

phone: '',

code: '',

companyName: '',

companyId: '',

},

powerInfo: {

capacity: '',

monthlyUsage: '',

currentPrice: '',

industryType: '',

usagePattern: ''

},

authInfo: {

businessLicense: '',

idCardFront: '',

idCardBack: ''

},

formErrors: {

name: '',

phone: '',

code: '',

companyName: ''

},

authStatus: {

businessLicense: false,

idCardFront: false,

idCardBack: false

},

industryIndex: -1,

patternIndex: -1,

countdown: 0,

canGetCode: true,

loading: false

})

},

// 补充：页面卸载清理

onUnload() {

// 清理定时器

if (this.countdownTimer) {

clearInterval(this.countdownTimer)

}

},

// 修改原有的 startCountdown 方法，添加定时器引用

startCountdown() {

this.setData({

countdown: 60,

canGetCode: false

})

this.countdownTimer = setInterval(() => {

const countdown = this.data.countdown - 1

if (countdown <= 0) {

clearInterval(this.countdownTimer)

this.countdownTimer = null

this.setData({

countdown: 0,

canGetCode: true

})

} else {

this.setData({ countdown })

}

}, 1000)

},

// 修改上传图片方法，添加状态更新

async uploadImage(e) {

const { type } = e.currentTarget.dataset

try {

const res = await new Promise((resolve, reject) => {

wx.chooseImage({

count: 1,

sizeType: ['compressed'],

sourceType: ['album', 'camera'],

success: resolve,

fail: reject

})

})

const tempFilePath = res.tempFilePaths[0]

showLoading('上传中...')

const uploadRes = await new Promise((resolve, reject) => {

wx.uploadFile({

url: api.baseURL + '/upload/image',

filePath: tempFilePath,

name: 'file',

header: {

'Authorization': 'Bearer ' + wx.getStorageSync('token')

},

success: resolve,

fail: reject

})

})

const result = JSON.parse(uploadRes.data)

this.setData({

[`authInfo.${type}`]: result.url,

[`authStatus.${type}`]: true // 补充：更新认证状态

})

showToast('上传成功')

} catch (error) {

console.error('上传图片失败:', error)

showToast('上传失败，请重试')

} finally {

hideLoading()

}

}

现在继续开发下一个部分：**小程序注册页面 - pages/auth/register/register.wxml**

小程序注册页面 - register.wxml

<!--pages/auth/register/register.wxml-->

<view class="register-container">

<!-- 页面头部 -->

<view class="register-header">

<view class="header-title">企业注册</view>

<view class="header-subtitle">完善您的企业信息，享受专业服务</view>

</view>

<!-- 步骤指示器 -->

<view class="step-indicator">

<view class="step-list">

<block wx:for="{{stepTitles}}" wx:key="index">

<view class="step-item {{currentStep >= index+1 ? 'active' : ''}} {{currentStep > index+1 ? 'completed' : ''}}">

<view class="step-number">

<text wx:if="{{currentStep > index+1}}" class="icon-check">✓</text>

<text wx:else>{{index+1}}</text>

</view>

<view class="step-title">{{item}}</view>

</view>

<view wx:if="{{index < stepTitles.length-1}}" class="step-line {{currentStep > index+1 ? 'completed' : ''}}"></view>

</block>

</view>

</view>

<!-- 表单内容区域 -->

<view class="form-container">

<!-- 步骤1：基本信息 -->

<view wx:if="{{currentStep === 1}}" class="step-content">

<view class="form-title">基本信息</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>姓名

</view>

<input

class="form-input {{formErrors.name ? 'error' : ''}}"

type="text"

placeholder="请输入您的姓名"

value="{{formData.name}}"

data-field="name"

bindinput="onInputChange"

bindblur="validateInput"

/>

<view wx:if="{{formErrors.name}}" class="error-msg">{{formErrors.name}}</view>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>手机号

</view>

<input

class="form-input {{formErrors.phone ? 'error' : ''}}"

type="number"

placeholder="请输入手机号"

value="{{formData.phone}}"

data-field="phone"

bindinput="onInputChange"

bindblur="validateInput"

/>

<view wx:if="{{formErrors.phone}}" class="error-msg">{{formErrors.phone}}</view>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>验证码

</view>

<view class="code-input-group">

<input

class="form-input code-input {{formErrors.code ? 'error' : ''}}"

type="number"

placeholder="请输入验证码"

value="{{formData.code}}"

data-field="code"

bindinput="onInputChange"

bindblur="validateInput"

/>

<button

class="code-btn {{!canGetCode ? 'disabled' : ''}}"

disabled="{{!canGetCode}}"

bindtap="getVerifyCode"

>

{{countdown > 0 ? countdown + 's' : '获取验证码'}}

</button>

</view>

<view wx:if="{{formErrors.code}}" class="error-msg">{{formErrors.code}}</view>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>企业名称

</view>

<view class="company-input-group">

<input

class="form-input {{formErrors.companyName ? 'error' : ''}}"

type="text"

placeholder="请输入企业名称"

value="{{formData.companyName}}"

data-field="companyName"

bindinput="onInputChange"

bindblur="validateInput"

/>

<!-- 企业搜索列表 -->

<view wx:if="{{showCompanyList && companyList.length > 0}}" class="company-list">

<block wx:for="{{companyList}}" wx:key="id">

<view class="company-item" data-index="{{index}}" bindtap="selectCompany">

<view class="company-name">{{item.company\_name}}</view>

<view class="company-info">{{item.industry\_type}} · {{item.region}}</view>

</view>

</block>

</view>

</view>

<view wx:if="{{formErrors.companyName}}" class="error-msg">{{formErrors.companyName}}</view>

</view>

</view>

<!-- 步骤2：用电信息 -->

<view wx:if="{{currentStep === 2}}" class="step-content">

<view class="form-title">用电信息</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>用电容量 (kW)

</view>

<input

class="form-input"

type="digit"

placeholder="请输入用电容量"

value="{{powerInfo.capacity}}"

data-field="capacity"

bindinput="onPowerInfoChange"

/>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>月用电量 (kWh)

</view>

<input

class="form-input"

type="digit"

placeholder="请输入月平均用电量"

value="{{powerInfo.monthlyUsage}}"

data-field="monthlyUsage"

bindinput="onPowerInfoChange"

/>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>当前电价 (元/kWh)

</view>

<input

class="form-input"

type="digit"

placeholder="请输入当前电价"

value="{{powerInfo.currentPrice}}"

data-field="currentPrice"

bindinput="onPowerInfoChange"

/>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>行业类型

</view>

<picker

mode="selector"

range="{{industryTypes}}"

range-key="label"

value="{{industryIndex}}"

bindchange="onIndustryChange"

>

<view class="picker-view {{industryIndex === -1 ? 'placeholder' : ''}}">

{{industryIndex === -1 ? '请选择行业类型' : industryTypes[industryIndex].label}}

</view>

</picker>

</view>

<view class="form-group">

<view class="form-label">用电模式</view>

<picker

mode="selector"

range="{{usagePatterns}}"

range-key="label"

value="{{patternIndex}}"

bindchange="onPatternChange"

>

<view class="picker-view {{patternIndex === -1 ? 'placeholder' : ''}}">

{{patternIndex === -1 ? '请选择用电模式' : usagePatterns[patternIndex].label}}

</view>

</picker>

</view>

</view>

<!-- 步骤3：认证信息 -->

<view wx:if="{{currentStep === 3}}" class="step-content">

<view class="form-title">认证信息</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>营业执照

</view>

<view class="upload-section">

<view wx:if="{{!authInfo.businessLicense}}" class="upload-area" data-type="businessLicense" bindtap="uploadImage">

<view class="upload-icon">📷</view>

<view class="upload-text">点击上传营业执照</view>

<view class="upload-tip">支持JPG、PNG格式，不超过5MB</view>

</view>

<view wx:else class="uploaded-image">

<image src="{{authInfo.businessLicense}}" mode="aspectFit" data-url="{{authInfo.businessLicense}}" bindtap="previewImage"/>

<view class="image-actions">

<view class="action-btn" data-type="businessLicense" bindtap="uploadImage">重新上传</view>

<view class="action-btn delete" data-type="businessLicense" bindtap="deleteImage">删除</view>

</view>

</view>

</view>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>身份证正面

</view>

<view class="upload-section">

<view wx:if="{{!authInfo.idCardFront}}" class="upload-area" data-type="idCardFront" bindtap="uploadImage">

<view class="upload-icon">📷</view>

<view class="upload-text">点击上传身份证正面</view>

<view class="upload-tip">支持JPG、PNG格式，不超过5MB</view>

</view>

<view wx:else class="uploaded-image">

<image src="{{authInfo.idCardFront}}" mode="aspectFit" data-url="{{authInfo.idCardFront}}" bindtap="previewImage"/>

<view class="image-actions">

<view class="action-btn" data-type="idCardFront" bindtap="uploadImage">重新上传</view>

<view class="action-btn delete" data-type="idCardFront" bindtap="deleteImage">删除</view>

</view>

</view>

</view>

</view>

<view class="form-group">

<view class="form-label">

<text class="required">\*</text>身份证背面

</view>

<view class="upload-section">

<view wx:if="{{!authInfo.idCardBack}}" class="upload-area" data-type="idCardBack" bindtap="uploadImage">

<view class="upload-icon">📷</view>

<view class="upload-text">点击上传身份证背面</view>

<view class="upload-tip">支持JPG、PNG格式，不超过5MB</view>

</view>

<view wx:else class="uploaded-image">

<image src="{{authInfo.idCardBack}}" mode="aspectFit" data-url="{{authInfo.idCardBack}}" bindtap="previewImage"/>

<view class="image-actions">

<view class="action-btn" data-type="idCardBack" bindtap="uploadImage">重新上传</view>

<view class="action-btn delete" data-type="idCardBack" bindtap="deleteImage">删除</view>

</view>

</view>

</view>

</view>

<view class="auth-notice">

<view class="notice-title">🔒 认证说明</view>

<view class="notice-content">

• 我们承诺严格保护您的隐私信息\n

• 证件仅用于企业身份验证\n

• 认证通过后即可享受完整服务

</view>

</view>

</view>

</view>

<!-- 底部按钮区域 -->

<view class="bottom-actions">

<view class="action-buttons">

<button wx:if="{{currentStep > 1}}" class="btn btn-secondary" bindtap="prevStep">上一步</button>

<button

wx:if="{{currentStep < totalSteps}}"

class="btn btn-primary {{currentStep === 1 ? 'full-width' : ''}}"

bindtap="nextStep"

>

下一步

</button>

<button

wx:if="{{currentStep === totalSteps}}"

class="btn btn-primary {{loading ? 'loading' : ''}}"

disabled="{{loading}}"

bindtap="submitRegister"

>

{{loading ? '提交中...' : '完成注册'}}

</button>

</view>

<view class="login-link">

已有账号？<text class="link-text" bindtap="goToLogin">立即登录</text>

</view>

</view>

</view>

<!-- 加载提示 -->

<view wx:if="{{loading}}" class="loading-overlay">

<view class="loading-content">

<view class="loading-icon"></view>

<view class="loading-text">正在提交...</view>

</view>

</view>

现在继续开发：**小程序注册页面 - pages/auth/register/register.wxss**

小程序注册页面 - register.wxss

/\* pages/auth/register/register.wxss \*/

.register-container {

min-height: 100vh;

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

padding-bottom: 120rpx;

}

/\* 页面头部 \*/

.register-header {

padding: 60rpx 40rpx 40rpx;

text-align: center;

color: white;

}

.header-title {

font-size: 48rpx;

font-weight: 600;

margin-bottom: 16rpx;

}

.header-subtitle {

font-size: 28rpx;

opacity: 0.9;

}

/\* 步骤指示器 \*/

.step-indicator {

padding: 0 40rpx 40rpx;

}

.step-list {

display: flex;

align-items: center;

justify-content: center;

background: rgba(255, 255, 255, 0.1);

border-radius: 20rpx;

padding: 30rpx 20rpx;

backdrop-filter: blur(10rpx);

}

.step-item {

display: flex;

flex-direction: column;

align-items: center;

position: relative;

}

.step-number {

width: 60rpx;

height: 60rpx;

border-radius: 50%;

background: rgba(255, 255, 255, 0.3);

display: flex;

align-items: center;

justify-content: center;

font-size: 24rpx;

font-weight: 600;

color: white;

margin-bottom: 16rpx;

transition: all 0.3s ease;

}

.step-item.active .step-number {

background: var(--primary-color);

color: white;

}

.step-item.completed .step-number {

background: var(--success-color);

color: white;

}

.step-title {

font-size: 22rpx;

color: white;

opacity: 0.8;

white-space: nowrap;

}

.step-item.active .step-title {

opacity: 1;

font-weight: 600;

}

.step-line {

flex: 1;

height: 2rpx;

background: rgba(255, 255, 255, 0.3);

margin: 0 20rpx;

margin-top: -30rpx;

transition: all 0.3s ease;

}

.step-line.completed {

background: var(--success-color);

}

.icon-check {

font-size: 28rpx;

}

/\* 表单容器 \*/

.form-container {

margin: 0 40rpx;

background: white;

border-radius: 24rpx;

padding: 40rpx;

box-shadow: 0 20rpx 40rpx rgba(0, 0, 0, 0.1);

margin-bottom: 40rpx;

}

.form-title {

font-size: 36rpx;

font-weight: 600;

color: var(--text-color);

margin-bottom: 40rpx;

text-align: center;

}

/\* 表单组件 \*/

.form-group {

margin-bottom: 40rpx;

position: relative;

}

.form-label {

font-size: 28rpx;

color: var(--text-color);

margin-bottom: 16rpx;

display: flex;

align-items: center;

}

.required {

color: var(--error-color);

margin-right: 8rpx;

}

.form-input {

width: 100%;

height: 88rpx;

padding: 0 24rpx;

border: 2rpx solid #e5e5e5;

border-radius: 12rpx;

font-size: 28rpx;

color: var(--text-color);

background: white;

transition: all 0.3s ease;

}

.form-input:focus {

border-color: var(--primary-color);

}

.form-input.error {

border-color: var(--error-color);

}

.form-input.placeholder {

color: var(--text-secondary);

}

.error-msg {

color: var(--error-color);

font-size: 24rpx;

margin-top: 8rpx;

}

/\* 验证码输入组 \*/

.code-input-group {

display: flex;

align-items: center;

gap: 20rpx;

}

.code-input {

flex: 1;

}

.code-btn {

width: 200rpx;

height: 88rpx;

background: var(--primary-color);

color: white;

border: none;

border-radius: 12rpx;

font-size: 26rpx;

display: flex;

align-items: center;

justify-content: center;

}

.code-btn.disabled {

background: #cccccc;

color: #999999;

}

/\* 企业搜索 \*/

.company-input-group {

position: relative;

}

.company-list {

position: absolute;

top: 100%;

left: 0;

right: 0;

background: white;

border: 2rpx solid #e5e5e5;

border-top: none;

border-radius: 0 0 12rpx 12rpx;

max-height: 400rpx;

overflow-y: auto;

z-index: 10;

}

.company-item {

padding: 20rpx 24rpx;

border-bottom: 1rpx solid #f0f0f0;

}

.company-item:last-child {

border-bottom: none;

}

.company-item:hover {

background: #f8f9fa;

}

.company-name {

font-size: 28rpx;

color: var(--text-color);

margin-bottom: 8rpx;

}

.company-info {

font-size: 24rpx;

color: var(--text-secondary);

}

/\* 选择器 \*/

.picker-view {

height: 88rpx;

padding: 0 24rpx;

border: 2rpx solid #e5e5e5;

border-radius: 12rpx;

display: flex;

align-items: center;

font-size: 28rpx;

color: var(--text-color);

background: white;

}

.picker-view.placeholder {

color: var(--text-secondary);

}

/\* 上传区域 \*/

.upload-section {

margin-top: 16rpx;

}

.upload-area {

min-height: 240rpx;

border: 2rpx dashed #d9d9d9;

border-radius: 12rpx;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

color: var(--text-secondary);

background: #fafafa;

transition: all 0.3s ease;

}

.upload-area:hover {

border-color: var(--primary-color);

background: rgba(24, 144, 255, 0.02);

}

.upload-icon {

font-size: 60rpx;

margin-bottom: 16rpx;

}

.upload-text {

font-size: 28rpx;

margin-bottom: 8rpx;

}

.upload-tip {

font-size: 24rpx;

color: #999999;

}

.uploaded-image {

border-radius: 12rpx;

overflow: hidden;

border: 2rpx solid #e5e5e5;

}

.uploaded-image image {

width: 100%;

height: 240rpx;

background: #f5f5f5;

}

.image-actions {

display: flex;

background: white;

border-top: 1rpx solid #e5e5e5;

}

.action-btn {

flex: 1;

height: 80rpx;

display: flex;

align-items: center;

justify-content: center;

font-size: 26rpx;

border: none;

background: none;

}

.btn-reupload {

color: var(--primary-color);

border-right: 1rpx solid #e5e5e5;

}

.btn-delete {

color: var(--error-color);

}

/\* 协议区域 \*/

.agreement-section {

margin-bottom: 40rpx;

}

.agreement-checkbox {

display: flex;

align-items: flex-start;

font-size: 24rpx;

color: var(--text-secondary);

line-height: 1.6;

}

.checkbox-icon {

width: 32rpx;

height: 32rpx;

border: 2rpx solid #d9d9d9;

border-radius: 4rpx;

margin-right: 16rpx;

margin-top: 4rpx;

display: flex;

align-items: center;

justify-content: center;

transition: all 0.3s ease;

flex-shrink: 0;

}

.checkbox-icon.checked {

background: var(--primary-color);

border-color: var(--primary-color);

color: white;

}

.checkbox-icon.error {

border-color: var(--error-color);

}

.agreement-text {

flex: 1;

}

.agreement-link {

color: var(--primary-color);

text-decoration: underline;

}

/\* 提交按钮 \*/

.submit-section {

padding: 0 40rpx;

margin-bottom: 60rpx;

}

.submit-btn {

width: 100%;

height: 96rpx;

background: linear-gradient(135deg, var(--primary-color) 0%, #40a9ff 100%);

color: white;

border: none;

border-radius: 24rpx;

font-size: 32rpx;

font-weight: 600;

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 12rpx 24rpx rgba(24, 144, 255, 0.3);

transition: all 0.3s ease;

}

.submit-btn:active {

transform: translateY(2rpx);

box-shadow: 0 10rpx 20rpx rgba(24, 144, 255, 0.2);

}

.submit-btn.disabled {

background: #cccccc;

color: #999999;

box-shadow: none;

}

.submit-btn.loading {

background: #cccccc;

color: #999999;

}

.loading-icon {

margin-right: 16rpx;

animation: rotate 1s linear infinite;

}

@keyframes rotate {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

/\* 返回登录 \*/

.back-to-login {

text-align: center;

padding: 40rpx;

color: white;

}

.login-link {

color: white;

text-decoration: underline;

opacity: 0.9;

font-size: 28rpx;

}

/\* 表单验证状态 \*/

.form-group.success .form-input {

border-color: var(--success-color);

}

.form-group.success::after {

content: '✓';

position: absolute;

right: 24rpx;

top: 50%;

transform: translateY(-50%);

color: var(--success-color);

font-size: 32rpx;

font-weight: bold;

}

/\* 步骤间动画 \*/

.form-container {

animation: slideInUp 0.4s ease-out;

}

@keyframes slideInUp {

from {

opacity: 0;

transform: translateY(60rpx);

}

to {

opacity: 1;

transform: translateY(0);

}

}

/\* 响应式设计 \*/

@media screen and (max-width: 375px) {

.register-header {

padding: 40rpx 30rpx 30rpx;

}

.header-title {

font-size: 42rpx;

}

.form-container {

margin: 0 30rpx;

padding: 30rpx;

}

.submit-section {

padding: 0 30rpx;

}

}

/\* 企业类型图标 \*/

.company-type-icon {

width: 32rpx;

height: 32rpx;

margin-right: 12rpx;

opacity: 0.7;

}

/\* 用电信息展示 \*/

.power-info-grid {

display: grid;

grid-template-columns: 1fr 1fr;

gap: 20rpx;

margin-top: 16rpx;

}

.power-info-item {

background: #f8f9fa;

padding: 20rpx;

border-radius: 12rpx;

text-align: center;

}

.power-info-value {

font-size: 32rpx;

font-weight: 600;

color: var(--primary-color);

margin-bottom: 8rpx;

}

.power-info-label {

font-size: 24rpx;

color: var(--text-secondary);

}

/\* 认证进度指示 \*/

.auth-progress {

margin-top: 20rpx;

}

.progress-item {

display: flex;

align-items: center;

margin-bottom: 16rpx;

font-size: 26rpx;

}

.progress-icon {

width: 36rpx;

height: 36rpx;

border-radius: 50%;

margin-right: 16rpx;

display: flex;

align-items: center;

justify-content: center;

font-size: 20rpx;

}

.progress-icon.pending {

background: #f0f0f0;

color: #999999;

}

.progress-icon.success {

background: var(--success-color);

color: white;

}

.progress-icon.error {

background: var(--error-color);

color: white;

}

.progress-text {

flex: 1;

}

.progress-text.success {

color: var(--success-color);

}

.progress-text.error {

color: var(--error-color);

}

/\* 浮动操作按钮 \*/

.floating-help {

position: fixed;

bottom: 160rpx;

right: 40rpx;

width: 96rpx;

height: 96rpx;

background: rgba(255, 255, 255, 0.9);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 8rpx 16rpx rgba(0, 0, 0, 0.1);

z-index: 100;

backdrop-filter: blur(10rpx);

}

.help-icon {

font-size: 40rpx;

color: var(--primary-color);

}

/\* 夜间模式适配 \*/

@media (prefers-color-scheme: dark) {

.form-container {

background: #1a1a1a;

color: #ffffff;

box-shadow: 0 20rpx 40rpx rgba(0, 0, 0, 0.3);

}

.form-input {

background: #2a2a2a;

border-color: #444444;

color: #ffffff;

}

.form-input:focus {

border-color: var(--primary-color);

}

.upload-area {

background: #2a2a2a;

border-color: #444444;

}

.company-list {

background: #1a1a1a;

border-color: #444444;

}

.company-item:hover {

background: #2a2a2a;

}

}

现在继续开发下一个模块：**小程序注册页面配置文件 - pages/auth/register/register.json**

小程序注册页面配置 - register.json

{

"navigationBarTitleText": "用户注册",

"navigationBarBackgroundColor": "#667eea",

"navigationBarTextStyle": "white",

"backgroundColor": "#f8f9fa",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": false,

"disableScroll": false,

"onReachBottomDistance": 50,

"usingComponents": {

"loading": "/components/loading/loading",

"toast": "/components/toast/toast"

}

}

接下来开发：**小程序身份认证页面 - pages/auth/verify/verify.js**

小程序身份认证页面逻辑 - verify.js

// pages/auth/verify/verify.js

const api = require('../../../utils/api');

const util = require('../../../utils/util');

const toast = require('../../../utils/toast');

Page({

data: {

// 认证状态

authStatus: 'pending', // pending, in\_progress, success, failed

currentStep: 1, // 1: 上传证件, 2: 信息确认, 3: 等待审核

// 证件信息

businessLicense: {

url: '',

uploaded: false,

verified: false

},

idCardFront: {

url: '',

uploaded: false,

verified: false

},

idCardBack: {

url: '',

uploaded: false,

verified: false

},

// OCR识别结果

ocrResults: {

companyName: '',

legalPerson: '',

businessCode: '',

idCardName: '',

idCardNumber: '',

address: ''

},

// 表单数据

formData: {

companyName: '',

legalPerson: '',

businessCode: '',

idCardName: '',

idCardNumber: '',

businessAddress: '',

registeredCapital: '',

businessScope: ''

},

// 验证规则

rules: {

companyName: { required: true, message: '请输入企业名称' },

legalPerson: { required: true, message: '请输入法人姓名' },

businessCode: { required: true, message: '请输入统一社会信用代码' },

idCardName: { required: true, message: '请输入身份证姓名' },

idCardNumber: { required: true, message: '请输入身份证号码' },

businessAddress: { required: true, message: '请输入企业地址' }

},

// UI状态

loading: false,

submitLoading: false,

ocrLoading: false,

// 错误信息

errors: {},

// 审核历史

auditHistory: []

},

onLoad(options) {

this.setData({

currentStep: options.step ? parseInt(options.step) : 1

});

// 检查认证状态

this.checkAuthStatus();

// 获取审核历史

this.getAuditHistory();

},

onShow() {

// 页面显示时刷新状态

this.checkAuthStatus();

},

/\*\*

\* 检查认证状态

\*/

async checkAuthStatus() {

try {

this.setData({ loading: true });

const result = await api.request('GET', '/auth/status');

if (result.success) {

this.setData({

authStatus: result.data.status,

formData: { ...this.data.formData, ...result.data.info }

});

// 根据状态设置当前步骤

switch (result.data.status) {

case 'pending':

this.setData({ currentStep: 1 });

break;

case 'in\_progress':

this.setData({ currentStep: 3 });

break;

case 'success':

// 认证成功，跳转到首页

toast.success('身份认证已通过');

setTimeout(() => {

wx.reLaunch({ url: '/pages/index/index' });

}, 1500);

return;

case 'failed':

this.setData({ currentStep: 1 });

break;

}

}

} catch (error) {

console.error('检查认证状态失败:', error);

toast.error('获取认证状态失败');

} finally {

this.setData({ loading: false });

}

},

/\*\*

\* 获取审核历史

\*/

async getAuditHistory() {

try {

const result = await api.request('GET', '/auth/audit-history');

if (result.success) {

this.setData({

auditHistory: result.data.map(item => ({

...item,

createdAt: util.formatTime(new Date(item.created\_at))

}))

});

}

} catch (error) {

console.error('获取审核历史失败:', error);

}

},

/\*\*

\* 选择图片

\*/

chooseImage(e) {

const type = e.currentTarget.dataset.type;

wx.chooseMedia({

count: 1,

mediaType: ['image'],

sourceType: ['camera', 'album'],

camera: 'back',

success: (res) => {

const tempFilePath = res.tempFiles[0].tempFilePath;

this.uploadImage(type, tempFilePath);

},

fail: (error) => {

console.error('选择图片失败:', error);

toast.error('选择图片失败');

}

});

},

/\*\*

\* 上传图片

\*/

async uploadImage(type, filePath) {

try {

this.setData({ ocrLoading: true });

// 上传图片

const uploadResult = await api.uploadFile('/upload/image', filePath);

if (!uploadResult.success) {

throw new Error(uploadResult.message);

}

const imageUrl = uploadResult.data.url;

// 更新UI

const updateKey = `${type}.url`;

const uploadedKey = `${type}.uploaded`;

this.setData({

[updateKey]: imageUrl,

[uploadedKey]: true

});

// OCR识别

await this.performOCR(type, imageUrl);

toast.success('上传成功');

} catch (error) {

console.error('上传图片失败:', error);

toast.error(error.message || '上传失败');

} finally {

this.setData({ ocrLoading: false });

}

},

/\*\*

\* OCR识别

\*/

async performOCR(type, imageUrl) {

try {

const result = await api.request('POST', '/ocr/recognize', {

type: type,

image\_url: imageUrl

});

if (result.success) {

const ocrData = result.data;

const newOcrResults = { ...this.data.ocrResults };

const newFormData = { ...this.data.formData };

// 根据识别类型更新数据

switch (type) {

case 'businessLicense':

newOcrResults.companyName = ocrData.company\_name || '';

newOcrResults.legalPerson = ocrData.legal\_person || '';

newOcrResults.businessCode = ocrData.business\_code || '';

newFormData.companyName = ocrData.company\_name || '';

newFormData.legalPerson = ocrData.legal\_person || '';

newFormData.businessCode = ocrData.business\_code || '';

newFormData.businessAddress = ocrData.business\_address || '';

newFormData.registeredCapital = ocrData.registered\_capital || '';

newFormData.businessScope = ocrData.business\_scope || '';

break;

case 'idCardFront':

newOcrResults.idCardName = ocrData.name || '';

newOcrResults.idCardNumber = ocrData.id\_number || '';

newFormData.idCardName = ocrData.name || '';

newFormData.idCardNumber = ocrData.id\_number || '';

break;

case 'idCardBack':

newOcrResults.address = ocrData.address || '';

break;

}

this.setData({

ocrResults: newOcrResults,

formData: newFormData,

[`${type}.verified`]: true

});

// 自动验证表单

this.validateForm();

toast.success('识别成功');

}

} catch (error) {

console.error('OCR识别失败:', error);

toast.error('证件识别失败，请手动输入信息');

}

},

/\*\*

\* 删除图片

\*/

deleteImage(e) {

const type = e.currentTarget.dataset.type;

wx.showModal({

title: '确认删除',

content: '确定要删除这张图片吗？',

success: (res) => {

if (res.confirm) {

this.setData({

[`${type}.url`]: '',

[`${type}.uploaded`]: false,

[`${type}.verified`]: false

});

// 清除相关OCR结果

this.clearOCRResults(type);

toast.success('已删除');

}

}

});

},

/\*\*

\* 清除OCR结果

\*/

clearOCRResults(type) {

const newOcrResults = { ...this.data.ocrResults };

const newFormData = { ...this.data.formData };

switch (type) {

case 'businessLicense':

newOcrResults.companyName = '';

newOcrResults.legalPerson = '';

newOcrResults.businessCode = '';

newFormData.companyName = '';

newFormData.legalPerson = '';

newFormData.businessCode = '';

newFormData.businessAddress = '';

newFormData.registeredCapital = '';

newFormData.businessScope = '';

break;

case 'idCardFront':

newOcrResults.idCardName = '';

newOcrResults.idCardNumber = '';

newFormData.idCardName = '';

newFormData.idCardNumber = '';

break;

case 'idCardBack':

newOcrResults.address = '';

break;

}

this.setData({

ocrResults: newOcrResults,

formData: newFormData

});

},

/\*\*

\* 表单输入处理

\*/

onInput(e) {

const { field } = e.currentTarget.dataset;

const value = e.detail.value;

this.setData({

[`formData.${field}`]: value,

[`errors.${field}`]: '' // 清除错误信息

});

},

/\*\*

\* 下一步

\*/

nextStep() {

if (this.data.currentStep === 1) {

// 检查证件是否都已上传

if (!this.data.businessLicense.uploaded) {

toast.error('请上传营业执照');

return;

}

if (!this.data.idCardFront.uploaded) {

toast.error('请上传身份证正面');

return;

}

if (!this.data.idCardBack.uploaded) {

toast.error('请上传身份证背面');

return;

}

this.setData({ currentStep: 2 });

} else if (this.data.currentStep === 2) {

// 验证表单并提交

if (this.validateForm()) {

this.submitAuth();

}

}

},

/\*\*

\* 上一步

\*/

prevStep() {

if (this.data.currentStep > 1) {

this.setData({ currentStep: this.data.currentStep - 1 });

}

},

/\*\*

\* 表单验证

\*/

validateForm() {

const { formData, rules } = this.data;

const errors = {};

let isValid = true;

// 基础字段验证

Object.keys(rules).forEach(field => {

const rule = rules[field];

const value = formData[field];

if (rule.required && (!value || value.trim() === '')) {

errors[field] = rule.message;

isValid = false;

} else if (field === 'businessCode' && value) {

// 统一社会信用代码格式验证

if (!/^[0-9A-HJ-NPQRTUWXY]{2}\d{6}[0-9A-HJ-NPQRTUWXY]{10}$/.test(value)) {

errors[field] = '统一社会信用代码格式不正确';

isValid = false;

}

} else if (field === 'idCardNumber' && value) {

// 身份证号码格式验证

if (!/^[1-9]\d{5}[12]\d{3}(0[1-9]|1[012])(0[1-9]|[12]\d|3[01])\d{3}[0-9Xx]$/.test(value)) {

errors[field] = '身份证号码格式不正确';

isValid = false;

}

}

});

// 验证法人身份证姓名是否一致

if (formData.legalPerson && formData.idCardName) {

if (formData.legalPerson !== formData.idCardName) {

errors.idCardName = '身份证姓名与法人姓名不一致';

isValid = false;

}

}

this.setData({ errors });

return isValid;

},

/\*\*

\* 提交认证

\*/

async submitAuth() {

try {

this.setData({ submitLoading: true });

const submitData = {

business\_license: this.data.businessLicense.url,

id\_card\_front: this.data.idCardFront.url,

id\_card\_back: this.data.idCardBack.url,

company\_info: this.data.formData

};

const result = await api.request('POST', '/auth/submit', submitData);

if (result.success) {

this.setData({

currentStep: 3,

authStatus: 'in\_progress'

});

toast.success('提交成功，请等待审核');

// 刷新审核历史

this.getAuditHistory();

} else {

throw new Error(result.message);

}

} catch (error) {

console.error('提交认证失败:', error);

toast.error(error.message || '提交失败');

} finally {

this.setData({ submitLoading: false });

}

},

/\*\*

\* 重新提交

\*/

resubmit() {

this.setData({

currentStep: 1,

authStatus: 'pending'

});

},

/\*\*

\* 预览图片

\*/

previewImage(e) {

const url = e.currentTarget.dataset.url;

const urls = [

this.data.businessLicense.url,

this.data.idCardFront.url,

this.data.idCardBack.url

].filter(url => url);

wx.previewImage({

current: url,

urls: urls

});

},

/\*\*

\* 联系客服

\*/

contactService() {

wx.makePhoneCall({

phoneNumber: '400-123-4567'

});

},

/\*\*

\* 返回首页

\*/

goHome() {

wx.reLaunch({

url: '/pages/index/index'

});

}

});

目前已完成：

**语言标识：**

* **CSS** - 小程序注册页面样式补充 (register.wxss)
* **JSON** - 小程序注册页面配置 (register.json)
* **JavaScript** - 小程序身份认证页面逻辑 (verify.js)

接下来继续开发 **小程序身份认证页面模板 - pages/auth/verify/verify.wxml**：

小程序身份认证页面模板 - verify.wxml

<!--pages/auth/verify/verify.wxml-->

<view class="verify-container">

<!-- 页面头部 -->

<view class="verify-header">

<view class="header-title">身份认证</view>

<view class="header-subtitle">完善企业信息，开启电力服务</view>

</view>

<!-- 步骤指示器 -->

<view class="step-indicator">

<view class="step-list">

<view class="step-item {{currentStep >= 1 ? 'active' : ''}}">

<view class="step-number {{currentStep > 1 ? 'completed' : ''}}">

<text wx:if="{{currentStep > 1}}" class="iconfont icon-check"></text>

<text wx:else>1</text>

</view>

<view class="step-title">上传证件</view>

</view>

<view class="step-line {{currentStep > 1 ? 'completed' : ''}}"></view>

<view class="step-item {{currentStep >= 2 ? 'active' : ''}}">

<view class="step-number {{currentStep > 2 ? 'completed' : ''}}">

<text wx:if="{{currentStep > 2}}" class="iconfont icon-check"></text>

<text wx:else>2</text>

</view>

<view class="step-title">信息确认</view>

</view>

<view class="step-line {{currentStep > 2 ? 'completed' : ''}}"></view>

<view class="step-item {{currentStep >= 3 ? 'active' : ''}}">

<view class="step-number {{authStatus === 'success' ? 'completed' : ''}}">

<text wx:if="{{authStatus === 'success'}}" class="iconfont icon-check"></text>

<text wx:else>3</text>

</view>

<view class="step-title">等待审核</view>

</view>

</view>

</view>

<!-- 加载状态 -->

<loading wx:if="{{loading}}" text="加载中..." />

<!-- 步骤1: 上传证件 -->

<view wx:if="{{currentStep === 1}}" class="step-content">

<view class="form-container">

<view class="form-title">上传企业证件</view>

<!-- 营业执照 -->

<view class="upload-section">

<view class="upload-label">

<text class="required">\*</text>

营业执照

</view>

<view class="upload-tip">请上传清晰的营业执照照片</view>

<view wx:if="{{!businessLicense.uploaded}}" class="upload-area" bindtap="chooseImage" data-type="businessLicense">

<view class="upload-icon iconfont icon-camera"></view>

<view class="upload-text">点击上传营业执照</view>

<view class="upload-hint">支持 JPG、PNG 格式，大小于5MB</view>

</view>

<view wx:else class="upload-preview">

<image class="preview-image" src="{{businessLicense.url}}" mode="aspectFit" />

<view class="preview-mask" bindtap="previewImage" data-url="{{businessLicense.url}}">

<view class="preview-icon iconfont icon-eye"></view>

</view>

<view class="upload-status">

<text wx:if="{{businessLicense.status === 'uploading'}}" class="status-uploading">上传中...</text>

<text wx:elif="{{businessLicense.status === 'success'}}" class="status-success iconfont icon-check-circle">上传成功</text>

<text wx:elif="{{businessLicense.status === 'error'}}" class="status-error iconfont icon-close-circle">上传失败</text>

</view>

<view class="reupload-btn" bindtap="chooseImage" data-type="businessLicense">重新上传</view>

</view>

</view>

<!-- 法人身份证正面 -->

<view class="upload-section">

<view class="upload-label">

<text class="required">\*</text>

法人身份证正面

</view>

<view class="upload-tip">请上传身份证人像面</view>

<view wx:if="{{!idCardFront.uploaded}}" class="upload-area" bindtap="chooseImage" data-type="idCardFront">

<view class="upload-icon iconfont icon-id-card"></view>

<view class="upload-text">点击上传身份证正面</view>

<view class="upload-hint">支持 JPG、PNG 格式，大小5MB</view>

</view>

<view wx:else class="upload-preview">

<image class="preview-image" src="{{idCardFront.url}}" mode="aspectFit" />

<view class="preview-mask" bindtap="previewImage" data-url="{{idCardFront.url}}">

<view class="preview-icon iconfont icon-eye"></view>

</view>

<view class="upload-status">

<text wx:if="{{idCardFront.status === 'uploading'}}" class="status-uploading">上传中...</text>

<text wx:elif="{{idCardFront.status === 'success'}}" class="status-success iconfont icon-check-circle">上传成功</text>

<text wx:elif="{{idCardFront.status === 'error'}}" class="status-error iconfont icon-close-circle">上传失败</text>

</view>

<view class="reupload-btn" bindtap="chooseImage" data-type="idCardFront">重新上传</view>

</view>

</view>

<!-- 法人身份证背面 -->

<view class="upload-section">

<view class="upload-label">

<text class="required">\*</text>

法人身份证背面

</view>

<view class="upload-tip">请上传身份证国徽面</view>

<view wx:if="{{!idCardBack.uploaded}}" class="upload-area" bindtap="chooseImage" data-type="idCardBack">

<view class="upload-icon iconfont icon-id-card"></view>

<view class="upload-text">点击上传身份证背面</view>

<view class="upload-hint">支持 JPG、PNG 格式，大小5MB</view>

</view>

<view wx:else class="upload-preview">

<image class="preview-image" src="{{idCardBack.url}}" mode="aspectFit" />

<view class="preview-mask" bindtap="previewImage" data-url="{{idCardBack.url}}">

<view class="preview-icon iconfont icon-eye"></view>

</view>

<view class="upload-status">

<text wx:if="{{idCardBack.status === 'uploading'}}" class="status-uploading">上传中...</text>

<text wx:elif="{{idCardBack.status === 'success'}}" class="status-success iconfont icon-check-circle">上传成功</text>

<text wx:elif="{{idCardBack.status === 'error'}}" class="status-error iconfont icon-close-circle">上传失败</text>

</view>

<view class="reupload-btn" bindtap="chooseImage" data-type="idCardBack">重新上传</view>

</view>

</view>

</view>

<!-- 下一步按钮 -->

<view class="form-footer">

<button class="btn-primary {{canNext ? '' : 'disabled'}}" bindtap="nextStep" disabled="{{!canNext}}">

下一步

</button>

</view>

</view>

<!-- 步骤2: 信息确认 -->

<view wx:if="{{currentStep === 2}}" class="step-content">

<view class="form-container">

<view class="form-title">确认企业信息</view>

<!-- OCR识别结果 -->

<view wx:if="{{ocrData}}" class="ocr-result">

<view class="section-title">

<text class="iconfont icon-check-circle"></text>

识别结果

</view>

<view class="info-group">

<view class="info-item">

<view class="info-label">企业名称：</view>

<view class="info-value">{{ocrData.companyName || '未识别'}}</view>

</view>

<view class="info-item">

<view class="info-label">统一社会信用代码：</view>

<view class="info-value">{{ocrData.creditCode || '未识别'}}</view>

</view>

<view class="info-item">

<view class="info-label">法定代表人：</view>

<view class="info-value">{{ocrData.legalPerson || '未识别'}}</view>

</view>

<view class="info-item">

<view class="info-label">注册地址：</view>

<view class="info-value">{{ocrData.address || '未识别'}}</view>

</view>

</view>

<view class="confirm-tip">

<text class="iconfont icon-info-circle"></text>

请核对上述信息是否正确，如有错误请手动修改

</view>

</view>

<!-- 手动填写表单 -->

<view class="manual-form">

<view class="section-title">完善企业信息</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

企业名称

</view>

<input class="form-input"

placeholder="请输入企业全称"

value="{{formData.companyName}}"

bindinput="onInputChange"

data-field="companyName" />

</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

统一社会信用代码

</view>

<input class="form-input"

placeholder="请输入18位统一社会信用代码"

value="{{formData.creditCode}}"

bindinput="onInputChange"

data-field="creditCode" />

</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

法定代表人

</view>

<input class="form-input"

placeholder="请输入法定代表人姓名"

value="{{formData.legalPerson}}"

bindinput="onInputChange"

data-field="legalPerson" />

</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

注册地址

</view>

<textarea class="form-textarea"

placeholder="请输入详细注册地址"

value="{{formData.address}}"

bindinput="onInputChange"

data-field="address" />

</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

企业电话

</view>

<input class="form-input"

placeholder="请输入企业联系电话"

value="{{formData.companyPhone}}"

bindinput="onInputChange"

data-field="companyPhone" />

</view>

<view class="form-item">

<view class="form-label">经营范围</view>

<textarea class="form-textarea"

placeholder="请输入企业经营范围（可选）"

value="{{formData.businessScope}}"

bindinput="onInputChange"

data-field="businessScope" />

</view>

</view>

<!-- 用电信息 -->

<view class="power-info">

<view class="section-title">用电信息</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

用电容量 (kVA)

</view>

<input class="form-input"

type="number"

placeholder="请输入变压器容量"

value="{{formData.capacity}}"

bindinput="onInputChange"

data-field="capacity" />

</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

月平均用电量 (kWh)

</view>

<input class="form-input"

type="number"

placeholder="请输入月平均用电量"

value="{{formData.monthlyUsage}}"

bindinput="onInputChange"

data-field="monthlyUsage" />

</view>

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

目前电价 (元/kWh)

</view>

<input class="form-input"

type="digit"

placeholder="请输入目前电价"

value="{{formData.currentPrice}}"

bindinput="onInputChange"

data-field="currentPrice" />

</view>

<view class="form-item">

<view class="form-label">行业类型</view>

<picker bindchange="onPickerChange"

data-field="industryType"

value="{{industryIndex}}"

range="{{industryList}}"

range-key="name">

<view class="picker-input">

<text wx:if="{{formData.industryType}}">{{formData.industryType}}</text>

<text wx:else class="placeholder">请选择行业类型</text>

<text class="iconfont icon-arrow-right"></text>

</view>

</picker>

</view>

</view>

</view>

<!-- 操作按钮 -->

<view class="form-footer">

<button class="btn-secondary" bindtap="prevStep">上一步</button>

<button class="btn-primary {{canSubmit ? '' : 'disabled'}}" bindtap="submitAuth" disabled="{{!canSubmit}}">

提交认证

</button>

</view>

</view>

<!-- 步骤3: 审核状态 -->

<view wx:if="{{currentStep === 3}}" class="step-content">

<view class="status-container">

<!-- 提交成功状态 -->

<view wx:if="{{authStatus === 'pending'}}" class="status-content">

<view class="status-icon success">

<text class="iconfont icon-clock"></text>

</view>

<view class="status-title">认证信息已提交</view>

<view class="status-message">

您的企业认证信息已成功提交，我们将在1-3个工作日内完成审核，请耐心等待。

</view>

<view class="status-tips">

<view class="tip-item">

<text class="iconfont icon-check"></text>

认证通过后，您将获得更多服务权限

</view>

<view class="tip-item">

<text class="iconfont icon-check"></text>

如有疑问，请联系客服：400-xxx-xxxx

</view>

</view>

</view>

<!-- 认证成功状态 -->

<view wx:if="{{authStatus === 'success'}}" class="status-content">

<view class="status-icon success">

<text class="iconfont icon-check-circle"></text>

</view>

<view class="status-title">认证成功</view>

<view class="status-message">

恭喜您，企业认证已通过！现在您可以享受完整的电力服务。

</view>

<view class="certified-badge">

<text class="iconfont icon-verified"></text>

已认证企业

</view>

</view>

<!-- 认证失败状态 -->

<view wx:if="{{authStatus === 'failed'}}" class="status-content">

<view class="status-icon error">

<text class="iconfont icon-close-circle"></text>

</view>

<view class="status-title">认证失败</view>

<view class="status-message">

很抱歉，您的认证信息未通过审核。

</view>

<view class="reject-reason" wx:if="{{rejectReason}}">

<view class="reason-title">失败原因：</view>

<view class="reason-content">{{rejectReason}}</view>

</view>

</view>

</view>

<!-- 操作按钮 -->

<view class="form-footer">

<button wx:if="{{authStatus === 'success'}}" class="btn-primary" bindtap="goToHome">

开始使用

</button>

<button wx:elif="{{authStatus === 'failed'}}" class="btn-primary" bindtap="retryAuth">

重新认证

</button>

<button wx:else class="btn-secondary" bindtap="goToHome">

返回首页

</button>

</view>

</view>

</view>

<!-- 图片预览弹窗 -->

<view wx:if="{{showPreview}}" class="preview-modal" bindtap="hidePreview">

<view class="preview-content" catchtap="">

<view class="preview-header">

<view class="preview-title">图片预览</view>

<view class="preview-close" bindtap="hidePreview">

<text class="iconfont icon-close"></text>

</view>

</view>

<view class="preview-body">

<image class="preview-full-image" src="{{previewUrl}}" mode="aspectFit" />

</view>

</view>

</view>

现在继续开发下一部分 - 小程序身份认证页面样式文件 (verify.wxss)：小程序身份认证页面样式 - verify.wxss

/\* pages/auth/verify/verify.wxss \*/

.verify-container {

min-height: 100vh;

background: linear-gradient(180deg, #f8fafc 0%, #ffffff 100%);

padding-bottom: 120rpx;

}

/\* 页面头部 \*/

.verify-header {

padding: 60rpx 40rpx 40rpx;

text-align: center;

}

.header-title {

font-size: 48rpx;

font-weight: 600;

color: #1a202c;

margin-bottom: 16rpx;

}

.header-subtitle {

font-size: 28rpx;

color: #718096;

}

/\* 步骤指示器 \*/

.step-indicator {

margin: 0 40rpx 60rpx;

}

.step-list {

display: flex;

align-items: center;

justify-content: space-between;

position: relative;

}

.step-item {

display: flex;

flex-direction: column;

align-items: center;

flex: 1;

position: relative;

}

.step-number {

width: 60rpx;

height: 60rpx;

border-radius: 50%;

background: #e2e8f0;

color: #a0aec0;

display: flex;

align-items: center;

justify-content: center;

font-size: 28rpx;

font-weight: 600;

margin-bottom: 16rpx;

transition: all 0.3s ease;

}

.step-item.active .step-number {

background: var(--primary-color);

color: #ffffff;

}

.step-number.completed {

background: var(--success-color);

color: #ffffff;

}

.step-title {

font-size: 24rpx;

color: #a0aec0;

transition: all 0.3s ease;

}

.step-item.active .step-title {

color: var(--primary-color);

font-weight: 500;

}

.step-line {

position: absolute;

top: 30rpx;

left: 50%;

width: calc(100% - 60rpx);

height: 4rpx;

background: #e2e8f0;

z-index: -1;

transition: all 0.3s ease;

}

.step-line.completed {

background: var(--success-color);

}

/\* 步骤内容 \*/

.step-content {

margin: 0 40rpx;

}

.form-container {

background: #ffffff;

border-radius: 24rpx;

padding: 40rpx;

box-shadow: 0 4rpx 20rpx rgba(0, 0, 0, 0.08);

margin-bottom: 40rpx;

}

.form-title {

font-size: 36rpx;

font-weight: 600;

color: #1a202c;

margin-bottom: 40rpx;

text-align: center;

}

.section-title {

font-size: 32rpx;

font-weight: 600;

color: #2d3748;

margin-bottom: 32rpx;

display: flex;

align-items: center;

}

.section-title .iconfont {

margin-right: 12rpx;

color: var(--success-color);

}

/\* 上传区域 \*/

.upload-section {

margin-bottom: 60rpx;

}

.upload-label {

font-size: 28rpx;

color: #2d3748;

font-weight: 500;

margin-bottom: 12rpx;

display: flex;

align-items: center;

}

.required {

color: var(--danger-color);

margin-right: 8rpx;

}

.upload-tip {

font-size: 24rpx;

color: #718096;

margin-bottom: 24rpx;

}

.upload-area {

border: 4rpx dashed #cbd5e0;

border-radius: 16rpx;

padding: 80rpx 40rpx;

text-align: center;

background: #f7fafc;

transition: all 0.3s ease;

}

.upload-area:active {

border-color: var(--primary-color);

background: rgba(66, 153, 225, 0.05);

}

.upload-icon {

font-size: 80rpx;

color: #cbd5e0;

margin-bottom: 24rpx;

}

.upload-text {

font-size: 28rpx;

color: #4a5568;

font-weight: 500;

margin-bottom: 12rpx;

}

.upload-hint {

font-size: 24rpx;

color: #a0aec0;

}

.upload-preview {

position: relative;

border-radius: 16rpx;

overflow: hidden;

background: #f7fafc;

}

.preview-image {

width: 100%;

height: 400rpx;

background: #ffffff;

}

.preview-mask {

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.6);

display: flex;

align-items: center;

justify-content: center;

opacity: 0;

transition: all 0.3s ease;

}

.upload-preview:active .preview-mask {

opacity: 1;

}

.preview-icon {

font-size: 60rpx;

color: #ffffff;

}

.upload-status {

position: absolute;

top: 20rpx;

right: 20rpx;

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 24rpx;

font-weight: 500;

}

.status-uploading {

background: rgba(255, 193, 7, 0.9);

color: #ffffff;

}

.status-success {

background: rgba(56, 178, 172, 0.9);

color: #ffffff;

}

.status-error {

background: rgba(245, 101, 101, 0.9);

color: #ffffff;

}

.reupload-btn {

position: absolute;

bottom: 20rpx;

right: 20rpx;

background: rgba(0, 0, 0, 0.6);

color: #ffffff;

padding: 12rpx 24rpx;

border-radius: 20rpx;

font-size: 24rpx;

}

/\* OCR识别结果 \*/

.ocr-result {

margin-bottom: 40rpx;

}

.info-group {

background: #f7fafc;

border-radius: 16rpx;

padding: 32rpx;

margin-bottom: 24rpx;

}

.info-item {

display: flex;

align-items: flex-start;

margin-bottom: 24rpx;

}

.info-item:last-child {

margin-bottom: 0;

}

.info-label {

font-size: 28rpx;

color: #4a5568;

min-width: 180rpx;

flex-shrink: 0;

}

.info-value {

font-size: 28rpx;

color: #1a202c;

font-weight: 500;

flex: 1;

word-break: break-all;

}

.confirm-tip {

display: flex;

align-items: flex-start;

padding: 24rpx;

background: #fff3cd;

border-radius: 12rpx;

font-size: 24rpx;

color: #856404;

}

.confirm-tip .iconfont {

margin-right: 12rpx;

margin-top: 4rpx;

flex-shrink: 0;

}

/\* 表单项 \*/

.manual-form,

.power-info {

margin-bottom: 40rpx;

}

.form-item {

margin-bottom: 32rpx;

}

.form-label {

font-size: 28rpx;

color: #2d3748;

font-weight: 500;

margin-bottom: 16rpx;

display: flex;

align-items: center;

}

.form-input,

.form-textarea {

width: 100%;

height: 88rpx;

padding: 0 24rpx;

border: 2rpx solid #e2e8f0;

border-radius: 12rpx;

font-size: 28rpx;

color: #1a202c;

background: #ffffff;

box-sizing: border-box;

transition: all 0.3s ease;

}

.form-textarea {

height: 160rpx;

padding: 24rpx;

line-height: 1.5;

}

.form-input:focus,

.form-textarea:focus {

border-color: var(--primary-color);

box-shadow: 0 0 0 6rpx rgba(66, 153, 225, 0.1);

}

.form-input::placeholder,

.form-textarea::placeholder {

color: #a0aec0;

}

.picker-input {

height: 88rpx;

padding: 0 24rpx;

border: 2rpx solid #e2e8f0;

border-radius: 12rpx;

background: #ffffff;

display: flex;

align-items: center;

justify-content: space-between;

font-size: 28rpx;

color: #1a202c;

}

.picker-input .placeholder {

color: #a0aec0;

}

.picker-input .iconfont {

color: #a0aec0;

transform: rotate(90deg);

}

/\* 状态页面 \*/

.status-container {

padding: 80rpx 40rpx;

text-align: center;

}

.status-content {

background: #ffffff;

border-radius: 24rpx;

padding: 80rpx 40rpx;

box-shadow: 0 4rpx 20rpx rgba(0, 0, 0, 0.08);

}

.status-icon {

width: 120rpx;

height: 120rpx;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

margin: 0 auto 40rpx;

font-size: 60rpx;

}

.status-icon.success {

background: rgba(56, 178, 172, 0.1);

color: var(--success-color);

}

.status-icon.error {

background: rgba(245, 101, 101, 0.1);

color: var(--danger-color);

}

.status-title {

font-size: 40rpx;

font-weight: 600;

color: #1a202c;

margin-bottom: 24rpx;

}

.status-message {

font-size: 28rpx;

color: #718096;

line-height: 1.6;

margin-bottom: 40rpx;

}

.status-tips {

text-align: left;

margin-bottom: 40rpx;

}

.tip-item {

display: flex;

align-items: center;

font-size: 26rpx;

color: #4a5568;

margin-bottom: 16rpx;

}

.tip-item .iconfont {

color: var(--success-color);

margin-right: 16rpx;

font-size: 24rpx;

}

.certified-badge {

display: inline-flex;

align-items: center;

padding: 16rpx 32rpx;

background: rgba(56, 178, 172, 0.1);

border: 2rpx solid var(--success-color);

border-radius: 40rpx;

font-size: 26rpx;

color: var(--success-color);

font-weight: 500;

}

.certified-badge .iconfont {

margin-right: 12rpx;

font-size: 24rpx;

}

.reject-reason {

background: #fed7d7;

border-radius: 12rpx;

padding: 24rpx;

margin-bottom: 40rpx;

text-align: left;

}

.reason-title {

font-size: 26rpx;

color: #c53030;

font-weight: 500;

margin-bottom: 12rpx;

}

.reason-content {

font-size: 26rpx;

color: #e53e3e;

line-height: 1.5;

}

/\* 底部按钮 \*/

.form-footer {

position: fixed;

bottom: 0;

left: 0;

right: 0;

padding: 30rpx 40rpx;

background: #ffffff;

box-shadow: 0 -4rpx 20rpx rgba(0, 0, 0, 0.08);

display: flex;

gap: 20rpx;

}

.btn-primary,

.btn-secondary {

flex: 1;

height: 88rpx;

border-radius: 12rpx;

font-size: 28rpx;

font-weight: 500;

border: none;

display: flex;

align-items: center;

justify-content: center;

}

.btn-primary {

background: var(--primary-color);

color: #ffffff;

}

.btn-primary:disabled {

background: #cbd5e0;

color: #a0aec0;

}

.btn-secondary {

background: #ffffff;

color: var(--primary-color);

border: 2rpx solid var(--primary-color);

}

.btn-secondary:disabled {

background: #f7fafc;

color: #a0aec0;

border-color: #cbd5e0;

}

/\* 加载状态 \*/

.loading-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.5);

display: flex;

align-items: center;

justify-content: center;

z-index: 9999;

}

.loading-content {

background: #ffffff;

border-radius: 16rpx;

padding: 60rpx;

display: flex;

flex-direction: column;

align-items: center;

}

.loading-icon {

width: 80rpx;

height: 80rpx;

border: 4rpx solid #e2e8f0;

border-top: 4rpx solid var(--primary-color);

border-radius: 50%;

animation: spin 1s linear infinite;

margin-bottom: 32rpx;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

.loading-text {

font-size: 28rpx;

color: #4a5568;

text-align: center;

}

/\* 进度条 \*/

.progress-bar {

margin: 0 40rpx 40rpx;

background: #f1f5f9;

border-radius: 8rpx;

height: 8rpx;

overflow: hidden;

}

.progress-fill {

height: 100%;

background: linear-gradient(90deg, var(--primary-color) 0%, #38b2ac 100%);

border-radius: 8rpx;

transition: width 0.3s ease;

}

/\* 错误提示 \*/

.error-message {

background: #fed7d7;

border: 2rpx solid #feb2b2;

border-radius: 12rpx;

padding: 24rpx;

margin-bottom: 32rpx;

display: flex;

align-items: flex-start;

}

.error-icon {

color: #e53e3e;

margin-right: 16rpx;

margin-top: 4rpx;

flex-shrink: 0;

font-size: 32rpx;

}

.error-content {

flex: 1;

}

.error-title {

font-size: 26rpx;

color: #c53030;

font-weight: 500;

margin-bottom: 8rpx;

}

.error-text {

font-size: 24rpx;

color: #e53e3e;

line-height: 1.5;

}

/\* 成功提示 \*/

.success-message {

background: #c6f6d5;

border: 2rpx solid #9ae6b4;

border-radius: 12rpx;

padding: 24rpx;

margin-bottom: 32rpx;

display: flex;

align-items: flex-start;

}

.success-icon {

color: var(--success-color);

margin-right: 16rpx;

margin-top: 4rpx;

flex-shrink: 0;

font-size: 32rpx;

}

.success-content {

flex: 1;

}

.success-title {

font-size: 26rpx;

color: #276749;

font-weight: 500;

margin-bottom: 8rpx;

}

.success-text {

font-size: 24rpx;

color: #38a169;

line-height: 1.5;

}

/\* 空状态 \*/

.empty-state {

text-align: center;

padding: 80rpx 40rpx;

}

.empty-icon {

font-size: 120rpx;

color: #cbd5e0;

margin-bottom: 32rpx;

}

.empty-title {

font-size: 32rpx;

color: #4a5568;

font-weight: 500;

margin-bottom: 16rpx;

}

.empty-text {

font-size: 26rpx;

color: #a0aec0;

line-height: 1.5;

}

/\* 动画效果 \*/

.fade-in {

animation: fadeIn 0.3s ease-in-out;

}

@keyframes fadeIn {

from {

opacity: 0;

transform: translateY(20rpx);

}

to {

opacity: 1;

transform: translateY(0);

}

}

.slide-up {

animation: slideUp 0.3s ease-out;

}

@keyframes slideUp {

from {

opacity: 0;

transform: translateY(100rpx);

}

to {

opacity: 1;

transform: translateY(0);

}

}

/\* 响应式适配 \*/

@media (max-width: 375px) {

.form-container {

padding: 32rpx;

}

.header-title {

font-size: 44rpx;

}

.section-title {

font-size: 30rpx;

}

}

/\* 暗色主题适配 \*/

@media (prefers-color-scheme: dark) {

.verify-container {

background: linear-gradient(180deg, #1a202c 0%, #2d3748 100%);

}

.form-container {

background: #2d3748;

box-shadow: 0 4rpx 20rpx rgba(0, 0, 0, 0.3);

}

.header-title,

.form-title,

.section-title,

.info-value {

color: #f7fafc;

}

.header-subtitle,

.info-label {

color: #cbd5e0;

}

.form-input,

.form-textarea,

.picker-input {

background: #4a5568;

border-color: #718096;

color: #f7fafc;

}

.form-input::placeholder,

.form-textarea::placeholder {

color: #a0aec0;

}

}

**语言：CSS (微信小程序样式)**

现在继续下一个模块的开发：

小程序身份认证页面配置 - verify.json

{

"navigationBarTitleText": "身份认证",

"navigationBarTextStyle": "black",

"navigationBarBackgroundColor": "#ffffff",

"backgroundColor": "#f8fafc",

"enablePullDownRefresh": false,

"disableScroll": false,

"usingComponents": {

"custom-loading": "/components/loading/loading",

"custom-toast": "/components/toast/toast",

"image-cropper": "/components/image-cropper/image-cropper"

}

}

**语言：JSON (微信小程序配置)**

接下来开发产品展示模块小程序页面：

产品列表页面逻辑 - pages/products/index/index.js

// pages/products/index/index.js

const app = getApp()

const api = require('../../../utils/api')

const utils = require('../../../utils/util')

Page({

data: {

// 产品列表

products: [],

loading: true,

hasMore: true,

page: 1,

pageSize: 10,

// 筛选条件

filters: {

category: '',

priceRange: '',

suitable: false

},

// 分类选项

categories: [

{ id: '', name: '全部' },

{ id: 'standard', name: '标准套餐' },

{ id: 'premium', name: '优选套餐' },

{ id: 'custom', name: '定制套餐' }

],

// 价格区间选项

priceRanges: [

{ id: '', name: '全部价格' },

{ id: '0-0.4', name: '0.4元以下' },

{ id: '0.4-0.5', name: '0.4-0.5元' },

{ id: '0.5-0.6', name: '0.5-0.6元' },

{ id: '0.6-999', name: '0.6元以上' }

],

// 搜索

searchKeyword: '',

searchFocused: false,

// 用户信息

userInfo: null,

powerInfo: null,

// UI状态

showFilter: false,

refreshing: false,

error: null

},

onLoad(options) {

this.getUserInfo()

this.loadProducts()

// 处理分享参数

if (options.category) {

this.setData({

'filters.category': options.category

})

}

},

onShow() {

// 刷新用户信息和产品推荐

this.getUserInfo()

},

onPullDownRefresh() {

this.refreshProducts()

},

onReachBottom() {

if (this.data.hasMore && !this.data.loading) {

this.loadMoreProducts()

}

},

onShareAppMessage() {

return {

title: '电力优选套餐，为您节省用电成本',

path: '/pages/products/index/index',

imageUrl: '/images/share-products.jpg'

}

},

// 获取用户信息

async getUserInfo() {

try {

const userInfo = app.globalData.userInfo || await app.getUserInfo()

const powerInfo = await api.getUserPowerInfo()

this.setData({

userInfo,

powerInfo

})

} catch (error) {

console.log('获取用户信息失败:', error)

}

},

// 加载产品列表

async loadProducts(refresh = false) {

if (this.data.loading && !refresh) return

try {

this.setData({

loading: true,

error: null

})

const params = {

page: refresh ? 1 : this.data.page,

pageSize: this.data.pageSize,

...this.data.filters,

keyword: this.data.searchKeyword

}

// 如果用户已认证，添加用电信息用于推荐

if (this.data.powerInfo) {

params.capacity = this.data.powerInfo.capacity

params.monthlyUsage = this.data.powerInfo.monthlyUsage

params.industryType = this.data.powerInfo.industryType

}

const result = await api.getProducts(params)

const products = refresh ? result.items : [...this.data.products, ...result.items]

this.setData({

products,

hasMore: result.hasMore,

page: refresh ? 2 : this.data.page + 1,

loading: false

})

if (refresh) {

wx.stopPullDownRefresh()

this.setData({ refreshing: false })

}

} catch (error) {

console.error('加载产品失败:', error)

this.setData({

loading: false,

error: '加载失败，请重试'

})

if (refresh) {

wx.stopPullDownRefresh()

this.setData({ refreshing: false })

}

wx.showToast({

title: '加载失败',

icon: 'none'

})

}

},

// 刷新产品列表

refreshProducts() {

this.setData({

refreshing: true,

page: 1,

hasMore: true

})

this.loadProducts(true)

},

// 加载更多产品

loadMoreProducts() {

this.loadProducts()

},

// 重试加载

retryLoad() {

this.setData({

error: null,

page: 1,

hasMore: true,

products: []

})

this.loadProducts()

},

// 搜索功能

onSearchInput(e) {

this.setData({

searchKeyword: e.detail.value

})

},

onSearchFocus() {

this.setData({ searchFocused: true })

},

onSearchBlur() {

this.setData({ searchFocused: false })

},

onSearchConfirm(e) {

const keyword = e.detail.value.trim()

this.setData({ searchKeyword: keyword })

this.refreshProducts()

},

clearSearch() {

this.setData({ searchKeyword: '' })

this.refreshProducts()

},

// 筛选功能

toggleFilter() {

this.setData({ showFilter: !this.data.showFilter })

},

onCategoryChange(e) {

const category = e.currentTarget.dataset.category

this.setData({

'filters.category': category

})

},

onPriceRangeChange(e) {

const priceRange = e.currentTarget.dataset.range

this.setData({

'filters.priceRange': priceRange

})

},

onSuitableToggle() {

this.setData({

'filters.suitable': !this.data.filters.suitable

})

},

resetFilters() {

this.setData({

filters: {

category: '',

priceRange: '',

suitable: false

},

searchKeyword: ''

})

},

applyFilters() {

this.setData({ showFilter: false })

this.refreshProducts()

},

// 产品操作

onProductTap(e) {

const productId = e.currentTarget.dataset.productId

const product = this.data.products.find(p => p.id === productId)

// 记录点击行为

this.trackProductView(product)

wx.navigateTo({

url: `/pages/products/detail/detail?id=${productId}`

})

},

onCalculatorTap(e) {

const productId = e.currentTarget.dataset.productId

wx.navigateTo({

url: `/pages/products/calculator/calculator?productId=${productId}`

})

},

onQuickOrderTap(e) {

const productId = e.currentTarget.dataset.productId

// 检查用户认证状态

if (!this.data.userInfo?.isAuthenticated) {

wx.showModal({

title: '需要认证',

content: '下单前需要完成企业认证，是否前往认证？',

success: (res) => {

if (res.confirm) {

wx.navigateTo({

url: '/pages/auth/verify/verify'

})

}

}

})

return

}

wx.navigateTo({

url: `/pages/orders/create/create?productId=${productId}`

})

},

// 数据统计

trackProductView(product) {

api.trackEvent('product\_view', {

productId: product.id,

productName: product.name,

category: product.category,

price: product.price

}).catch(console.error)

},

// 跳转到详情页

goToDetail(e) {

const productId = e.currentTarget.dataset.productId

wx.navigateTo({

url: `/pages/products/detail/detail?id=${productId}`

})

},

// 跳转到计算器页面

goToCalculator() {

if (!this.data.powerInfo) {

wx.showModal({

title: '提示',

content: '请先完成用电信息填写，以获得准确的收益计算',

success: (res) => {

if (res.confirm) {

wx.navigateTo({

url: '/pages/user/power-info/power-info'

})

}

}

})

return

}

wx.navigateTo({

url: '/pages/products/calculator/calculator'

})

},

// 跳转到认证页面

goToAuth() {

wx.navigateTo({

url: '/pages/auth/verify/verify'

})

},

// 格式化价格显示

formatPrice(price) {

if (typeof price === 'number') {

return price.toFixed(3)

}

return price

},

// 格式化节省金额

formatSavings(amount) {

if (amount >= 10000) {

return `${(amount / 10000).toFixed(1)}万`

}

return amount.toLocaleString()

},

// 获取产品适用标签

getProductTags(product) {

const tags = []

if (product.isRecommended) {

tags.push({ text: '推荐', type: 'primary' })

}

if (product.isHot) {

tags.push({ text: '热门', type: 'danger' })

}

if (product.isSuitable && this.data.powerInfo) {

tags.push({ text: '适合', type: 'success' })

}

if (product.hasDiscount) {

tags.push({ text: '优惠', type: 'warning' })

}

return tags

}

})

**语言：JavaScript (微信小程序)**

现在继续开发产品列表页面的模板文件：

产品列表页面模板 - pages/products/index/index.wxml

<!--pages/products/index/index.wxml-->

<view class="products-container">

<!-- 搜索栏 -->

<view class="search-bar">

<view class="search-input-wrapper {{searchFocused ? 'focused' : ''}}">

<icon class="iconfont icon-search search-icon"></icon>

<input class="search-input"

placeholder="搜索产品套餐..."

value="{{searchKeyword}}"

bindinput="onSearchInput"

bindfocus="onSearchFocus"

bindblur="onSearchBlur"

bindconfirm="onSearchConfirm" />

<view class="clear-btn {{searchKeyword ? 'show' : ''}}" bindtap="clearSearch">

<icon class="iconfont icon-close"></icon>

</view>

</view>

<view class="filter-btn {{showFilter ? 'active' : ''}}" bindtap="toggleFilter">

<icon class="iconfont icon-filter"></icon>

<text>筛选</text>

</view>

</view>

<!-- 筛选面板 -->

<view class="filter-panel {{showFilter ? 'show' : ''}}">

<view class="filter-content">

<!-- 分类筛选 -->

<view class="filter-section">

<view class="filter-title">产品分类</view>

<view class="filter-options">

<view class="filter-option {{filters.category === item.id ? 'active' : ''}}"

wx:for="{{categories}}" wx:key="id"

data-category="{{item.id}}"

bindtap="onCategoryChange">

{{item.name}}

</view>

</view>

</view>

<!-- 价格筛选 -->

<view class="filter-section">

<view class="filter-title">价格区间</view>

<view class="filter-options">

<view class="filter-option {{filters.priceRange === item.id ? 'active' : ''}}"

wx:for="{{priceRanges}}" wx:key="id"

data-range="{{item.id}}"

bindtap="onPriceRangeChange">

{{item.name}}

</view>

</view>

</view>

<!-- 个性化推荐 -->

<view class="filter-section" wx:if="{{powerInfo}}">

<view class="filter-title">个性化推荐</view>

<view class="filter-toggle">

<view class="toggle-item" bindtap="onSuitableToggle">

<text>只看适合我的</text>

<icon class="iconfont {{filters.suitable ? 'icon-toggle-on' : 'icon-toggle-off'}} toggle-icon"></icon>

</view>

</view>

</view>

</view>

<!-- 筛选操作按钮 -->

<view class="filter-actions">

<view class="btn-secondary filter-btn" bindtap="resetFilters">重置</view>

<view class="btn-primary filter-btn" bindtap="applyFilters">确定</view>

</view>

</view>

<!-- 用户状态提示 -->

<view class="user-status" wx:if="{{!userInfo.isAuthenticated}}">

<view class="status-card">

<icon class="iconfont icon-warning status-icon"></icon>

<view class="status-content">

<text class="status-title">完成认证，享受专属优惠</text>

<text class="status-desc">认证企业用户可享受更优惠的电价和专属服务</text>

</view>

<view class="btn-primary status-btn" bindtap="goToAuth">立即认证</view>

</view>

</view>

<!-- 快捷计算器入口 -->

<view class="calculator-entry" wx:if="{{powerInfo}}">

<view class="entry-card" bindtap="goToCalculator">

<view class="entry-icon">

<icon class="iconfont icon-calculator"></icon>

</view>

<view class="entry-content">

<text class="entry-title">节电收益计算器</text>

<text class="entry-desc">快速计算不同套餐的节省金额</text>

</view>

<icon class="iconfont icon-arrow-right entry-arrow"></icon>

</view>

</view>

<!-- 产品列表 -->

<view class="products-list">

<!-- 加载状态 -->

<view class="loading-placeholder" wx:if="{{loading && products.length === 0}}">

<view class="product-skeleton" wx:for="{{[1,2,3]}}" wx:key="\*this">

<view class="skeleton-image"></view>

<view class="skeleton-content">

<view class="skeleton-line skeleton-title"></view>

<view class="skeleton-line skeleton-desc"></view>

<view class="skeleton-line skeleton-price"></view>

</view>

</view>

</view>

<!-- 产品卡片 -->

<view class="product-card"

wx:for="{{products}}" wx:key="id"

data-product-id="{{item.id}}"

bindtap="onProductTap">

<!-- 产品标签 -->

<view class="product-tags" wx:if="{{item.tags && item.tags.length > 0}}">

<text class="product-tag tag-{{tag.type}}"

wx:for="{{item.tags}}" wx:for-item="tag" wx:key="text">

{{tag.text}}

</text>

</view>

<!-- 产品头部 -->

<view class="product-header">

<view class="product-title">{{item.name}}</view>

<view class="product-type">{{item.categoryName}}</view>

</view>

<!-- 价格信息 -->

<view class="product-price">

<view class="price-main">

<text class="price-value">{{item.priceRange || item.price}}</text>

<text class="price-unit">元/kWh</text>

</view>

<view class="price-desc" wx:if="{{item.priceDesc}}">{{item.priceDesc}}</view>

</view>

<!-- 节省预估 -->

<view class="product-savings" wx:if="{{item.estimatedSavings}}">

<text class="savings-label">预计年节省：</text>

<text class="savings-amount">¥{{item.estimatedSavings}}</text>

</view>

<!-- 产品特点 -->

<view class="product-features" wx:if="{{item.features && item.features.length > 0}}">

<text class="feature-item" wx:for="{{item.features}}" wx:key="\*this">

{{item}}

</text>

</view>

<!-- 适用说明 -->

<view class="product-suitable" wx:if="{{item.suitableDesc}}">

<icon class="iconfont icon-info suitable-icon"></icon>

<text class="suitable-text">{{item.suitableDesc}}</text>

</view>

<!-- 操作按钮 -->

<view class="product-actions">

<view class="btn-secondary action-btn"

data-product-id="{{item.id}}"

catchtap="onCalculatorTap">

<icon class="iconfont icon-calculator"></icon>

<text>计算收益</text>

</view>

<view class="btn-primary action-btn"

data-product-id="{{item.id}}"

catchtap="onQuickOrderTap">

<icon class="iconfont icon-order"></icon>

<text>立即咨询</text>

</view>

</view>

</view>

</view>

<!-- 空状态 -->

<view class="empty-state" wx:if="{{!loading && products.length === 0 && !error}}">

<icon class="iconfont icon-empty empty-icon"></icon>

<text class="empty-title">暂无产品</text>

<text class="empty-desc">{{searchKeyword ? '没有找到相关产品，试试其他关键词' : '暂时没有可用的产品套餐'}}</text>

<view class="btn-primary empty-btn" bindtap="refreshProducts">刷新重试</view>

</view>

<!-- 错误状态 -->

<view class="error-state" wx:if="{{error && products.length === 0}}">

<icon class="iconfont icon-error error-icon"></icon>

<text class="error-title">加载失败</text>

<text class="error-desc">{{error}}</text>

<view class="btn-primary error-btn" bindtap="retryLoad">重新加载</view>

</view>

<!-- 加载更多 -->

<view class="load-more" wx:if="{{products.length > 0}}">

<view class="load-more-loading" wx:if="{{loading}}">

<icon class="loading-icon"></icon>

<text>加载中...</text>

</view>

<view class="load-more-end" wx:elif="{{!hasMore}}">

<text>已显示全部产品</text>

</view>

</view>

</view>

<!-- 筛选遮罩 -->

<view class="filter-mask {{showFilter ? 'show' : ''}}" bindtap="toggleFilter"></view>

**产品列表页面样式文件**

产品列表页面样式 - pages/products/index/index.wxss

/\* 产品列表页面样式 \*/

.container {

background-color: #f5f5f5;

min-height: 100vh;

}

/\* 搜索区域 \*/

.search-section {

padding: 20rpx;

background-color: #fff;

margin-bottom: 20rpx;

}

.search-bar {

position: relative;

background-color: #f8f8f8;

border-radius: 50rpx;

padding: 20rpx 40rpx;

display: flex;

align-items: center;

border: 2rpx solid #e0e0e0;

}

.search-icon {

width: 32rpx;

height: 32rpx;

margin-right: 20rpx;

opacity: 0.6;

}

.search-input {

flex: 1;

font-size: 28rpx;

color: #333;

}

.search-input.placeholder {

color: #999;

}

/\* 筛选区域 \*/

.filter-section {

padding: 20rpx;

background-color: #fff;

margin-bottom: 20rpx;

border-bottom: 1rpx solid #f0f0f0;

}

.filter-tabs {

display: flex;

align-items: center;

}

.filter-item {

flex: 1;

text-align: center;

padding: 20rpx 0;

position: relative;

font-size: 28rpx;

color: #666;

transition: all 0.3s ease;

}

.filter-item.active {

color: var(--primary-color);

font-weight: 600;

}

.filter-item.active::after {

content: '';

position: absolute;

bottom: 0;

left: 50%;

transform: translateX(-50%);

width: 40rpx;

height: 4rpx;

background-color: var(--primary-color);

border-radius: 2rpx;

}

/\* 产品列表区域 \*/

.product-list {

padding: 0 20rpx;

}

.product-card {

background-color: #fff;

border-radius: 20rpx;

margin-bottom: 20rpx;

padding: 30rpx;

box-shadow: 0 4rpx 20rpx rgba(0, 0, 0, 0.06);

transition: all 0.3s ease;

position: relative;

overflow: hidden;

}

.product-card::before {

content: '';

position: absolute;

top: 0;

left: 0;

width: 6rpx;

height: 100%;

background: linear-gradient(180deg, var(--primary-color) 0%, var(--secondary-color) 100%);

}

.product-card:active {

transform: translateY(-4rpx);

box-shadow: 0 8rpx 30rpx rgba(0, 0, 0, 0.12);

}

/\* 产品头部 \*/

.product-header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 20rpx;

}

.product-title {

flex: 1;

font-size: 32rpx;

font-weight: 600;

color: #333;

line-height: 1.4;

margin-right: 20rpx;

}

.product-tag {

background: linear-gradient(135deg, #ff6b6b 0%, #ff8e53 100%);

color: #fff;

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 20rpx;

font-weight: 500;

white-space: nowrap;

}

.product-tag.hot {

background: linear-gradient(135deg, #ff6b6b 0%, #ff8e53 100%);

}

.product-tag.new {

background: linear-gradient(135deg, #4ecdc4 0%, #44a08d 100%);

}

.product-tag.recommend {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

/\* 产品信息 \*/

.product-info {

margin-bottom: 25rpx;

}

.price-section {

display: flex;

align-items: baseline;

margin-bottom: 15rpx;

}

.price-range {

font-size: 36rpx;

font-weight: 700;

color: var(--primary-color);

margin-right: 10rpx;

}

.price-unit {

font-size: 24rpx;

color: #666;

}

.savings {

font-size: 26rpx;

color: #ff6b6b;

margin-top: 8rpx;

display: flex;

align-items: center;

}

.savings-icon {

width: 28rpx;

height: 28rpx;

margin-right: 8rpx;

}

/\* 产品特点 \*/

.product-features {

display: flex;

flex-wrap: wrap;

gap: 15rpx;

margin-bottom: 25rpx;

}

.feature-item {

background-color: #f8f9ff;

color: var(--primary-color);

padding: 12rpx 20rpx;

border-radius: 30rpx;

font-size: 24rpx;

border: 1rpx solid rgba(64, 158, 255, 0.2);

}

/\* 适用信息 \*/

.suitable-info {

font-size: 24rpx;

color: #666;

line-height: 1.5;

margin-bottom: 25rpx;

background-color: #fafafa;

padding: 20rpx;

border-radius: 12rpx;

border-left: 4rpx solid var(--primary-color);

}

/\* 产品底部 \*/

.product-footer {

display: flex;

justify-content: space-between;

align-items: center;

}

.view-detail {

background: linear-gradient(135deg, var(--primary-color) 0%, var(--secondary-color) 100%);

color: #fff;

padding: 20rpx 40rpx;

border-radius: 50rpx;

font-size: 28rpx;

font-weight: 500;

box-shadow: 0 8rpx 20rpx rgba(64, 158, 255, 0.3);

transition: all 0.3s ease;

}

.view-detail:active {

transform: translateY(2rpx);

box-shadow: 0 4rpx 15rpx rgba(64, 158, 255, 0.4);

}

.customer-info {

font-size: 22rpx;

color: #999;

display: flex;

align-items: center;

}

.customer-count {

color: var(--primary-color);

font-weight: 600;

margin-left: 8rpx;

}

/\* 加载状态 \*/

.loading-section {

padding: 60rpx 20rpx;

text-align: center;

color: #999;

font-size: 28rpx;

}

.loading-icon {

width: 60rpx;

height: 60rpx;

margin-bottom: 20rpx;

animation: rotate 1s linear infinite;

}

@keyframes rotate {

from { transform: rotate(0deg); }

to { transform: rotate(360deg); }

}

/\* 空状态 \*/

.empty-section {

padding: 100rpx 20rpx;

text-align: center;

}

.empty-icon {

width: 120rpx;

height: 120rpx;

margin-bottom: 30rpx;

opacity: 0.6;

}

.empty-text {

font-size: 28rpx;

color: #999;

line-height: 1.5;

}

/\* 底部加载更多 \*/

.load-more {

padding: 40rpx 20rpx;

text-align: center;

font-size: 26rpx;

color: #999;

}

.load-more.loading {

color: var(--primary-color);

}

/\* 返回顶部按钮 \*/

.back-to-top {

position: fixed;

bottom: 120rpx;

right: 30rpx;

width: 80rpx;

height: 80rpx;

background: linear-gradient(135deg, var(--primary-color) 0%, var(--secondary-color) 100%);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 8rpx 25rpx rgba(64, 158, 255, 0.4);

z-index: 999;

transition: all 0.3s ease;

}

.back-to-top:active {

transform: scale(0.95);

}

.back-to-top-icon {

width: 36rpx;

height: 36rpx;

opacity: 0.9;

}

/\* 响应式适配 \*/

@media screen and (max-width: 320px) {

.product-title {

font-size: 30rpx;

}

.price-range {

font-size: 32rpx;

}

.product-card {

padding: 25rpx;

}

}

/\* 暗色模式适配 \*/

@media (prefers-color-scheme: dark) {

.container {

background-color: #1a1a1a;

}

.search-section,

.filter-section,

.product-card {

background-color: #2d2d2d;

}

.search-bar {

background-color: #3a3a3a;

border-color: #4a4a4a;

}

.product-title {

color: #fff;

}

.suitable-info {

background-color: #3a3a3a;

}

.feature-item {

background-color: rgba(64, 158, 255, 0.1);

}

}

**产品列表页面配置文件**

产品列表页面配置 - pages/products/index/index.json

{

"navigationBarTitleText": "产品选择",

"navigationBarBackgroundColor": "#409EFF",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f5f5",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"disableScroll": false,

"usingComponents": {

"loading": "/components/loading/loading",

"empty": "/components/empty/empty"

}

}

**产品详情页面逻辑**

产品详情页面逻辑 - pages/products/detail/detail.js

// 产品详情页面逻辑

const api = require('../../../utils/api');

const util = require('../../../utils/util');

Page({

data: {

productId: null,

product: null,

loading: true,

showCalculator: false,

userInfo: null,

// 节电计算器数据

calculatorData: {

currentCapacity: '',

monthlyUsage: '',

currentPrice: '',

results: null

},

// 轮播图当前索引

currentSwiper: 0,

// 选项卡索引

currentTab: 0,

tabList: ['产品介绍', '服务内容', '客户案例', '常见问题'],

// 客户案例数据

customerCases: [],

// 常见问题数据

faqs: [],

// 是否显示更多介绍

showMoreIntro: false

},

onLoad(options) {

console.log('产品详情页面加载:', options);

if (options.id) {

this.setData({

productId: options.id

});

this.loadProductDetail();

} else {

wx.showToast({

title: '产品信息错误',

icon: 'error'

});

setTimeout(() => {

wx.navigateBack();

}, 1500);

}

// 获取用户信息

this.getUserInfo();

},

onShow() {

// 页面显示时刷新用户信息

this.getUserInfo();

},

onShareAppMessage() {

const { product } = this.data;

return {

title: `推荐产品：${product?.name || '优质购电产品'}`,

path: `/pages/products/detail/detail?id=${this.data.productId}`,

imageUrl: product?.images?.[0] || '/images/default-product.png'

};

},

onShareTimeline() {

const { product } = this.data;

return {

title: `${product?.name || '优质购电产品'} - 立即查看详情`,

imageUrl: product?.images?.[0] || '/images/default-product.png'

};

},

// 获取用户信息

async getUserInfo() {

try {

const userInfo = wx.getStorageSync('userInfo');

if (userInfo) {

this.setData({ userInfo });

// 如果有用电信息，自动填充计算器

if (userInfo.powerConsumption) {

this.setData({

'calculatorData.currentCapacity': userInfo.powerConsumption.capacity || '',

'calculatorData.monthlyUsage': userInfo.powerConsumption.monthlyUsage || '',

'calculatorData.currentPrice': userInfo.powerConsumption.currentPrice || ''

});

}

}

} catch (error) {

console.error('获取用户信息失败:', error);

}

},

// 加载产品详情

async loadProductDetail() {

try {

wx.showLoading({ title: '加载中...' });

const response = await api.request({

url: `/products/${this.data.productId}`,

method: 'GET'

});

if (response.code === 200) {

this.setData({

product: response.data,

customerCases: response.data.customerCases || [],

faqs: response.data.faqs || []

});

// 更新页面标题

wx.setNavigationBarTitle({

title: response.data.name

});

// 记录浏览记录

this.recordBrowseHistory();

} else {

throw new Error(response.message || '加载产品详情失败');

}

} catch (error) {

console.error('加载产品详情失败:', error);

wx.showToast({

title: error.message || '加载失败',

icon: 'error'

});

setTimeout(() => {

wx.navigateBack();

}, 1500);

} finally {

wx.hideLoading();

this.setData({ loading: false });

}

},

// 记录浏览记录

async recordBrowseHistory() {

try {

await api.request({

url: '/user/browse-history',

method: 'POST',

data: {

productId: this.data.productId,

productName: this.data.product.name,

browseTime: new Date().toISOString()

}

});

} catch (error) {

console.error('记录浏览历史失败:', error);

}

},

// 轮播图变化

onSwiperChange(e) {

this.setData({

currentSwiper: e.detail.current

});

},

// 选项卡切换

onTabChange(e) {

const index = e.currentTarget.dataset.index;

this.setData({

currentTab: index

});

},

// 预览图片

previewImage(e) {

const current = e.currentTarget.dataset.url;

const urls = this.data.product.images || [];

wx.previewImage({

current,

urls

});

},

// 显示/隐藏计算器

toggleCalculator() {

this.setData({

showCalculator: !this.data.showCalculator

});

},

// 计算器输入处理

onCalculatorInput(e) {

const field = e.currentTarget.dataset.field;

const value = e.detail.value;

this.setData({

[`calculatorData.${field}`]: value

});

// 自动计算

this.autoCalculate();

},

// 自动计算节电效益

autoCalculate() {

const { currentCapacity, monthlyUsage, currentPrice } = this.data.calculatorData;

const { product } = this.data;

if (currentCapacity && monthlyUsage && currentPrice && product) {

const capacity = parseFloat(currentCapacity);

const usage = parseFloat(monthlyUsage);

const price = parseFloat(currentPrice);

if (capacity > 0 && usage > 0 && price > 0) {

// 计算节省金额（假设产品可以节省10-20%的电费）

const savingRate = (product.maxSavingRate + product.minSavingRate) / 2 / 100;

const currentCost = usage \* price;

const monthlySaving = currentCost \* savingRate;

const yearlySaving = monthlySaving \* 12;

// 计算投资回报周期（假设有初始投资）

const investmentCost = capacity \* (product.investmentPerKw || 100);

const paybackPeriod = investmentCost / monthlySaving;

this.setData({

'calculatorData.results': {

currentCost: currentCost.toFixed(2),

monthlySaving: monthlySaving.toFixed(2),

yearlySaving: yearlySaving.toFixed(2),

savingRate: (savingRate \* 100).toFixed(1),

paybackPeriod: paybackPeriod.toFixed(1)

}

});

}

}

},

// 立即购买

buyNow() {

const { userInfo, product } = this.data;

// 检查登录状态

if (!userInfo) {

wx.showModal({

title: '提示',

content: '请先登录后再购买产品',

confirmText: '去登录',

success: (res) => {

if (res.confirm) {

wx.navigateTo({

url: '/pages/auth/login/login'

});

}

}

});

return;

}

// 检查认证状态

if (userInfo.authStatus !== 'verified') {

wx.showModal({

title: '提示',

content: '请先完成身份认证后再购买产品',

confirmText: '去认证',

success: (res) => {

if (res.confirm) {

wx.navigateTo({

url: '/pages/auth/verify/verify'

});

}

}

});

return;

}

// 跳转到订单创建页面

const orderData = {

productId: product.id,

productName: product.name,

productType: product.type,

estimatedAmount: this.data.calculatorData.results?.yearlySaving || 0

};

wx.navigateTo({

url: `/pages/orders/create/create?data=${encodeURIComponent(JSON.stringify(orderData))}`

});

},

// 咨询客服

contactService() {

const { product } = this.data;

wx.navigateTo({

url: `/pages/service/chat/chat?productId=${product.id}&productName=${encodeURIComponent(product.name)}`

});

},

// 收藏产品

async toggleFavorite() {

const { product, userInfo } = this.data;

if (!userInfo) {

wx.showToast({

title: '请先登录',

icon: 'error'

});

return;

}

try {

wx.showLoading({ title: '处理中...' });

const isFavorited = product.isFavorited;

const url = isFavorited ? '/user/favorites/remove' : '/user/favorites/add';

const response = await api.request({

url,

method: 'POST',

data: {

productId: product.id

}

});

if (response.code === 200) {

this.setData({

'product.isFavorited': !isFavorited

});

wx.showToast({

title: isFavorited ? '已取消收藏' : '收藏成功',

icon: 'success'

});

} else {

throw new Error(response.message);

}

} catch (error) {

console.error('收藏操作失败:', error);

wx.showToast({

title: error.message || '操作失败',

icon: 'error'

});

} finally {

wx.hideLoading();

}

},

// 切换介绍展开状态

toggleIntroExpand() {

this.setData({

showMoreIntro: !this.data.showMoreIntro

});

},

// 查看客户案例详情

viewCaseDetail(e) {

const index = e.currentTarget.dataset.index;

const caseItem = this.data.customerCases[index];

wx.navigateTo({

url: `/pages/products/case/case?data=${encodeURIComponent(JSON.stringify(caseItem))}`

});

},

// 展开/收起FAQ

toggleFaq(e) {

const index = e.currentTarget.dataset.index;

const faqs = this.data.faqs;

faqs[index].expanded = !faqs[index].expanded;

this.setData({ faqs });

},

// 页面滚动处理

onPageScroll(e) {

// 可以在这里添加滚动相关的逻辑，比如显示/隐藏返回顶部按钮

const scrollTop = e.scrollTop;

// 超过一屏高度显示返回顶部按钮

if (scrollTop > 800) {

this.setData({ showBackToTop: true });

} else {

this.setData({ showBackToTop: false });

}

},

// 返回顶部

backToTop() {

wx.pageScrollTo({

scrollTop: 0,

duration: 300

});

},

// 页面下拉刷新

onPullDownRefresh() {

this.loadProductDetail().finally(() => {

wx.stopPullDownRefresh();

});

}

});

**产品详情页面模板**

产品详情页面模板 - pages/products/detail/detail.wxml

<!-- 产品详情页面模板 -->

<view class="container">

<!-- 加载状态 -->

<loading wx:if="{{loading}}" />

<!-- 产品详情内容 -->

<view wx:else class="detail-content">

<!-- 产品图片轮播 -->

<view class="image-section">

<swiper

class="product-swiper"

indicator-dots="{{product.images.length > 1}}"

indicator-color="rgba(255,255,255,0.5)"

indicator-active-color="#fff"

autoplay="{{product.images.length > 1}}"

interval="5000"

duration="500"

bindchange="onSwiperChange"

>

<swiper-item wx:for="{{product.images}}" wx:key="index">

<image

class="product-image"

src="{{item}}"

mode="aspectFill"

bindtap="previewImage"

data-url="{{item}}"

/>

</swiper-item>

</swiper>

<!-- 图片指示器 -->

<view wx:if="{{product.images.length > 1}}" class="custom-indicator">

{{currentSwiper + 1}} / {{product.images.length}}

</view>

<!-- 收藏按钮 -->

<view class="favorite-btn" bindtap="toggleFavorite">

<image

class="favorite-icon"

src="{{product.isFavorited ? '/images/icons/heart-filled.png' : '/images/icons/heart.png'}}"

/>

</view>

</view>

<!-- 产品基本信息 -->

<view class="product-info-section">

<view class="product-header">

<view class="product-title">{{product.name}}</view>

<view wx:if="{{product.tags.length}}" class="product-tags">

<text wx:for="{{product.tags}}" wx:key="index" class="tag tag-{{item.type}}">

{{item.text}}

</text>

</view>

</view>

<view class="price-info">

<view class="price-main">

<text class="price-range">{{product.priceRange}}</text>

<text class="price-unit">元/kWh</text>

</view>

<view class="savings-info">

<image class="savings-icon" src="/images/icons/savings.png" />

<text>预估年节省 ¥{{product.estimatedSavings}}</text>

</view>

</view>

<view class="product-summary">

<text class="summary-text {{showMoreIntro ? 'expanded' : ''}}">

{{product.description}}

</text>

<text class="expand-btn" bindtap="toggleIntroExpand">

{{showMoreIntro ? '收起' : '展开'}}

</text>

</view>

</view>

<!-- 节电计算器 -->

<view class="calculator-section">

<view class="section-header" bindtap="toggleCalculator">

<text class="section-title">节电效益计算器</text>

<image

class="expand-icon {{showCalculator ? 'expanded' : ''}}"

src="/images/icons/expand.png"

/>

</view>

<view wx:if="{{showCalculator}}" class="calculator-content">

<view class="input-group">

<label class="input-label">用电容量 (kW)</label>

<input

class="input-field"

type="digit"

placeholder="请输入用电容量"

value="{{calculatorData.currentCapacity}}"

bindinput="onCalculatorInput"

data-field="currentCapacity"

/>

</view>

<view class="input-group">

<label class="input-label">月用电量 (kWh)</label>

<input

class="input-field"

type="digit"

placeholder="请输入月用电量"

value="{{calculatorData.monthlyUsage}}"

bindinput="onCalculatorInput"

data-field="monthlyUsage"

/>

</view>

<view class="input-group">

<label class="input-label">当前电价 (元/kWh)</label>

<input

class="input-field"

type="digit"

placeholder="请输入当前电价"

value="{{calculatorData.currentPrice}}"

bindinput="onCalculatorInput"

data-field="currentPrice"

/>

</view>

<!-- 计算结果 -->

<view wx:if="{{calculatorData.isCalculated}}" class="calculator-result">

<view class="result-title">计算结果</view>

<view class="result-item">

<text class="result-label">当前年电费：</text>

<text class="result-value">¥{{calculatorData.currentYearCost}}</text>

</view>

<view class="result-item">

<text class="result-label">选择此套餐年电费：</text>

<text class="result-value highlight">¥{{calculatorData.newYearCost}}</text>

</view>

<view class="result-item savings">

<text class="result-label">预计年节省：</text>

<text class="result-value savings-value">¥{{calculatorData.yearSavings}}</text>

</view>

<view class="result-item">

<text class="result-label">投资回报周期：</text>

<text class="result-value">{{calculatorData.paybackPeriod}}个月</text>

</view>

</view>

<view class="calculator-actions">

<button

class="calculate-btn"

bindtap="calculateSavings"

disabled="{{!calculatorData.isValid}}"

>

计算节省金额

</button>

</view>

</view>

</view>

<!-- 产品详细信息 -->

<view class="product-details-section">

<view class="detail-tabs">

<view

wx:for="{{detailTabs}}"

wx:key="index"

class="tab-item {{currentTab === index ? 'active' : ''}}"

bindtap="switchTab"

data-index="{{index}}"

>

{{item.title}}

</view>

</view>

<!-- 产品介绍 -->

<view wx:if="{{currentTab === 0}}" class="tab-content">

<view class="feature-list">

<view wx:for="{{product.features}}" wx:key="index" class="feature-item">

<image class="feature-icon" src="{{item.icon}}" />

<view class="feature-content">

<text class="feature-title">{{item.title}}</text>

<text class="feature-desc">{{item.description}}</text>

</view>

</view>

</view>

<view class="service-scope">

<text class="scope-title">服务范围</text>

<text class="scope-content">{{product.serviceScope}}</text>

</view>

</view>

<!-- 价格说明 -->

<view wx:if="{{currentTab === 1}}" class="tab-content">

<view class="price-structure">

<text class="structure-title">电价构成</text>

<view class="price-chart">

<canvas

canvas-id="priceChart"

class="chart-canvas"

bindtouchstart="chartTouchStart"

bindtouchmove="chartTouchMove"

bindtouchend="chartTouchEnd"

></canvas>

</view>

</view>

<view class="price-tiers">

<text class="tiers-title">阶梯价格</text>

<view class="tier-list">

<view wx:for="{{product.priceTiers}}" wx:key="index" class="tier-item">

<view class="tier-range">{{item.range}}</view>

<view class="tier-price">{{item.price}}元/kWh</view>

</view>

</view>

</view>

<view class="billing-info">

<text class="billing-title">结算方式</text>

<text class="billing-content">{{product.billingMethod}}</text>

</view>

</view>

<!-- 服务内容 -->

<view wx:if="{{currentTab === 2}}" class="tab-content">

<view class="service-list">

<view wx:for="{{product.services}}" wx:key="index" class="service-item">

<image class="service-icon" src="{{item.icon}}" />

<view class="service-info">

<text class="service-name">{{item.name}}</text>

<text class="service-desc">{{item.description}}</text>

</view>

<image class="check-icon" src="/images/icons/check.png" />

</view>

</view>

<view class="support-info">

<text class="support-title">技术支持</text>

<view class="support-content">

<text class="support-text">{{product.technicalSupport}}</text>

<button class="contact-support-btn" bindtap="contactSupport">

联系技术支持

</button>

</view>

</view>

</view>

<!-- 客户案例 -->

<view wx:if="{{currentTab === 3}}" class="tab-content">

<view class="case-list">

<view wx:for="{{product.cases}}" wx:key="index" class="case-item">

<view class="case-header">

<text class="case-company">{{item.companyName}}</text>

<text class="case-industry">{{item.industry}}</text>

</view>

<view class="case-stats">

<view class="stat-item">

<text class="stat-label">用电容量</text>

<text class="stat-value">{{item.capacity}}kW</text>

</view>

<view class="stat-item">

<text class="stat-label">年节省</text>

<text class="stat-value highlight">¥{{item.savings}}</text>

</view>

<view class="stat-item">

<text class="stat-label">节省比例</text>

<text class="stat-value">{{item.savingsRate}}%</text>

</view>

</view>

<view class="case-comment">

<text class="comment-text">"{{item.comment}}"</text>

<text class="comment-author">—— {{item.contactPerson}}</text>

</view>

</view>

</view>

</view>

</view>

<!-- 风险提示 -->

<view class="risk-notice-section">

<view class="notice-header">

<image class="warning-icon" src="/images/icons/warning.png" />

<text class="notice-title">风险提示</text>

</view>

<view class="notice-content">

<text class="notice-text">{{product.riskNotice}}</text>

</view>

</view>

<!-- 推荐产品 -->

<view class="recommended-section">

<text class="section-title">推荐产品</text>

<scroll-view class="recommended-list" scroll-x="true" show-scrollbar="false">

<view

wx:for="{{recommendedProducts}}"

wx:key="id"

class="recommended-item"

bindtap="goToProduct"

data-id="{{item.id}}"

>

<image class="recommended-image" src="{{item.image}}" mode="aspectFill" />

<view class="recommended-info">

<text class="recommended-name">{{item.name}}</text>

<text class="recommended-price">{{item.priceRange}}元/kWh</text>

</view>

</view>

</scroll-view>

</view>

</view>

<!-- 底部操作栏 -->

<view class="bottom-actions">

<button class="action-btn secondary" bindtap="addToFavorites">

<image class="btn-icon" src="/images/icons/heart-outline.png" />

收藏

</button>

<button class="action-btn secondary" bindtap="shareProduct">

<image class="btn-icon" src="/images/icons/share.png" />

分享

</button>

<button class="action-btn primary" bindtap="consultNow">

立即咨询

</button>

<button class="action-btn primary filled" bindtap="orderNow">

立即下单

</button>

</view>

<!-- 客服联系弹窗 -->

<view wx:if="{{showContactModal}}" class="modal-overlay" bindtap="hideContactModal">

<view class="contact-modal" catchtap="">

<view class="modal-header">

<text class="modal-title">联系客服</text>

<image class="close-icon" src="/images/icons/close.png" bindtap="hideContactModal" />

</view>

<view class="modal-content">

<view class="contact-methods">

<button class="contact-method" bindtap="callPhone">

<image class="method-icon" src="/images/icons/phone.png" />

<text class="method-text">电话咨询</text>

<text class="method-info">{{customerService.phone}}</text>

</button>

<button class="contact-method" open-type="contact">

<image class="method-icon" src="/images/icons/wechat.png" />

<text class="method-text">微信客服</text>

<text class="method-info">在线咨询</text>

</button>

</view>

<view class="service-time">

<text class="time-text">服务时间：{{customerService.workTime}}</text>

</view>

</view>

</view>

</view>

</view>

现在让我继续开发下一个模块：**产品详情页面样式文件 - pages/products/detail/detail.wxss**

产品详情页面样式文件 - detail.wxss

/\* pages/products/detail/detail.wxss \*/

@import '/styles/common.wxss';

.container {

background-color: var(--bg-color-light);

min-height: 100vh;

padding-bottom: 120rpx; /\* 为底部操作栏留出空间 \*/

}

/\* 加载状态 \*/

.loading {

display: flex;

justify-content: center;

align-items: center;

height: 60vh;

}

/\* 产品图片区域 \*/

.image-section {

position: relative;

background-color: #fff;

}

.product-swiper {

height: 600rpx;

}

.product-image {

width: 100%;

height: 100%;

}

.custom-indicator {

position: absolute;

bottom: 20rpx;

right: 20rpx;

background-color: rgba(0, 0, 0, 0.6);

color: #fff;

padding: 10rpx 20rpx;

border-radius: 20rpx;

font-size: 24rpx;

}

.favorite-btn {

position: absolute;

top: 20rpx;

right: 20rpx;

width: 80rpx;

height: 80rpx;

background-color: rgba(255, 255, 255, 0.9);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.favorite-icon {

width: 40rpx;

height: 40rpx;

}

/\* 产品基本信息 \*/

.product-info-section {

background-color: #fff;

padding: 30rpx;

margin-bottom: 20rpx;

}

.product-header {

margin-bottom: 20rpx;

}

.product-title {

font-size: 36rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 15rpx;

line-height: 1.4;

}

.product-tags {

display: flex;

flex-wrap: wrap;

gap: 10rpx;

}

.tag {

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 22rpx;

line-height: 1;

}

.tag-recommend {

background-color: var(--primary-color);

color: #fff;

}

.tag-hot {

background-color: #ff4757;

color: #fff;

}

.tag-new {

background-color: #26de81;

color: #fff;

}

.tag-discount {

background-color: #ffa726;

color: #fff;

}

/\* 价格信息 \*/

.price-info {

margin-bottom: 25rpx;

}

.price-main {

display: flex;

align-items: baseline;

margin-bottom: 15rpx;

}

.price-range {

font-size: 42rpx;

font-weight: 700;

color: var(--primary-color);

margin-right: 10rpx;

}

.price-unit {

font-size: 28rpx;

color: var(--text-color-secondary);

}

.savings-info {

display: flex;

align-items: center;

background-color: #f8f9fa;

padding: 15rpx 20rpx;

border-radius: 12rpx;

}

.savings-icon {

width: 32rpx;

height: 32rpx;

margin-right: 10rpx;

}

.savings-info text {

font-size: 26rpx;

color: #26de81;

font-weight: 600;

}

/\* 产品简介 \*/

.product-summary {

position: relative;

}

.summary-text {

font-size: 28rpx;

color: var(--text-color-secondary);

line-height: 1.6;

display: -webkit-box;

-webkit-line-clamp: 3;

-webkit-box-orient: vertical;

overflow: hidden;

}

.summary-text.expanded {

display: block;

-webkit-line-clamp: none;

}

.expand-btn {

color: var(--primary-color);

font-size: 26rpx;

margin-left: 10rpx;

cursor: pointer;

}

/\* 计算器区域 \*/

.calculator-section {

background-color: #fff;

margin-bottom: 20rpx;

}

.section-header {

padding: 30rpx;

display: flex;

align-items: center;

justify-content: space-between;

border-bottom: 1rpx solid #f0f0f0;

}

.section-title {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

}

.expand-icon {

width: 32rpx;

height: 32rpx;

transition: transform 0.3s ease;

}

.expand-icon.expanded {

transform: rotate(180deg);

}

.calculator-content {

padding: 30rpx;

}

.input-group {

margin-bottom: 30rpx;

}

.input-label {

display: block;

font-size: 28rpx;

color: var(--text-color-primary);

margin-bottom: 15rpx;

font-weight: 500;

}

.input-field {

width: 100%;

height: 80rpx;

border: 2rpx solid #e0e0e0;

border-radius: 12rpx;

padding: 0 20rpx;

font-size: 28rpx;

box-sizing: border-box;

}

.input-field:focus {

border-color: var(--primary-color);

}

/\* 计算结果 \*/

.calculator-result {

background: linear-gradient(135deg, var(--primary-color), #667eea);

border-radius: 16rpx;

padding: 30rpx;

margin: 30rpx 0;

color: #fff;

}

.result-title {

font-size: 30rpx;

font-weight: 600;

margin-bottom: 25rpx;

text-align: center;

}

.result-item {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 15rpx;

font-size: 26rpx;

}

.result-item:last-child {

margin-bottom: 0;

}

.result-label {

opacity: 0.9;

}

.result-value {

font-weight: 600;

}

.result-value.highlight {

color: #26de81;

font-size: 28rpx;

}

.result-item.savings {

padding-top: 15rpx;

border-top: 1rpx solid rgba(255, 255, 255, 0.2);

margin-top: 15rpx;

}

.savings-value {

color: #26de81;

font-size: 32rpx !important;

font-weight: 700;

}

.calculator-actions {

margin-top: 30rpx;

}

.calculate-btn {

width: 100%;

height: 88rpx;

background: linear-gradient(135deg, var(--primary-color), #667eea);

color: #fff;

border: none;

border-radius: 44rpx;

font-size: 30rpx;

font-weight: 600;

}

.calculate-btn[disabled] {

background: #e0e0e0;

color: #999;

}

/\* 产品详情选项卡 \*/

.product-details-section {

background-color: #fff;

margin-bottom: 20rpx;

}

.detail-tabs {

display: flex;

border-bottom: 1rpx solid #f0f0f0;

}

.tab-item {

flex: 1;

text-align: center;

padding: 30rpx 0;

font-size: 28rpx;

color: var(--text-color-secondary);

position: relative;

}

.tab-item.active {

color: var(--primary-color);

font-weight: 600;

}

.tab-item.active::after {

content: '';

position: absolute;

bottom: 0;

left: 50%;

transform: translateX(-50%);

width: 60rpx;

height: 4rpx;

background-color: var(--primary-color);

border-radius: 2rpx;

}

.tab-content {

padding: 30rpx;

}

/\* 产品特性列表 \*/

.feature-list {

margin-bottom: 40rpx;

}

.feature-item {

display: flex;

align-items: flex-start;

margin-bottom: 30rpx;

}

.feature-item:last-child {

margin-bottom: 0;

}

.feature-icon {

width: 40rpx;

height: 40rpx;

margin-right: 20rpx;

margin-top: 5rpx;

}

.feature-content {

flex: 1;

}

.feature-title {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 8rpx;

}

.feature-desc {

font-size: 26rpx;

color: var(--text-color-secondary);

line-height: 1.5;

}

/\* 服务范围 \*/

.service-scope {

background-color: #f8f9fa;

padding: 25rpx;

border-radius: 12rpx;

}

.scope-title {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 15rpx;

}

.scope-content {

font-size: 26rpx;

color: var(--text-color-secondary);

line-height: 1.6;

}

/\* 价格结构 \*/

.price-structure {

margin-bottom: 40rpx;

}

.structure-title {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 20rpx;

}

.price-chart {

height: 400rpx;

background-color: #f8f9fa;

border-radius: 12rpx;

overflow: hidden;

}

.chart-canvas {

width: 100%;

height: 100%;

}

/\* 价格阶梯 \*/

.price-tiers {

margin-bottom: 40rpx;

}

.tiers-title {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 20rpx;

}

.tier-list {

background-color: #f8f9fa;

border-radius: 12rpx;

overflow: hidden;

}

.tier-item {

display: flex;

justify-content: space-between;

align-items: center;

padding: 25rpx 30rpx;

border-bottom: 1rpx solid #e0e0e0;

}

.tier-item:last-child {

border-bottom: none;

}

.tier-range {

font-size: 26rpx;

color: var(--text-color-secondary);

}

.tier-price {

font-size: 28rpx;

font-weight: 600;

color: var(--primary-color);

}

/\* 结算信息 \*/

.billing-info {

background-color: #f8f9fa;

padding: 25rpx;

border-radius: 12rpx;

}

.billing-title {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 15rpx;

}

.billing-content {

font-size: 26rpx;

color: var(--text-color-secondary);

line-height: 1.6;

}

/\* 服务列表 \*/

.service-list {

margin-bottom: 40rpx;

}

.service-item {

display: flex;

align-items: center;

padding: 25rpx 0;

border-bottom: 1rpx solid #f0f0f0;

}

.service-item:last-child {

border-bottom: none;

}

.service-icon {

width: 40rpx;

height: 40rpx;

margin-right: 20rpx;

}

.service-info {

flex: 1;

}

.service-name {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 8rpx;

}

.service-desc {

font-size: 24rpx;

color: var(--text-color-secondary);

line-height: 1.4;

}

.check-icon {

width: 32rpx;

height: 32rpx;

}

/\* 技术支持 \*/

.support-info {

background-color: #f8f9fa;

padding: 25rpx;

border-radius: 12rpx;

}

.support-title {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 15rpx;

}

.support-content {

display: flex;

justify-content: space-between;

align-items: flex-end;

}

.support-text {

flex: 1;

font-size: 26rpx;

color: var(--text-color-secondary);

line-height: 1.6;

margin-right: 20rpx;

}

.contact-support-btn {

padding: 15rpx 25rpx;

background-color: var(--primary-color);

color: #fff;

border: none;

border-radius: 8rpx;

font-size: 24rpx;

}

/\* 客户案例 \*/

.case-list {

display: flex;

flex-direction: column;

gap: 30rpx;

}

.case-item {

background-color: #f8f9fa;

padding: 30rpx;

border-radius: 16rpx;

border-left: 6rpx solid var(--primary-color);

}

.case-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 20rpx;

}

.case-company {

font-size: 30rpx;

font-weight: 600;

color: var(--text-color-primary);

}

.case-industry {

font-size: 24rpx;

color: var(--text-color-secondary);

background-color: #e0e0e0;

padding: 8rpx 16rpx;

border-radius: 20rpx;

}

.case-stats {

display: flex;

justify-content: space-between;

margin-bottom: 25rpx;

}

.stat-item {

text-align: center;

}

.stat-label {

display: block;

font-size: 24rpx;

color: var(--text-color-secondary);

margin-bottom: 8rpx;

}

.stat-value {

display: block;

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

}

.stat-value.highlight {

color: var(--primary-color);

font-size: 32rpx;

}

.case-comment {

background-color: #fff;

padding: 20rpx;

border-radius: 12rpx;

position: relative;

}

.comment-text {

display: block;

font-size: 26rpx;

color: var(--text-color-secondary);

line-height: 1.6;

margin-bottom: 15rpx;

font-style: italic;

}

.comment-author {

font-size: 24rpx;

color: var(--text-color-light);

text-align: right;

}

/\* 风险提示 \*/

.risk-notice-section {

background-color: #fff;

padding: 30rpx;

margin-bottom: 20rpx;

}

.notice-header {

display: flex;

align-items: center;

margin-bottom: 20rpx;

}

.warning-icon {

width: 32rpx;

height: 32rpx;

margin-right: 12rpx;

}

.notice-title {

font-size: 28rpx;

font-weight: 600;

color: #ff4757;

}

.notice-content {

padding: 20rpx;

background-color: #ffeaa3;

border-radius: 12rpx;

border-left: 6rpx solid #ff4757;

}

.notice-text {

font-size: 26rpx;

color: #d63031;

line-height: 1.6;

}

/\* 推荐产品 \*/

.recommended-section {

background-color: #fff;

padding: 30rpx;

margin-bottom: 20rpx;

}

.recommended-section .section-title {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 25rpx;

}

.recommended-list {

white-space: nowrap;

}

.recommended-item {

display: inline-block;

width: 200rpx;

margin-right: 20rpx;

vertical-align: top;

white-space: normal;

}

.recommended-item:last-child {

margin-right: 0;

}

.recommended-image {

width: 100%;

height: 150rpx;

border-radius: 12rpx;

margin-bottom: 15rpx;

}

.recommended-info {

display: flex;

flex-direction: column;

}

.recommended-name {

font-size: 24rpx;

color: var(--text-color-primary);

margin-bottom: 8rpx;

line-height: 1.3;

display: -webkit-box;

-webkit-line-clamp: 2;

-webkit-box-orient: vertical;

overflow: hidden;

}

.recommended-price {

font-size: 22rpx;

color: var(--primary-color);

font-weight: 600;

}

/\* 底部操作栏 \*/

.bottom-actions {

position: fixed;

bottom: 0;

left: 0;

right: 0;

background-color: #fff;

border-top: 1rpx solid #f0f0f0;

padding: 20rpx 30rpx;

padding-bottom: calc(20rpx + env(safe-area-inset-bottom));

display: flex;

align-items: center;

gap: 20rpx;

z-index: 1000;

box-shadow: 0 -2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.action-left {

display: flex;

align-items: center;

gap: 30rpx;

}

.contact-item {

display: flex;

flex-direction: column;

align-items: center;

gap: 8rpx;

}

.contact-icon {

width: 48rpx;

height: 48rpx;

}

.contact-text {

font-size: 20rpx;

color: var(--text-color-secondary);

}

.action-right {

flex: 1;

display: flex;

gap: 20rpx;

}

.action-btn {

flex: 1;

height: 80rpx;

border-radius: 40rpx;

display: flex;

align-items: center;

justify-content: center;

font-size: 28rpx;

font-weight: 600;

border: none;

}

.consult-btn {

background-color: #f8f9fa;

color: var(--text-color-primary);

border: 2rpx solid #e0e0e0;

}

.order-btn {

background: linear-gradient(135deg, var(--primary-color), #667eea);

color: #fff;

}

.action-btn[disabled] {

background-color: #e0e0e0;

color: #999;

border-color: #e0e0e0;

}

/\* 响应式适配 \*/

@media (max-width: 375px) {

.product-title {

font-size: 32rpx;

}

.price-range {

font-size: 38rpx;

}

.section-title {

font-size: 28rpx;

}

.bottom-actions {

padding: 15rpx 20rpx;

gap: 15rpx;

}

.action-btn {

height: 72rpx;

font-size: 26rpx;

}

}

/\* 深色模式适配 \*/

@media (prefers-color-scheme: dark) {

.container {

background-color: #121212;

}

.image-section,

.product-info-section,

.calculator-section,

.product-details-section,

.risk-notice-section,

.recommended-section {

background-color: #1e1e1e;

}

.input-field {

background-color: #2a2a2a;

border-color: #404040;

color: #fff;

}

.bottom-actions {

background-color: #1e1e1e;

border-top-color: #404040;

}

.consult-btn {

background-color: #2a2a2a;

border-color: #404040;

color: #fff;

}

}

/\* 动画效果 \*/

.product-image {

transition: transform 0.3s ease;

}

.favorite-btn {

transition: all 0.3s ease;

}

.favorite-btn:active {

transform: scale(0.95);

}

.tab-item {

transition: all 0.3s ease;

}

.action-btn {

transition: all 0.3s ease;

}

.action-btn:active {

transform: scale(0.98);

}

/\* 加载骨架屏 \*/

.skeleton {

background: linear-gradient(90deg, #f0f0f0 25%, #e0e0e0 50%, #f0f0f0 75%);

background-size: 200% 100%;

animation: loading 1.5s infinite;

}

@keyframes loading {

0% {

background-position: 200% 0;

}

100% {

background-position: -200% 0;

}

}

.skeleton-image {

width: 100%;

height: 600rpx;

border-radius: 0;

}

.skeleton-title {

width: 80%;

height: 36rpx;

border-radius: 4rpx;

margin-bottom: 15rpx;

}

.skeleton-price {

width: 60%;

height: 42rpx;

border-radius: 4rpx;

margin-bottom: 20rpx;

}

.skeleton-text {

width: 100%;

height: 28rpx;

border-radius: 4rpx;

margin-bottom: 10rpx;

}

.skeleton-text:last-child {

width: 70%;

}

/\* 空状态 \*/

.empty-state {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 100rpx 30rpx;

}

.empty-icon {

width: 120rpx;

height: 120rpx;

margin-bottom: 30rpx;

opacity: 0.3;

}

.empty-text {

font-size: 28rpx;

color: var(--text-color-secondary);

text-align: center;

line-height: 1.5;

}

/\* 错误状态 \*/

.error-state {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 100rpx 30rpx;

}

.error-icon {

width: 120rpx;

height: 120rpx;

margin-bottom: 30rpx;

}

.error-text {

font-size: 28rpx;

color: var(--text-color-secondary);

text-align: center;

line-height: 1.5;

margin-bottom: 30rpx;

}

.retry-btn {

padding: 20rpx 40rpx;

background-color: var(--primary-color);

color: #fff;

border: none;

border-radius: 8rpx;

font-size: 26rpx;

}

/\* 浮动按钮 \*/

.floating-actions {

position: fixed;

right: 30rpx;

bottom: 140rpx;

display: flex;

flex-direction: column;

gap: 20rpx;

z-index: 999;

}

.floating-btn {

width: 100rpx;

height: 100rpx;

border-radius: 50%;

background-color: #fff;

box-shadow: 0 4rpx 20rpx rgba(0, 0, 0, 0.15);

display: flex;

align-items: center;

justify-content: center;

border: none;

}

.floating-icon {

width: 48rpx;

height: 48rpx;

}

/\* 分享浮层 \*/

.share-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0, 0, 0, 0.5);

display: flex;

align-items: flex-end;

z-index: 2000;

}

.share-panel {

width: 100%;

background-color: #fff;

border-radius: 24rpx 24rpx 0 0;

padding: 40rpx 30rpx;

padding-bottom: calc(40rpx + env(safe-area-inset-bottom));

}

.share-title {

font-size: 32rpx;

font-weight: 600;

text-align: center;

margin-bottom: 40rpx;

color: var(--text-color-primary);

}

.share-options {

display: flex;

justify-content: space-around;

margin-bottom: 40rpx;

}

.share-option {

display: flex;

flex-direction: column;

align-items: center;

gap: 15rpx;

}

.share-option-icon {

width: 80rpx;

height: 80rpx;

border-radius: 50%;

background-color: #f8f9fa;

display: flex;

align-items: center;

justify-content: center;

}

.share-option-icon image {

width: 48rpx;

height: 48rpx;

}

.share-option-text {

font-size: 24rpx;

color: var(--text-color-secondary);

}

.share-cancel {

width: 100%;

height: 80rpx;

background-color: #f8f9fa;

border: none;

border-radius: 12rpx;

font-size: 28rpx;

color: var(--text-color-primary);

}

/\* 滚动指示器 \*/

.scroll-indicator {

position: fixed;

top: 50%;

right: 20rpx;

transform: translateY(-50%);

display: flex;

flex-direction: column;

gap: 8rpx;

z-index: 998;

}

.indicator-dot {

width: 12rpx;

height: 12rpx;

border-radius: 50%;

background-color: rgba(0, 0, 0, 0.2);

transition: all 0.3s ease;

}

.indicator-dot.active {

background-color: var(--primary-color);

transform: scale(1.2);

}

/\* 图片预览 \*/

.image-preview-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0, 0, 0, 0.9);

z-index: 3000;

display: flex;

align-items: center;

justify-content: center;

}

.preview-image {

max-width: 90%;

max-height: 90%;

}

.preview-close {

position: absolute;

top: 60rpx;

right: 30rpx;

width: 60rpx;

height: 60rpx;

background-color: rgba(255, 255, 255, 0.2);

border-radius: 50%;

color: #fff;

border: none;

font-size: 36rpx;

}

/\* 工具提示 \*/

.tooltip {

position: relative;

display: inline-block;

}

.tooltip-content {

position: absolute;

bottom: 125%;

left: 50%;

transform: translateX(-50%);

background-color: rgba(0, 0, 0, 0.8);

color: #fff;

padding: 10rpx 15rpx;

border-radius: 8rpx;

font-size: 22rpx;

white-space: nowrap;

opacity: 0;

visibility: hidden;

transition: all 0.3s ease;

z-index: 1001;

}

.tooltip-content::after {

content: '';

position: absolute;

top: 100%;

left: 50%;

transform: translateX(-50%);

border: 8rpx solid transparent;

border-top-color: rgba(0, 0, 0, 0.8);

}

.tooltip:hover .tooltip-content,

.tooltip.active .tooltip-content {

opacity: 1;

visibility: visible;

}

现在继续开发下一个模块：**产品详情页面配置文件**

产品详情页面配置 - pages/products/detail/detail.json

{

"navigationBarTitleText": "产品详情",

"navigationBarBackgroundColor": "#ffffff",

"navigationBarTextStyle": "black",

"backgroundColor": "#f5f6fa",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 100,

"disableScroll": false,

"usingComponents": {

"van-loading": "@vant/weapp/loading/index",

"van-empty": "@vant/weapp/empty/index",

"van-button": "@vant/weapp/button/index",

"van-toast": "@vant/weapp/toast/index",

"van-dialog": "@vant/weapp/dialog/index",

"van-action-sheet": "@vant/weapp/action-sheet/index",

"van-overlay": "@vant/weapp/overlay/index",

"van-transition": "@vant/weapp/transition/index"

},

"style": "v2",

"componentFramework": "glass-easel"

}

接下来开发**订单管理模块小程序页面**，首先是订单列表页面：

订单列表页面逻辑 - pages/orders/index/index.js

// pages/orders/index/index.js

const api = require('../../../utils/api');

const auth = require('../../../utils/auth');

const { formatDate, formatMoney } = require('../../../utils/format');

Page({

data: {

// 页面状态

loading: true,

refreshing: false,

loadingMore: false,

hasMore: true,

isEmpty: false,

// 筛选状态

activeTab: 'all',

tabList: [

{ key: 'all', name: '全部', count: 0 },

{ key: 'pending', name: '待确认', count: 0 },

{ key: 'confirmed', name: '已确认', count: 0 },

{ key: 'contract', name: '待签约', count: 0 },

{ key: 'active', name: '服务中', count: 0 },

{ key: 'completed', name: '已完成', count: 0 },

{ key: 'cancelled', name: '已取消', count: 0 }

],

// 订单数据

orderList: [],

page: 1,

pageSize: 10,

total: 0,

// 搜索

searchKeyword: '',

showSearch: false,

// 筛选

showFilter: false,

filterData: {

dateRange: '',

amountRange: '',

productType: ''

},

// 操作

showActionSheet: false,

actionSheetActions: [],

selectedOrder: null

},

onLoad(options) {

console.log('订单列表页面加载', options);

// 从参数获取状态筛选

if (options.status) {

this.setData({ activeTab: options.status });

}

this.initPage();

},

onShow() {

// 每次显示时刷新数据

this.refreshOrderList();

},

onPullDownRefresh() {

this.refreshOrderList();

},

onReachBottom() {

this.loadMoreOrders();

},

onShareAppMessage() {

return {

title: '我的订单',

path: '/pages/orders/index/index'

};

},

// 页面初始化

async initPage() {

try {

await this.loadOrderStats();

await this.loadOrderList();

} catch (error) {

console.error('页面初始化失败:', error);

wx.showToast({

title: '加载失败，请重试',

icon: 'none'

});

}

},

// 加载订单统计

async loadOrderStats() {

try {

const response = await api.request({

url: '/orders/stats',

method: 'GET'

});

if (response.success) {

const stats = response.data;

const tabList = this.data.tabList.map(tab => ({

...tab,

count: stats[tab.key] || 0

}));

this.setData({ tabList });

}

} catch (error) {

console.error('加载订单统计失败:', error);

}

},

// 加载订单列表

async loadOrderList(reset = false) {

if (this.data.loading && !reset) return;

const page = reset ? 1 : this.data.page;

this.setData({

loading: reset,

refreshing: reset

});

try {

const params = {

page,

page\_size: this.data.pageSize,

status: this.data.activeTab === 'all' ? '' : this.data.activeTab,

keyword: this.data.searchKeyword,

...this.data.filterData

};

const response = await api.request({

url: '/orders',

method: 'GET',

data: params

});

if (response.success) {

const { items, total, has\_more } = response.data;

// 处理订单数据

const processedOrders = items.map(order => this.processOrderData(order));

this.setData({

orderList: reset ? processedOrders : this.data.orderList.concat(processedOrders),

total,

hasMore: has\_more,

page: page + 1,

isEmpty: reset && processedOrders.length === 0

});

}

} catch (error) {

console.error('加载订单列表失败:', error);

wx.showToast({

title: '加载失败，请重试',

icon: 'none'

});

} finally {

this.setData({

loading: false,

refreshing: false,

loadingMore: false

});

if (this.data.refreshing) {

wx.stopPullDownRefresh();

}

}

},

// 处理订单数据

processOrderData(order) {

return {

...order,

formattedAmount: formatMoney(order.amount),

formattedCreateTime: formatDate(order.created\_at, 'MM-DD HH:mm'),

formattedUpdateTime: formatDate(order.updated\_at, 'MM-DD HH:mm'),

statusText: this.getStatusText(order.status),

statusColor: this.getStatusColor(order.status),

canCancel: ['pending', 'confirmed'].includes(order.status),

canPay: order.status === 'confirmed',

canViewContract: ['contract', 'active', 'completed'].includes(order.status),

progress: this.getOrderProgress(order.status)

};

},

// 获取状态文本

getStatusText(status) {

const statusMap = {

pending: '待确认',

confirmed: '已确认',

contract: '待签约',

active: '服务中',

completed: '已完成',

cancelled: '已取消'

};

return statusMap[status] || '未知状态';

},

// 获取状态颜色

getStatusColor(status) {

const colorMap = {

pending: '#ff9500',

confirmed: '#007aff',

contract: '#5856d6',

active: '#34c759',

completed: '#8e8e93',

cancelled: '#ff3b30'

};

return colorMap[status] || '#8e8e93';

},

// 获取订单进度

getOrderProgress(status) {

const progressMap = {

pending: 20,

confirmed: 40,

contract: 60,

active: 80,

completed: 100,

cancelled: 0

};

return progressMap[status] || 0;

},

// 刷新订单列表

refreshOrderList() {

this.setData({ page: 1 });

this.loadOrderList(true);

},

// 加载更多订单

loadMoreOrders() {

if (this.data.loadingMore || !this.data.hasMore) return;

this.setData({ loadingMore: true });

this.loadOrderList();

},

// 切换标签

onTabChange(e) {

const { key } = e.currentTarget.dataset;

if (key === this.data.activeTab) return;

this.setData({

activeTab: key,

page: 1

});

this.loadOrderList(true);

},

// 搜索功能

onSearchFocus() {

this.setData({ showSearch: true });

},

onSearchBlur() {

setTimeout(() => {

this.setData({ showSearch: false });

}, 200);

},

onSearchInput(e) {

this.setData({ searchKeyword: e.detail.value });

},

onSearchConfirm() {

this.refreshOrderList();

this.setData({ showSearch: false });

},

onClearSearch() {

this.setData({ searchKeyword: '' });

this.refreshOrderList();

},

// 筛选功能

onShowFilter() {

this.setData({ showFilter: true });

},

onHideFilter() {

this.setData({ showFilter: false });

},

onFilterChange(e) {

const { field, value } = e.currentTarget.dataset;

this.setData({

[`filterData.${field}`]: value

});

},

onApplyFilter() {

this.setData({ showFilter: false });

this.refreshOrderList();

},

onResetFilter() {

this.setData({

filterData: {

dateRange: '',

amountRange: '',

productType: ''

}

});

},

// 订单操作

onOrderTap(e) {

const { id } = e.currentTarget.dataset;

wx.navigateTo({

url: `/pages/orders/detail/detail?id=${id}`

});

},

onOrderAction(e) {

const { action, order } = e.currentTarget.dataset;

switch (action) {

case 'cancel':

this.cancelOrder(order);

break;

case 'pay':

this.payOrder(order);

break;

case 'contract':

this.viewContract(order);

break;

case 'more':

this.showMoreActions(order);

break;

default:

break;

}

},

// 取消订单

cancelOrder(order) {

wx.showModal({

title: '取消订单',

content: '确认要取消这个订单吗？',

success: async (res) => {

if (res.confirm) {

try {

wx.showLoading({ title: '处理中...' });

const response = await api.request({

url: `/orders/${order.id}/cancel`,

method: 'POST'

});

if (response.success) {

wx.showToast({

title: '订单已取消',

icon: 'success'

});

this.refreshOrderList();

} else {

throw new Error(response.message);

}

} catch (error) {

console.error('取消订单失败:', error);

wx.showToast({

title: error.message || '取消失败',

icon: 'none'

});

} finally {

wx.hideLoading();

}

}

}

});

},

// 支付订单

payOrder(order) {

wx.navigateTo({

url: `/pages/orders/pay/pay?id=${order.id}`

});

},

// 查看合同

viewContract(order) {

wx.navigateTo({

url: `/pages/contract/detail/detail?orderId=${order.id}`

});

},

// 显示更多操作

showMoreActions(order) {

const actions = [];

if (order.canCancel) {

actions.push({ name: '取消订单', action: 'cancel', color: '#ff3b30' });

}

actions.push(

{ name: '联系客服', action: 'contact' },

{ name: '分享订单', action: 'share' }

);

this.setData({

showActionSheet: true,

actionSheetActions: actions,

selectedOrder: order

});

},

// 操作面板选择

onActionSelect(e) {

const { action } = e.detail;

const order = this.data.selectedOrder;

this.setData({ showActionSheet: false });

switch (action) {

case 'cancel':

this.cancelOrder(order);

break;

case 'contact':

this.contactService(order);

break;

case 'share':

this.shareOrder(order);

break;

default:

break;

}

},

// 联系客服

contactService(order) {

wx.makePhoneCall({

phoneNumber: '400-888-8888',

fail: () => {

wx.showToast({

title: '拨号失败',

icon: 'none'

});

}

});

},

// 分享订单

shareOrder(order) {

wx.showShareMenu({

withShareTicket: true,

menus: ['shareAppMessage', 'shareTimeline']

});

},

// 重试加载

onRetryLoad() {

this.setData({

loading: true,

isEmpty: false

});

this.loadOrderList(true);

},

// 创建新订单

onCreateOrder() {

wx.navigateTo({

url: '/pages/products/index/index'

});

}

});

现在开发订单列表页面的模板文件：

订单列表页面模板 - pages/orders/index/index.wxml

<!--pages/orders/index/index.wxml-->

<view class="container">

<!-- 搜索栏 -->

<view class="search-bar">

<view class="search-input-wrapper">

<image class="search-icon" src="/images/icons/search.png" mode="aspectFit"></image>

<input

class="search-input"

placeholder="搜索订单号、产品名称"

value="{{searchKeyword}}"

bindinput="onSearchInput"

bindfocus="onSearchFocus"

bindblur="onSearchBlur"

bindconfirm="onSearchConfirm"

/>

<view class="search-clear {{searchKeyword ? 'show' : ''}}" bindtap="onClearSearch">

<image src="/images/icons/close.png" mode="aspectFit"></image>

</view>

</view>

<view class="filter-btn" bindtap="onShowFilter">

<image src="/images/icons/filter.png" mode="aspectFit"></image>

</view>

</view>

<!-- 状态标签栏 -->

<scroll-view class="tab-bar" scroll-x="true">

<view class="tab-list">

<view

class="tab-item {{activeTab === item.key ? 'active' : ''}}"

wx:for="{{tabList}}"

wx:key="key"

data-key="{{item.key}}"

bindtap="onTabChange"

>

<text class="tab-name">{{item.name}}</text>

<text class="tab-count" wx:if="{{item.count > 0}}">({{item.count}})</text>

</view>

</view>

</scroll-view>

<!-- 订单列表 -->

<view class="order-list-container">

<!-- 加载状态 -->

<van-loading size="24px" vertical wx:if="{{loading && orderList.length === 0}}">

</van-loading>

<!-- 空状态 -->

<view class="empty-state" wx:if="{{!loading && orderList.length === 0}}">

<image class="empty-icon" src="/images/empty/orders.png" mode="aspectFit"></image>

<text class="empty-text">暂无订单</text>

<text class="empty-desc">您还没有任何订单，快去选购产品吧~</text>

<button class="empty-btn" bindtap="onGoShopping">去选购</button>

</view>

<!-- 订单列表 -->

<view class="order-list" wx:if="{{orderList.length > 0}}">

<view

class="order-item"

wx:for="{{orderList}}"

wx:key="id"

data-order="{{item}}"

bindtap="onOrderItemTap"

>

<!-- 订单头部 -->

<view class="order-header">

<view class="order-info">

<text class="order-number">订单号：{{item.orderNumber}}</text>

<text class="order-date">{{item.createdAt}}</text>

</view>

<view class="order-status {{item.statusClass}}">

<text>{{item.statusText}}</text>

</view>

</view>

<!-- 产品信息 -->

<view class="product-info">

<image class="product-image" src="{{item.productImage}}" mode="aspectFill"></image>

<view class="product-details">

<text class="product-name">{{item.productName}}</text>

<text class="product-desc">{{item.productDesc}}</text>

<view class="product-specs">

<text class="spec-item">容量：{{item.capacity}}</text>

<text class="spec-item">期限：{{item.servicePeriod}}</text>

</view>

</view>

<view class="product-price">

<text class="price-symbol">¥</text>

<text class="price-amount">{{item.amount}}</text>

</view>

</view>

<!-- 服务信息 -->

<view class="service-info" wx:if="{{item.assignedEmployee}}">

<view class="service-item">

<image class="service-icon" src="/images/icons/service.png" mode="aspectFit"></image>

<text class="service-text">服务经理：{{item.assignedEmployee.name}}</text>

<text class="service-phone" catchtap="onCallService" data-phone="{{item.assignedEmployee.phone}}">

{{item.assignedEmployee.phone}}

</text>

</view>

</view>

<!-- 进度条 -->

<view class="progress-bar" wx:if="{{item.showProgress}}">

<view class="progress-steps">

<view

class="step-item {{index <= item.currentStep ? 'completed' : ''}}"

wx:for="{{item.progressSteps}}"

wx:key="id"

wx:for-index="index"

>

<view class="step-dot"></view>

<text class="step-text">{{item.name}}</text>

</view>

</view>

<view class="progress-line">

<view class="progress-fill" style="width: {{item.progressPercent}}%"></view>

</view>

</view>

<!-- 操作按钮 -->

<view class="order-actions">

<button

class="action-btn secondary"

wx:if="{{item.canCancel}}"

catchtap="onCancelOrder"

data-order-id="{{item.id}}"

>

取消订单

</button>

<button

class="action-btn secondary"

wx:if="{{item.canModify}}"

catchtap="onModifyOrder"

data-order-id="{{item.id}}"

>

修改订单

</button>

<button

class="action-btn primary"

wx:if="{{item.canPay}}"

catchtap="onPayOrder"

data-order-id="{{item.id}}"

>

立即支付

</button>

<button

class="action-btn secondary"

wx:if="{{item.canViewContract}}"

catchtap="onViewContract"

data-order-id="{{item.id}}"

>

查看合同

</button>

<button

class="action-btn primary"

wx:if="{{item.canConfirm}}"

catchtap="onConfirmOrder"

data-order-id="{{item.id}}"

>

确认收货

</button>

</view>

</view>

</view>

<!-- 加载更多 -->

<view class="load-more" wx:if="{{hasMore && orderList.length > 0}}">

<van-loading size="16px" wx:if="{{loadingMore}}">加载中...</van-loading>

<text wx:else>上拉加载更多</text>

</view>

<!-- 没有更多数据 -->

<view class="no-more" wx:if="{{!hasMore && orderList.length > 0}}">

<text>已显示全部订单</text>

</view>

</view>

<!-- 筛选弹窗 -->

<van-popup

show="{{showFilter}}"

position="bottom"

bind:close="onCloseFilter"

round="{{true}}"

close-icon="true"

>

<view class="filter-popup">

<view class="filter-header">

<text class="filter-title">筛选条件</text>

<text class="filter-reset" bindtap="onResetFilter">重置</text>

</view>

<view class="filter-content">

<!-- 订单状态筛选 -->

<view class="filter-section">

<view class="filter-section-title">订单状态</view>

<view class="filter-options">

<view

class="filter-option {{item.checked ? 'checked' : ''}}"

wx:for="{{filterOptions.status}}"

wx:key="value"

data-type="status"

data-value="{{item.value}}"

bindtap="onFilterOptionTap"

>

<text>{{item.label}}</text>

<image class="check-icon" src="/images/icons/check.png" mode="aspectFit"></image>

</view>

</view>

</view>

<!-- 时间范围筛选 -->

<view class="filter-section">

<view class="filter-section-title">下单时间</view>

<view class="filter-options">

<view

class="filter-option {{item.checked ? 'checked' : ''}}"

wx:for="{{filterOptions.dateRange}}"

wx:key="value"

data-type="dateRange"

data-value="{{item.value}}"

bindtap="onFilterOptionTap"

>

<text>{{item.label}}</text>

<image class="check-icon" src="/images/icons/check.png" mode="aspectFit"></image>

</view>

</view>

</view>

<!-- 金额范围筛选 -->

<view class="filter-section">

<view class="filter-section-title">订单金额</view>

<view class="filter-options">

<view

class="filter-option {{item.checked ? 'checked' : ''}}"

wx:for="{{filterOptions.amountRange}}"

wx:key="value"

data-type="amountRange"

data-value="{{item.value}}"

bindtap="onFilterOptionTap"

>

<text>{{item.label}}</text>

<image class="check-icon" src="/images/icons/check.png" mode="aspectFit"></image>

</view>

</view>

</view>

</view>

<view class="filter-footer">

<button class="filter-cancel" bindtap="onCloseFilter">取消</button>

<button class="filter-confirm" bindtap="onConfirmFilter">确定</button>

</view>

</view>

</van-popup>

<!-- 操作确认弹窗 -->

<van-dialog

show="{{showConfirmDialog}}"

title="{{confirmDialog.title}}"

message="{{confirmDialog.message}}"

show-cancel-button

bind:confirm="onConfirmAction"

bind:cancel="onCancelAction"

/>

<!-- 客服联系弹窗 -->

<van-action-sheet

show="{{showContactSheet}}"

actions="{{contactActions}}"

cancel-text="取消"

bind:select="onContactSelect"

bind:close="onCloseContactSheet"

/>

</view>

现在继续开发下一个模块：**订单列表页面样式文件 - pages/orders/index/index.wxss**

订单列表页面样式文件 - pages/orders/index/index.wxss

/\* pages/orders/index/index.wxss \*/

.container {

min-height: 100vh;

background-color: #f8f9fa;

}

/\* 搜索栏样式 \*/

.search-bar {

display: flex;

align-items: center;

padding: 20rpx 30rpx;

background-color: #fff;

border-bottom: 1rpx solid #eee;

}

.search-input-wrapper {

flex: 1;

position: relative;

display: flex;

align-items: center;

background-color: #f5f5f5;

border-radius: 50rpx;

padding: 0 30rpx;

margin-right: 20rpx;

}

.search-icon {

width: 32rpx;

height: 32rpx;

margin-right: 20rpx;

opacity: 0.6;

}

.search-input {

flex: 1;

height: 80rpx;

font-size: 28rpx;

color: #333;

}

.search-input::placeholder {

color: #999;

}

.search-clear {

position: absolute;

right: 30rpx;

width: 32rpx;

height: 32rpx;

opacity: 0;

transition: opacity 0.3s;

}

.search-clear.show {

opacity: 0.6;

}

.search-clear image {

width: 100%;

height: 100%;

}

.filter-btn {

width: 80rpx;

height: 80rpx;

display: flex;

align-items: center;

justify-content: center;

background-color: #f5f5f5;

border-radius: 50rpx;

}

.filter-btn image {

width: 32rpx;

height: 32rpx;

}

/\* 标签栏样式 \*/

.tab-bar {

background-color: #fff;

border-bottom: 1rpx solid #eee;

white-space: nowrap;

}

.tab-list {

display: flex;

padding: 0 30rpx;

}

.tab-item {

flex-shrink: 0;

padding: 30rpx 40rpx;

position: relative;

display: flex;

align-items: center;

}

.tab-item.active .tab-name {

color: var(--primary-color);

font-weight: 600;

}

.tab-item.active::after {

content: '';

position: absolute;

bottom: 0;

left: 50%;

transform: translateX(-50%);

width: 60rpx;

height: 6rpx;

background-color: var(--primary-color);

border-radius: 3rpx;

}

.tab-name {

font-size: 28rpx;

color: #666;

transition: color 0.3s;

}

.tab-count {

font-size: 24rpx;

color: #999;

margin-left: 8rpx;

}

/\* 订单列表容器 \*/

.order-list-container {

padding: 20rpx 30rpx;

}

/\* 空状态样式 \*/

.empty-state {

display: flex;

flex-direction: column;

align-items: center;

padding: 120rpx 0;

text-align: center;

}

.empty-icon {

width: 200rpx;

height: 200rpx;

margin-bottom: 40rpx;

opacity: 0.5;

}

.empty-text {

font-size: 32rpx;

color: #666;

margin-bottom: 20rpx;

}

.empty-desc {

font-size: 26rpx;

color: #999;

margin-bottom: 60rpx;

}

.empty-btn {

background-color: var(--primary-color);

color: #fff;

border-radius: 50rpx;

padding: 20rpx 60rpx;

font-size: 28rpx;

border: none;

}

/\* 订单列表样式 \*/

.order-list {

display: flex;

flex-direction: column;

gap: 20rpx;

}

.order-item {

background-color: #fff;

border-radius: 20rpx;

padding: 30rpx;

box-shadow: 0 2rpx 20rpx rgba(0, 0, 0, 0.05);

}

/\* 订单头部 \*/

.order-header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 30rpx;

padding-bottom: 20rpx;

border-bottom: 1rpx solid #f0f0f0;

}

.order-info {

flex: 1;

}

.order-number {

display: block;

font-size: 28rpx;

color: #333;

font-weight: 600;

margin-bottom: 10rpx;

}

.order-date {

font-size: 24rpx;

color: #999;

}

.order-status {

padding: 8rpx 20rpx;

border-radius: 30rpx;

font-size: 24rpx;

font-weight: 500;

}

.order-status.pending {

background-color: #fff3cd;

color: #856404;

}

.order-status.processing {

background-color: #d1ecf1;

color: #0c5460;

}

.order-status.completed {

background-color: #d4edda;

color: #155724;

}

.order-status.cancelled {

background-color: #f8d7da;

color: #721c24;

}

/\* 产品信息 \*/

.product-info {

display: flex;

align-items: flex-start;

margin-bottom: 30rpx;

}

.product-image {

width: 120rpx;

height: 120rpx;

border-radius: 16rpx;

margin-right: 24rpx;

flex-shrink: 0;

}

.product-details {

flex: 1;

min-width: 0;

}

.product-name {

display: block;

font-size: 30rpx;

color: #333;

font-weight: 600;

margin-bottom: 12rpx;

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap;

}

.product-desc {

display: block;

font-size: 26rpx;

color: #666;

margin-bottom: 16rpx;

line-height: 1.4;

}

.product-specs {

display: flex;

flex-wrap: wrap;

gap: 20rpx;

}

.spec-item {

font-size: 24rpx;

color: #999;

background-color: #f5f5f5;

padding: 8rpx 16rpx;

border-radius: 12rpx;

}

.product-price {

display: flex;

align-items: baseline;

flex-shrink: 0;

margin-left: 20rpx;

}

.price-symbol {

font-size: 24rpx;

color: var(--price-color);

margin-right: 4rpx;

}

.price-amount {

font-size: 36rpx;

color: var(--price-color);

font-weight: 600;

}

/\* 服务信息 \*/

.service-info {

margin-bottom: 30rpx;

}

.service-item {

display: flex;

align-items: center;

padding: 20rpx;

background-color: #f8f9fa;

border-radius: 16rpx;

}

.service-icon {

width: 32rpx;

height: 32rpx;

margin-right: 16rpx;

}

.service-text {

flex: 1;

font-size: 26rpx;

color: #666;

}

.service-phone {

font-size: 26rpx;

color: var(--primary-color);

font-weight: 500;

text-decoration: underline;

}

/\* 进度条样式 \*/

.progress-bar {

margin-bottom: 30rpx;

}

.progress-steps {

display: flex;

justify-content: space-between;

margin-bottom: 20rpx;

position: relative;

}

.step-item {

display: flex;

flex-direction: column;

align-items: center;

flex: 1;

position: relative;

}

.step-dot {

width: 20rpx;

height: 20rpx;

border-radius: 50%;

background-color: #ddd;

margin-bottom: 12rpx;

transition: background-color 0.3s;

}

.step-item.completed .step-dot {

background-color: var(--primary-color);

}

.step-text {

font-size: 22rpx;

color: #999;

text-align: center;

white-space: nowrap;

}

.step-item.completed .step-text {

color: var(--primary-color);

}

.progress-line {

height: 4rpx;

background-color: #eee;

border-radius: 2rpx;

position: relative;

overflow: hidden;

}

.progress-fill {

height: 100%;

background-color: var(--primary-color);

border-radius: 2rpx;

transition: width 0.3s ease;

}

/\* 操作按钮 \*/

.order-actions {

display: flex;

justify-content: flex-end;

gap: 20rpx;

}

.action-btn {

min-width: 160rpx;

height: 68rpx;

line-height: 68rpx;

text-align: center;

border-radius: 34rpx;

font-size: 26rpx;

border: none;

padding: 0 30rpx;

}

.action-btn.primary {

background-color: var(--primary-color);

color: #fff;

}

.action-btn.secondary {

background-color: #f5f5f5;

color: #666;

}

.action-btn:active {

opacity: 0.7;

}

/\* 加载更多 \*/

.load-more, .no-more {

display: flex;

justify-content: center;

align-items: center;

padding: 40rpx 0;

font-size: 26rpx;

color: #999;

}

/\* 筛选弹窗样式 \*/

.filter-popup {

background-color: #fff;

border-radius: 20rpx 20rpx 0 0;

max-height: 80vh;

}

.filter-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 40rpx 30rpx 20rpx;

border-bottom: 1rpx solid #eee;

}

.filter-title {

font-size: 32rpx;

font-weight: 600;

color: #333;

}

.filter-reset {

font-size: 28rpx;

color: var(--primary-color);

}

.filter-content {

padding: 0 30rpx;

max-height: 60vh;

overflow-y: auto;

}

.filter-section {

padding: 40rpx 0;

border-bottom: 1rpx solid #f0f0f0;

}

.filter-section:last-child {

border-bottom: none;

}

.filter-section-title {

font-size: 30rpx;

font-weight: 600;

color: #333;

margin-bottom: 30rpx;

}

.filter-options {

display: flex;

flex-direction: column;

gap: 20rpx;

}

.filter-option {

display: flex;

justify-content: space-between;

align-items: center;

padding: 20rpx 0;

position: relative;

}

.filter-option text {

font-size: 28rpx;

color: #666;

}

.filter-option.checked text {

color: var(--primary-color);

}

.check-icon {

width: 32rpx;

height: 32rpx;

opacity: 0;

transition: opacity 0.3s;

}

.filter-option.checked .check-icon {

opacity: 1;

}

.filter-footer {

display: flex;

padding: 30rpx;

gap: 20rpx;

border-top: 1rpx solid #eee;

background-color: #fff;

}

.filter-cancel, .filter-confirm {

flex: 1;

height: 88rpx;

line-height: 88rpx;

text-align: center;

border-radius: 44rpx;

font-size: 30rpx;

border: none;

}

.filter-cancel {

background-color: #f5f5f5;

color: #666;

}

.filter-confirm {

background-color: var(--primary-color);

color: #fff;

}

/\* 响应式适配 \*/

@media (max-width: 375px) {

.order-item {

padding: 24rpx;

}

.product-image {

width: 100rpx;

height: 100rpx;

}

.action-btn {

min-width: 120rpx;

padding: 0 20rpx;

font-size: 24rpx;

}

}

/\* 暗色模式适配 \*/

@media (prefers-color-scheme: dark) {

.container {

background-color: #1a1a1a;

}

.search-bar, .tab-bar, .order-item {

background-color: #2a2a2a;

border-color: #404040;

}

.search-input-wrapper, .action-btn.secondary {

background-color: #404040;

}

.tab-name, .order-number, .product-name {

color: #fff;

}

.order-date, .product-desc, .service-text {

color: #ccc;

}

}

继续开发下一个模块：**订单列表页面配置 - pages/orders/index/index.json**

{

"navigationBarTitleText": "我的订单",

"navigationBarBackgroundColor": "#ffffff",

"navigationBarTextStyle": "black",

"backgroundColor": "#f8f9fa",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 100,

"usingComponents": {

"van-loading": "@vant/weapp/loading/index",

"van-popup": "@vant/weapp/popup/index",

"van-dialog": "@vant/weapp/dialog/index",

"van-action-sheet": "@vant/weapp/action-sheet/index"

}

}

现在继续开发下一个模块：**订单详情页面逻辑 - pages/orders/detail/detail.js**

订单详情页面逻辑 - pages/orders/detail/detail.js

// pages/orders/detail/detail.js

const api = require('../../../utils/api');

const util = require('../../../utils/util');

Page({

data: {

orderId: null,

orderInfo: null,

loading: true,

refreshing: false,

// 订单状态流程配置

statusFlow: [

{ key: 'pending', name: '待确认', icon: 'pending' },

{ key: 'confirmed', name: '商务洽谈', icon: 'chat' },

{ key: 'contract', name: '合同签署', icon: 'contract' },

{ key: 'service', name: '服务开通', icon: 'service' },

{ key: 'completed', name: '服务中', icon: 'completed' }

],

// 操作确认弹窗

showConfirmDialog: false,

confirmDialog: {

title: '',

message: '',

action: ''

},

// 联系服务经理操作表

showContactSheet: false,

contactActions: [

{ name: '拨打电话', value: 'call' },

{ name: '发送短信', value: 'sms' },

{ name: '微信联系', value: 'wechat' }

],

// 支付方式选择

showPaymentSheet: false,

paymentMethods: [

{

name: '微信支付',

value: 'wechat',

icon: '/images/payment/wechat.png',

desc: '推荐使用，安全便捷'

},

{

name: '支付宝',

value: 'alipay',

icon: '/images/payment/alipay.png',

desc: '支持花呗分期付款'

},

{

name: '银行卡支付',

value: 'bank',

icon: '/images/payment/bank.png',

desc: '支持储蓄卡和信用卡'

}

],

// 合同预览

showContractPreview: false,

contractUrl: ''

},

onLoad(options) {

console.log('订单详情页面加载:', options);

if (options.id) {

this.setData({ orderId: options.id });

this.loadOrderDetail();

} else {

this.showError('订单ID不能为空');

}

},

onShow() {

// 页面显示时刷新数据

if (this.data.orderId) {

this.loadOrderDetail();

}

},

onPullDownRefresh() {

this.refreshOrderDetail();

},

onShareAppMessage() {

const orderInfo = this.data.orderInfo;

return {

title: `我的订单 - ${orderInfo?.productName || '电力服务'}`,

path: `/pages/orders/detail/detail?id=${this.data.orderId}`,

imageUrl: orderInfo?.productImage || '/images/share/order.png'

};

},

// 加载订单详情

async loadOrderDetail() {

try {

this.setData({ loading: true });

const response = await api.getOrderDetail(this.data.orderId);

if (response.success) {

const orderInfo = this.processOrderData(response.data);

this.setData({

orderInfo,

loading: false

});

// 设置页面标题

wx.setNavigationBarTitle({

title: `订单详情 - ${orderInfo.orderNo}`

});

} else {

this.showError(response.message || '获取订单详情失败');

}

} catch (error) {

console.error('加载订单详情失败:', error);

this.showError('网络错误，请重试');

} finally {

this.setData({ loading: false });

wx.stopPullDownRefresh();

}

},

// 刷新订单详情

async refreshOrderDetail() {

this.setData({ refreshing: true });

await this.loadOrderDetail();

this.setData({ refreshing: false });

},

// 处理订单数据

processOrderData(orderData) {

return {

...orderData,

// 格式化创建时间

createTimeFormatted: util.formatTime(new Date(orderData.createdAt)),

// 格式化更新时间

updateTimeFormatted: util.formatTime(new Date(orderData.updatedAt)),

// 格式化服务时间

serviceStartDateFormatted: orderData.serviceStartDate ?

util.formatDate(new Date(orderData.serviceStartDate)) : '',

serviceEndDateFormatted: orderData.serviceEndDate ?

util.formatDate(new Date(orderData.serviceEndDate)) : '',

// 格式化金额

amountFormatted: util.formatCurrency(orderData.amount),

// 计算当前状态在流程中的位置

currentStatusIndex: this.data.statusFlow.findIndex(item => item.key === orderData.status),

// 判断是否可以取消

canCancel: ['pending', 'confirmed'].includes(orderData.status),

// 判断是否可以支付

canPay: orderData.status === 'confirmed' && !orderData.isPaid,

// 判断是否可以查看合同

canViewContract: ['contract', 'service', 'completed'].includes(orderData.status),

// 格式化联系人信息

managerInfo: orderData.assignedEmployee ? {

...orderData.assignedEmployee,

phoneFormatted: this.formatPhone(orderData.assignedEmployee.phone)

} : null

};

},

// 格式化手机号

formatPhone(phone) {

if (!phone) return '';

return phone.replace(/(\d{3})(\d{4})(\d{4})/, '$1\*\*\*\*$3');

},

// 取消订单

onCancelOrder() {

this.setData({

showConfirmDialog: true,

confirmDialog: {

title: '取消订单',

message: '确定要取消这个订单吗？取消后无法恢复。',

action: 'cancel'

}

});

},

// 确认对话框操作

async onConfirmDialogConfirm() {

const action = this.data.confirmDialog.action;

this.setData({ showConfirmDialog: false });

if (action === 'cancel') {

await this.cancelOrder();

}

},

// 取消确认对话框

onConfirmDialogCancel() {

this.setData({ showConfirmDialog: false });

},

// 执行取消订单

async cancelOrder() {

try {

wx.showLoading({ title: '取消中...' });

const response = await api.cancelOrder(this.data.orderId);

if (response.success) {

wx.showToast({

title: '订单已取消',

icon: 'success'

});

// 刷新订单信息

await this.loadOrderDetail();

// 触发父页面刷新

const pages = getCurrentPages();

const prevPage = pages[pages.length - 2];

if (prevPage) {

prevPage.refreshOrderList && prevPage.refreshOrderList();

}

} else {

wx.showToast({

title: response.message || '取消失败',

icon: 'none'

});

}

} catch (error) {

console.error('取消订单失败:', error);

wx.showToast({

title: '网络错误，请重试',

icon: 'none'

});

} finally {

wx.hideLoading();

}

},

// 支付订单

onPayOrder() {

this.setData({ showPaymentSheet: true });

},

// 支付方式选择

onPaymentMethodSelect(e) {

const method = e.currentTarget.dataset.method;

this.setData({ showPaymentSheet: false });

this.processPayment(method);

},

// 取消支付选择

onPaymentSheetCancel() {

this.setData({ showPaymentSheet: false });

},

// 处理支付

async processPayment(method) {

try {

wx.showLoading({ title: '发起支付...' });

const response = await api.createPayment({

orderId: this.data.orderId,

paymentMethod: method

});

if (response.success) {

if (method === 'wechat') {

// 微信支付

await this.processWechatPayment(response.data);

} else if (method === 'alipay') {

// 支付宝支付

this.processAlipayPayment(response.data);

} else if (method === 'bank') {

// 银行卡支付

this.processBankPayment(response.data);

}

} else {

wx.showToast({

title: response.message || '支付发起失败',

icon: 'none'

});

}

} catch (error) {

console.error('支付处理失败:', error);

wx.showToast({

title: '网络错误，请重试',

icon: 'none'

});

} finally {

wx.hideLoading();

}

},

// 微信支付处理

async processWechatPayment(paymentData) {

return new Promise((resolve, reject) => {

wx.requestPayment({

timeStamp: paymentData.timeStamp,

nonceStr: paymentData.nonceStr,

package: paymentData.package,

signType: paymentData.signType,

paySign: paymentData.paySign,

success: (res) => {

wx.showToast({

title: '支付成功',

icon: 'success'

});

this.loadOrderDetail(); // 刷新订单状态

resolve(res);

},

fail: (res) => {

if (res.errMsg !== 'requestPayment:fail cancel') {

wx.showToast({

title: '支付失败',

icon: 'none'

});

}

reject(res);

}

});

});

},

// 支付宝支付处理

processAlipayPayment(paymentData) {

wx.showToast({

title: '暂不支持支付宝',

icon: 'none'

});

},

// 银行卡支付处理

processBankPayment(paymentData) {

wx.navigateTo({

url: `/pages/payment/bank/bank?orderId=${this.data.orderId}`

});

},

// 联系服务经理

onContactManager() {

if (!this.data.orderInfo.managerInfo) {

wx.showToast({

title: '暂未分配服务经理',

icon: 'none'

});

return;

}

this.setData({ showContactSheet: true });

},

// 联系方式选择

onContactMethodSelect(e) {

const method = e.currentTarget.dataset.method;

const managerInfo = this.data.orderInfo.managerInfo;

this.setData({ showContactSheet: false });

switch (method) {

case 'call':

wx.makePhoneCall({

phoneNumber: managerInfo.phone,

fail: (res) => {

wx.showToast({

title: '拨打失败',

icon: 'none'

});

}

});

break;

case 'sms':

wx.showToast({

title: '短信功能开发中',

icon: 'none'

});

break;

case 'wechat':

wx.showToast({

title: '微信联系功能开发中',

icon: 'none'

});

break;

}

},

// 取消联系选择

onContactSheetCancel() {

this.setData({ showContactSheet: false });

},

// 查看合同

onViewContract() {

if (!this.data.orderInfo.contractUrl) {

wx.showToast({

title: '合同文件不存在',

icon: 'none'

});

return;

}

// 预览合同文件

wx.downloadFile({

url: this.data.orderInfo.contractUrl,

success: (res) => {

wx.openDocument({

filePath: res.tempFilePath,

fileType: 'pdf',

fail: (error) => {

console.error('打开合同文件失败:', error);

wx.showToast({

title: '文件打开失败',

icon: 'none'

});

}

});

},

fail: (error) => {

console.error('下载合同文件失败:', error);

wx.showToast({

title: '文件下载失败',

icon: 'none'

});

}

});

},

// 再次购买

onBuyAgain() {

const productId = this.data.orderInfo.productId;

if (productId) {

wx.navigateTo({

url: `/pages/products/detail/detail?id=${productId}`

});

}

},

// 申请退款

onRequestRefund() {

wx.navigateTo({

url: `/pages/refund/apply/apply?orderId=${this.data.orderId}`

});

},

// 查看物流

onViewLogistics() {

wx.navigateTo({

url: `/pages/logistics/detail/detail?orderId=${this.data.orderId}`

});

},

// 评价订单

onEvaluateOrder() {

wx.navigateTo({

url: `/pages/evaluation/create/create?orderId=${this.data.orderId}`

});

},

// 复制订单号

onCopyOrderNo() {

wx.setClipboardData({

data: this.data.orderInfo.orderNo,

success: () => {

wx.showToast({

title: '订单号已复制',

icon: 'success'

});

}

});

},

// 复制联系方式

onCopyPhone() {

const phone = this.data.orderInfo.managerInfo?.phone;

if (phone) {

wx.setClipboardData({

data: phone,

success: () => {

wx.showToast({

title: '电话号码已复制',

icon: 'success'

});

}

});

}

},

// 显示错误信息

showError(message) {

wx.showToast({

title: message,

icon: 'none',

duration: 3000

});

// 如果是严重错误，返回上一页

setTimeout(() => {

wx.navigateBack();

}, 2000);

}

});

现在继续开发下一个模块 - 订单详情页面模板文件：

订单详情页面模板 - pages/orders/detail/detail.wxml

<!--pages/orders/detail/detail.wxml-->

<view class="order-detail-container">

<!-- 加载状态 -->

<view wx:if="{{loading}}" class="loading-container">

<view class="loading-spinner"></view>

<text class="loading-text">加载中...</text>

</view>

<!-- 订单详情内容 -->

<view wx:else class="order-detail-content">

<!-- 订单状态卡片 -->

<view class="status-card">

<view class="status-header">

<view class="status-icon {{orderInfo.status}}">

<text class="iconfont icon-{{orderInfo.status}}"></text>

</view>

<view class="status-info">

<text class="status-title">{{orderInfo.statusText}}</text>

<text class="status-desc">{{orderInfo.statusDesc}}</text>

</view>

</view>

<!-- 订单进度条 -->

<view class="progress-container">

<view class="progress-line">

<view class="progress-fill" style="width: {{(orderInfo.currentStatusIndex + 1) / statusFlow.length \* 100}}%;"></view>

</view>

<view class="progress-steps">

<view

wx:for="{{statusFlow}}"

wx:key="key"

class="progress-step {{index <= orderInfo.currentStatusIndex ? 'active' : ''}}"

>

<view class="step-icon">

<text class="iconfont icon-{{item.icon}}"></text>

</view>

<text class="step-name">{{item.name}}</text>

</view>

</view>

</view>

</view>

<!-- 订单基本信息 -->

<view class="info-card">

<view class="card-header">

<text class="card-title">订单信息</text>

<view class="copy-btn" bindtap="onCopyOrderNo">

<text class="iconfont icon-copy"></text>

<text>复制订单号</text>

</view>

</view>

<view class="info-list">

<view class="info-item">

<text class="info-label">订单编号</text>

<text class="info-value">{{orderInfo.orderNo}}</text>

</view>

<view class="info-item">

<text class="info-label">下单时间</text>

<text class="info-value">{{orderInfo.createTimeFormatted}}</text>

</view>

<view class="info-item">

<text class="info-label">订单金额</text>

<text class="info-value price">{{orderInfo.amountFormatted}}</text>

</view>

<view class="info-item" wx:if="{{orderInfo.serviceStartDateFormatted}}">

<text class="info-label">服务开始</text>

<text class="info-value">{{orderInfo.serviceStartDateFormatted}}</text>

</view>

<view class="info-item" wx:if="{{orderInfo.serviceEndDateFormatted}}">

<text class="info-label">服务结束</text>

<text class="info-value">{{orderInfo.serviceEndDateFormatted}}</text>

</view>

</view>

</view>

<!-- 产品信息 -->

<view class="product-card">

<view class="card-header">

<text class="card-title">产品信息</text>

</view>

<view class="product-info">

<image class="product-image" src="{{orderInfo.productImage}}" mode="aspectFill"></image>

<view class="product-details">

<text class="product-name">{{orderInfo.productName}}</text>

<text class="product-type">{{orderInfo.productType}}</text>

<text class="product-desc">{{orderInfo.productDesc}}</text>

<view class="product-price">

<text class="price-label">单价：</text>

<text class="price-value">{{orderInfo.productPrice}}</text>

</view>

</view>

</view>

</view>

<!-- 服务经理信息 -->

<view class="manager-card" wx:if="{{orderInfo.managerInfo}}">

<view class="card-header">

<text class="card-title">服务经理</text>

<view class="contact-btn" bindtap="onContactManager">

<text class="iconfont icon-phone"></text>

<text>联系</text>

</view>

</view>

<view class="manager-info">

<image class="manager-avatar" src="{{orderInfo.managerInfo.avatar}}" mode="aspectFill"></image>

<view class="manager-details">

<text class="manager-name">{{orderInfo.managerInfo.name}}</text>

<text class="manager-title">{{orderInfo.managerInfo.position}}</text>

<view class="manager-contact">

<text class="contact-phone" bindtap="onCopyPhone">{{orderInfo.managerInfo.phoneFormatted}}</text>

<text class="contact-email">{{orderInfo.managerInfo.email}}</text>

</view>

</view>

</view>

</view>

<!-- 合同信息 -->

<view class="contract-card" wx:if="{{orderInfo.canViewContract}}">

<view class="card-header">

<text class="card-title">合同信息</text>

<view class="view-btn" bindtap="onViewContract">

<text class="iconfont icon-document"></text>

<text>查看合同</text>

</view>

</view>

<view class="contract-info">

<view class="contract-item">

<text class="contract-label">合同编号</text>

<text class="contract-value">{{orderInfo.contractNo}}</text>

</view>

<view class="contract-item">

<text class="contract-label">签署时间</text>

<text class="contract-value">{{orderInfo.contractSignTime}}</text>

</view>

<view class="contract-item">

<text class="contract-label">合同状态</text>

<text class="contract-value {{orderInfo.contractStatus}}">{{orderInfo.contractStatusText}}</text>

</view>

</view>

</view>

<!-- 跟进记录 -->

<view class="follow-card" wx:if="{{orderInfo.followLogs && orderInfo.followLogs.length > 0}}">

<view class="card-header">

<text class="card-title">跟进记录</text>

</view>

<view class="follow-timeline">

<view

wx:for="{{orderInfo.followLogs}}"

wx:key="id"

class="timeline-item"

>

<view class="timeline-dot"></view>

<view class="timeline-content">

<view class="timeline-header">

<text class="timeline-operator">{{item.employeeName}}</text>

<text class="timeline-time">{{item.createTimeFormatted}}</text>

</view>

<text class="timeline-content-text">{{item.content}}</text>

</view>

</view>

</view>

</view>

<!-- 订单操作 -->

<view class="action-card">

<view class="action-buttons">

<!-- 支付按钮 -->

<button

wx:if="{{orderInfo.canPay}}"

class="action-btn primary"

bindtap="onPayOrder"

>

立即支付

</button>

<!-- 取消订单 -->

<button

wx:if="{{orderInfo.canCancel}}"

class="action-btn cancel"

bindtap="onCancelOrder"

>

取消订单

</button>

<!-- 再次购买 -->

<button

wx:if="{{orderInfo.status === 'completed'}}"

class="action-btn secondary"

bindtap="onBuyAgain"

>

再次购买

</button>

<!-- 申请退款 -->

<button

wx:if="{{orderInfo.canRefund}}"

class="action-btn refund"

bindtap="onRequestRefund"

>

申请退款

</button>

<!-- 评价订单 -->

<button

wx:if="{{orderInfo.canEvaluate}}"

class="action-btn evaluate"

bindtap="onEvaluateOrder"

>

评价服务

</button>

</view>

</view>

</view>

</view>

<!-- 确认对话框 -->

<view class="modal-overlay" wx:if="{{showConfirmDialog}}" bindtap="onConfirmDialogCancel">

<view class="modal-content" catchtap="">

<view class="modal-header">

<text class="modal-title">{{confirmDialog.title}}</text>

</view>

<view class="modal-body">

<text class="modal-message">{{confirmDialog.message}}</text>

</view>

<view class="modal-footer">

<button class="modal-btn cancel" bindtap="onConfirmDialogCancel">取消</button>

<button class="modal-btn confirm" bindtap="onConfirmDialogConfirm">确定</button>

</view>

</view>

</view>

<!-- 联系方式选择 -->

<view class="action-sheet-overlay" wx:if="{{showContactSheet}}" bindtap="onContactSheetCancel">

<view class="action-sheet" catchtap="">

<view class="action-sheet-header">

<text class="action-sheet-title">联系服务经理</text>

</view>

<view class="action-sheet-body">

<view

wx:for="{{contactActions}}"

wx:key="value"

class="action-sheet-item"

data-method="{{item.value}}"

bindtap="onContactMethodSelect"

>

<text class="action-item-text">{{item.name}}</text>

</view>

</view>

<view class="action-sheet-footer">

<button class="action-sheet-cancel" bindtap="onContactSheetCancel">取消</button>

</view>

</view>

</view>

<!-- 支付方式选择 -->

<view class="action-sheet-overlay" wx:if="{{showPaymentSheet}}" bindtap="onPaymentSheetCancel">

<view class="action-sheet" catchtap="">

<view class="action-sheet-header">

<text class="action-sheet-title">选择支付方式</text>

</view>

<view class="action-sheet-body">

<view

wx:for="{{paymentMethods}}"

wx:key="value"

class="payment-method-item"

data-method="{{item.value}}"

bindtap="onPaymentMethodSelect"

>

<image class="payment-icon" src="{{item.icon}}" mode="aspectFit"></image>

<view class="payment-info">

<text class="payment-name">{{item.name}}</text>

<text class="payment-desc">{{item.desc}}</text>

</view>

<text class="iconfont icon-arrow-right"></text>

</view>

</view>

<view class="action-sheet-footer">

<button class="action-sheet-cancel" bindtap="onPaymentSheetCancel">取消</button>

</view>

</view>

</view>

现在继续开发订单详情页面的样式文件：

订单详情页面样式文件 - pages/orders/detail/detail.wxss

/\* pages/orders/detail/detail.wxss \*/

.order-detail-container {

min-height: 100vh;

background-color: var(--bg-color-light);

padding-bottom: 120rpx; /\* 为底部操作栏留空间 \*/

}

/\* 加载状态 \*/

.loading-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #fff;

}

.loading-spinner {

width: 60rpx;

height: 60rpx;

border: 4rpx solid var(--border-color);

border-top: 4rpx solid var(--primary-color);

border-radius: 50%;

animation: spin 1s linear infinite;

}

.loading-text {

margin-top: 20rpx;

font-size: 28rpx;

color: var(--text-color-secondary);

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

/\* 订单详情内容 \*/

.order-detail-content {

padding: 20rpx;

}

/\* 卡片基础样式 \*/

.status-card,

.info-card,

.product-card,

.manager-card,

.contract-card,

.follow-card,

.action-card {

background-color: #fff;

border-radius: 16rpx;

margin-bottom: 20rpx;

overflow: hidden;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.05);

}

/\* 卡片头部 \*/

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 30rpx;

border-bottom: 1rpx solid var(--border-color);

}

.card-title {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

}

/\* 订单状态卡片 \*/

.status-card {

background: linear-gradient(135deg, var(--primary-color), var(--primary-color-dark));

color: #fff;

padding: 40rpx 30rpx;

}

.status-header {

display: flex;

align-items: center;

margin-bottom: 40rpx;

}

.status-icon {

width: 80rpx;

height: 80rpx;

border-radius: 50%;

background-color: rgba(255, 255, 255, 0.2);

display: flex;

align-items: center;

justify-content: center;

margin-right: 24rpx;

}

.status-icon .iconfont {

font-size: 36rpx;

color: #fff;

}

.status-info {

flex: 1;

}

.status-title {

display: block;

font-size: 36rpx;

font-weight: 600;

margin-bottom: 8rpx;

}

.status-desc {

font-size: 26rpx;

opacity: 0.8;

}

/\* 进度条 \*/

.progress-container {

position: relative;

}

.progress-line {

height: 4rpx;

background-color: rgba(255, 255, 255, 0.3);

border-radius: 2rpx;

margin-bottom: 30rpx;

}

.progress-fill {

height: 100%;

background-color: #fff;

border-radius: 2rpx;

transition: width 0.3s ease;

}

.progress-steps {

display: flex;

justify-content: space-between;

}

.progress-step {

flex: 1;

display: flex;

flex-direction: column;

align-items: center;

opacity: 0.6;

transition: opacity 0.3s ease;

}

.progress-step.active {

opacity: 1;

}

.step-icon {

width: 40rpx;

height: 40rpx;

border-radius: 50%;

background-color: rgba(255, 255, 255, 0.3);

display: flex;

align-items: center;

justify-content: center;

margin-bottom: 12rpx;

}

.progress-step.active .step-icon {

background-color: #fff;

}

.progress-step.active .step-icon .iconfont {

color: var(--primary-color);

}

.step-icon .iconfont {

font-size: 20rpx;

color: #fff;

}

.step-name {

font-size: 22rpx;

text-align: center;

line-height: 1.2;

}

/\* 操作按钮 \*/

.copy-btn,

.contact-btn {

background-color: rgba(255, 255, 255, 0.2);

color: #fff;

border: 1rpx solid rgba(255, 255, 255, 0.3);

border-radius: 12rpx;

padding: 12rpx 24rpx;

font-size: 24rpx;

margin-left: 16rpx;

transition: all 0.3s ease;

}

.copy-btn:active,

.contact-btn:active {

background-color: rgba(255, 255, 255, 0.3);

transform: scale(0.95);

}

/\* 信息卡片 \*/

.info-card .card-body {

padding: 30rpx;

}

.info-item {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 24rpx;

padding-bottom: 24rpx;

border-bottom: 1rpx solid var(--border-color-light);

}

.info-item:last-child {

margin-bottom: 0;

padding-bottom: 0;

border-bottom: none;

}

.info-label {

font-size: 28rpx;

color: var(--text-color-secondary);

min-width: 140rpx;

line-height: 1.4;

}

.info-value {

font-size: 28rpx;

color: var(--text-color-primary);

text-align: right;

flex: 1;

margin-left: 20rpx;

line-height: 1.4;

}

.info-value.highlight {

color: var(--primary-color);

font-weight: 600;

}

.info-value.amount {

font-size: 32rpx;

font-weight: 600;

color: var(--error-color);

}

/\* 产品卡片 \*/

.product-card .card-body {

padding: 30rpx;

}

.product-item {

display: flex;

align-items: flex-start;

padding: 20rpx 0;

border-bottom: 1rpx solid var(--border-color-light);

}

.product-item:last-child {

border-bottom: none;

}

.product-image {

width: 120rpx;

height: 120rpx;

border-radius: 12rpx;

background-color: var(--bg-color-light);

margin-right: 20rpx;

flex-shrink: 0;

}

.product-info {

flex: 1;

}

.product-name {

font-size: 30rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 12rpx;

line-height: 1.3;

}

.product-desc {

font-size: 26rpx;

color: var(--text-color-secondary);

margin-bottom: 16rpx;

line-height: 1.4;

}

.product-price {

display: flex;

justify-content: space-between;

align-items: center;

}

.price-range {

font-size: 28rpx;

color: var(--primary-color);

font-weight: 600;

}

.price-unit {

font-size: 24rpx;

color: var(--text-color-secondary);

}

/\* 客户经理卡片 \*/

.manager-card .card-body {

padding: 30rpx;

}

.manager-info {

display: flex;

align-items: center;

margin-bottom: 30rpx;

}

.manager-avatar {

width: 100rpx;

height: 100rpx;

border-radius: 50%;

background-color: var(--bg-color-light);

margin-right: 24rpx;

display: flex;

align-items: center;

justify-content: center;

flex-shrink: 0;

}

.manager-avatar .iconfont {

font-size: 40rpx;

color: var(--text-color-disabled);

}

.manager-details {

flex: 1;

}

.manager-name {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 8rpx;

}

.manager-title {

font-size: 26rpx;

color: var(--text-color-secondary);

margin-bottom: 8rpx;

}

.manager-phone {

font-size: 26rpx;

color: var(--primary-color);

}

.manager-actions {

display: flex;

gap: 20rpx;

}

.action-btn {

flex: 1;

height: 72rpx;

border-radius: 12rpx;

display: flex;

align-items: center;

justify-content: center;

font-size: 28rpx;

border: none;

transition: all 0.3s ease;

}

.action-btn.primary {

background-color: var(--primary-color);

color: #fff;

}

.action-btn.secondary {

background-color: var(--bg-color-light);

color: var(--text-color-primary);

border: 1rpx solid var(--border-color);

}

.action-btn:active {

transform: scale(0.95);

}

.action-btn .iconfont {

font-size: 24rpx;

margin-right: 8rpx;

}

/\* 合同卡片 \*/

.contract-card .card-body {

padding: 30rpx;

}

.contract-item {

display: flex;

align-items: center;

padding: 24rpx 0;

border-bottom: 1rpx solid var(--border-color-light);

}

.contract-item:last-child {

border-bottom: none;

}

.contract-icon {

width: 80rpx;

height: 80rpx;

border-radius: 12rpx;

background-color: var(--bg-color-light);

display: flex;

align-items: center;

justify-content: center;

margin-right: 20rpx;

flex-shrink: 0;

}

.contract-icon .iconfont {

font-size: 32rpx;

color: var(--primary-color);

}

.contract-info {

flex: 1;

}

.contract-name {

font-size: 30rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 8rpx;

}

.contract-status {

font-size: 26rpx;

color: var(--text-color-secondary);

margin-bottom: 8rpx;

}

.contract-date {

font-size: 24rpx;

color: var(--text-color-disabled);

}

.contract-actions {

display: flex;

flex-direction: column;

gap: 12rpx;

}

.contract-btn {

padding: 12rpx 24rpx;

border-radius: 8rpx;

font-size: 24rpx;

border: none;

text-align: center;

transition: all 0.3s ease;

}

.contract-btn.view {

background-color: var(--primary-color);

color: #fff;

}

.contract-btn.download {

background-color: var(--bg-color-light);

color: var(--text-color-primary);

border: 1rpx solid var(--border-color);

}

.contract-btn:active {

transform: scale(0.95);

}

/\* 跟进记录卡片 \*/

.follow-card .card-body {

padding: 0;

}

.follow-item {

padding: 30rpx;

border-bottom: 1rpx solid var(--border-color-light);

position: relative;

}

.follow-item:last-child {

border-bottom: none;

}

.follow-item::before {

content: '';

position: absolute;

left: 30rpx;

top: 60rpx;

width: 12rpx;

height: 12rpx;

border-radius: 50%;

background-color: var(--primary-color);

}

.follow-item::after {

content: '';

position: absolute;

left: 35rpx;

top: 80rpx;

width: 2rpx;

height: calc(100% - 50rpx);

background-color: var(--border-color);

}

.follow-item:last-child::after {

display: none;

}

.follow-content {

margin-left: 40rpx;

}

.follow-header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 16rpx;

}

.follow-type {

font-size: 28rpx;

font-weight: 600;

color: var(--text-color-primary);

}

.follow-time {

font-size: 24rpx;

color: var(--text-color-disabled);

}

.follow-text {

font-size: 26rpx;

color: var(--text-color-secondary);

line-height: 1.5;

margin-bottom: 16rpx;

}

.follow-operator {

font-size: 24rpx;

color: var(--text-color-disabled);

}

.follow-images {

display: flex;

flex-wrap: wrap;

gap: 12rpx;

margin-top: 16rpx;

}

.follow-image {

width: 120rpx;

height: 120rpx;

border-radius: 8rpx;

background-color: var(--bg-color-light);

}

/\* 底部操作栏 \*/

.bottom-actions {

position: fixed;

bottom: 0;

left: 0;

right: 0;

background-color: #fff;

padding: 24rpx 30rpx;

padding-bottom: calc(24rpx + env(safe-area-inset-bottom));

box-shadow: 0 -4rpx 12rpx rgba(0, 0, 0, 0.1);

z-index: 100;

}

.action-buttons {

display: flex;

gap: 20rpx;

}

.action-button {

flex: 1;

height: 88rpx;

border-radius: 16rpx;

display: flex;

align-items: center;

justify-content: center;

font-size: 32rpx;

font-weight: 600;

border: none;

transition: all 0.3s ease;

}

.action-button.secondary {

background-color: var(--bg-color-light);

color: var(--text-color-primary);

border: 1rpx solid var(--border-color);

}

.action-button.primary {

background-color: var(--primary-color);

color: #fff;

}

.action-button.danger {

background-color: var(--error-color);

color: #fff;

}

.action-button:active {

transform: scale(0.98);

}

.action-button:disabled {

opacity: 0.5;

pointer-events: none;

}

/\* 空状态 \*/

.empty-state {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 80rpx 40rpx;

text-align: center;

}

.empty-icon {

width: 120rpx;

height: 120rpx;

margin-bottom: 24rpx;

opacity: 0.3;

}

.empty-text {

font-size: 28rpx;

color: var(--text-color-secondary);

}

/\* 状态标签 \*/

.status-tag {

display: inline-block;

padding: 8rpx 16rpx;

border-radius: 8rpx;

font-size: 24rpx;

font-weight: 500;

}

.status-tag.pending {

background-color: rgba(250, 173, 20, 0.1);

color: #FAAD14;

}

.status-tag.processing {

background-color: rgba(24, 144, 255, 0.1);

color: #1890FF;

}

.status-tag.completed {

background-color: rgba(82, 196, 26, 0.1);

color: #52C41A;

}

.status-tag.cancelled {

background-color: rgba(255, 77, 79, 0.1);

color: #FF4D4F;

}

/\* 响应式调整 \*/

@media (max-width: 750rpx) {

.progress-steps {

margin: 0 -10rpx;

}

.progress-step {

margin: 0 10rpx;

}

.step-name {

font-size: 20rpx;

}

.manager-actions {

flex-direction: column;

}

.contract-actions {

flex-direction: row;

gap: 8rpx;

}

.contract-btn {

flex: 1;

padding: 8rpx 16rpx;

font-size: 22rpx;

}

}

现在继续开发下一个模块：**订单详情页面配置 - pages/orders/detail/detail.json**

订单详情页面配置 - detail.json

{

"navigationBarTitleText": "订单详情",

"navigationBarBackgroundColor": "#1890FF",

"navigationBarTextStyle": "white",

"backgroundColor": "#F5F7FA",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"disableScroll": false,

"usingComponents": {

"loading": "/components/loading/loading",

"empty": "/components/empty/empty",

"image-viewer": "/components/image-viewer/image-viewer"

},

"style": "v2"

}

接下来开发：**用户中心模块小程序页面 - 用户中心首页逻辑 - pages/profile/index/index.js**

用户中心首页逻辑 - pages/profile/index/index.js

// pages/profile/index/index.js

const app = getApp()

const api = require('../../../utils/api')

const auth = require('../../../utils/auth')

Page({

data: {

userInfo: null,

stats: {

totalOrders: 0,

totalAmount: 0,

totalSavings: 0,

carbonReduction: 0

},

menuItems: [

{

id: 'orders',

icon: 'icon-order',

title: '我的订单',

subtitle: '查看订单状态',

url: '/pages/orders/index/index',

badge: 0

},

{

id: 'contracts',

icon: 'icon-contract',

title: '我的合同',

subtitle: '合同管理',

url: '/pages/profile/contracts/contracts',

badge: 0

},

{

id: 'consumption',

icon: 'icon-chart',

title: '用电数据',

subtitle: '用电统计分析',

url: '/pages/profile/consumption/consumption'

},

{

id: 'savings',

icon: 'icon-money',

title: '节费记录',

subtitle: '查看节省费用',

url: '/pages/profile/savings/savings'

},

{

id: 'notifications',

icon: 'icon-message',

title: '消息通知',

subtitle: '系统通知消息',

url: '/pages/profile/notifications/notifications',

badge: 0

},

{

id: 'settings',

icon: 'icon-setting',

title: '设置',

subtitle: '个人设置',

url: '/pages/profile/settings/settings'

}

],

quickActions: [

{

id: 'contact-manager',

icon: 'icon-phone',

title: '联系客户经理',

color: '#1890FF'

},

{

id: 'online-service',

icon: 'icon-service',

title: '在线客服',

color: '#52C41A'

},

{

id: 'feedback',

icon: 'icon-feedback',

title: '意见反馈',

color: '#FAAD14'

},

{

id: 'help',

icon: 'icon-help',

title: '帮助中心',

color: '#722ED1'

}

],

showAuthDialog: false,

loading: true,

refreshing: false

},

onLoad() {

this.checkAuth()

},

onShow() {

// 每次显示页面时刷新数据

if (this.data.userInfo) {

this.loadUserData()

this.loadStats()

this.loadNotificationCount()

}

},

onPullDownRefresh() {

this.setData({ refreshing: true })

this.loadUserData().finally(() => {

this.setData({ refreshing: false })

wx.stopPullDownRefresh()

})

},

// 检查登录状态

async checkAuth() {

try {

const isLoggedIn = await auth.checkAuth()

if (isLoggedIn) {

await this.loadUserData()

await this.loadStats()

await this.loadNotificationCount()

} else {

this.setData({

showAuthDialog: true,

loading: false

})

}

} catch (error) {

console.error('检查登录状态失败:', error)

this.setData({

showAuthDialog: true,

loading: false

})

}

},

// 加载用户数据

async loadUserData() {

try {

this.setData({ loading: true })

const userInfo = await api.getCurrentUser()

this.setData({

userInfo,

loading: false

})

// 更新全局用户信息

app.globalData.userInfo = userInfo

} catch (error) {

console.error('加载用户信息失败:', error)

this.setData({ loading: false })

if (error.code === 401) {

this.setData({ showAuthDialog: true })

} else {

wx.showToast({

title: '加载失败，请重试',

icon: 'none'

})

}

}

},

// 加载统计数据

async loadStats() {

try {

const stats = await api.getUserStats()

this.setData({ stats })

} catch (error) {

console.error('加载统计数据失败:', error)

}

},

// 加载通知数量

async loadNotificationCount() {

try {

const result = await api.getUnreadNotificationCount()

const menuItems = this.data.menuItems.map(item => {

if (item.id === 'notifications') {

return { ...item, badge: result.count || 0 }

}

return item

})

this.setData({ menuItems })

} catch (error) {

console.error('加载通知数量失败:', error)

}

},

// 处理头像点击

onAvatarTap() {

if (!this.data.userInfo) {

this.setData({ showAuthDialog: true })

return

}

wx.navigateTo({

url: '/pages/profile/edit/edit'

})

},

// 处理菜单点击

onMenuTap(e) {

const { item } = e.currentTarget.dataset

if (!this.data.userInfo) {

this.setData({ showAuthDialog: true })

return

}

// 清除对应菜单的徽章

if (item.badge > 0) {

const menuItems = this.data.menuItems.map(menu => {

if (menu.id === item.id) {

return { ...menu, badge: 0 }

}

return menu

})

this.setData({ menuItems })

}

wx.navigateTo({

url: item.url

})

},

// 处理快捷操作

onQuickActionTap(e) {

const { action } = e.currentTarget.dataset

switch (action.id) {

case 'contact-manager':

this.contactManager()

break

case 'online-service':

this.openOnlineService()

break

case 'feedback':

this.openFeedback()

break

case 'help':

this.openHelp()

break

}

},

// 联系客户经理

async contactManager() {

if (!this.data.userInfo) {

this.setData({ showAuthDialog: true })

return

}

try {

const managerInfo = await api.getCustomerManager()

if (managerInfo && managerInfo.phone) {

wx.showActionSheet({

itemList: ['拨打电话', '发送短信'],

success: (res) => {

if (res.tapIndex === 0) {

wx.makePhoneCall({

phoneNumber: managerInfo.phone

})

} else if (res.tapIndex === 1) {

// 发送短信（小程序无法直接发送短信，可以复制号码）

wx.setClipboardData({

data: managerInfo.phone,

success: () => {

wx.showToast({

title: '号码已复制',

icon: 'success'

})

}

})

}

}

})

} else {

wx.showModal({

title: '提示',

content: '暂未分配客户经理，请联系在线客服',

showCancel: false

})

}

} catch (error) {

console.error('获取客户经理信息失败:', error)

wx.showToast({

title: '获取信息失败',

icon: 'none'

})

}

},

// 打开在线客服

openOnlineService() {

if (!this.data.userInfo) {

this.setData({ showAuthDialog: true })

return

}

wx.navigateTo({

url: '/pages/service/chat/chat'

})

},

// 打开意见反馈

openFeedback() {

if (!this.data.userInfo) {

this.setData({ showAuthDialog: true })

return

}

wx.navigateTo({

url: '/pages/profile/feedback/feedback'

})

},

// 打开帮助中心

openHelp() {

wx.navigateTo({

url: '/pages/help/index/index'

})

},

// 处理认证对话框

onAuthConfirm() {

this.setData({ showAuthDialog: false })

wx.navigateTo({

url: '/pages/auth/login/login'

})

},

onAuthCancel() {

this.setData({ showAuthDialog: false })

},

// 处理统计卡片点击

onStatsTap(e) {

const { type } = e.currentTarget.dataset

if (!this.data.userInfo) {

this.setData({ showAuthDialog: true })

return

}

switch (type) {

case 'orders':

wx.navigateTo({

url: '/pages/orders/index/index'

})

break

case 'amount':

case 'savings':

wx.navigateTo({

url: '/pages/profile/savings/savings'

})

break

case 'carbon':

wx.navigateTo({

url: '/pages/profile/consumption/consumption'

})

break

}

},

// 格式化金额显示

formatAmount(amount) {

if (amount >= 10000) {

return (amount / 10000).toFixed(1) + '万'

}

return amount.toFixed(2)

},

// 格式化数字显示

formatNumber(num) {

if (num >= 10000) {

return (num / 10000).toFixed(1) + '万'

}

return num.toString()

},

// 分享功能

onShareAppMessage() {

return {

title: '电力渠道销售平台 - 专业的电力服务',

path: '/pages/index/index',

imageUrl: '/images/share-bg.png'

}

},

onShareTimeline() {

return {

title: '电力渠道销售平台 - 专业的电力服务',

query: '',

imageUrl: '/images/share-bg.png'

}

}

})

继续开发：**用户中心首页模板 - pages/profile/index/index.wxml**

用户中心首页模板 - pages/profile/index/index.wxml

<!--pages/profile/index/index.wxml-->

<view class="profile-container">

<!-- 加载状态 -->

<loading wx:if="{{loading}}" text="加载中..." />

<!-- 用户信息区域 -->

<view class="user-section" wx:if="{{!loading}}">

<view class="user-header">

<view class="user-info" bindtap="onAvatarTap">

<view class="avatar-container">

<image

class="user-avatar"

src="{{userInfo ? userInfo.avatar\_url : '/images/default-avatar.png'}}"

mode="aspectFill"

/>

<view class="edit-icon">

<text class="iconfont icon-edit"></text>

</view>

</view>

<view class="user-details">

<view class="user-name">

{{userInfo ? userInfo.name : '未登录'}}

</view>

<view class="user-company" wx:if="{{userInfo && userInfo.company\_name}}">

{{userInfo.company\_name}}

</view>

<view class="login-tip" wx:else>

点击登录，享受更多服务

</view>

</view>

</view>

<!-- 认证状态 -->

<view class="auth-status" wx:if="{{userInfo}}">

<view class="status-item {{userInfo.auth\_status === 'verified' ? 'verified' : 'pending'}}">

<text class="iconfont {{userInfo.auth\_status === 'verified' ? 'icon-check-circle' : 'icon-time-circle'}}"></text>

<text class="status-text">

{{userInfo.auth\_status === 'verified' ? '已认证' : '待认证'}}

</text>

</view>

</view>

</view>

<!-- 数据统计 -->

<view class="stats-section" wx:if="{{userInfo}}">

<view class="stats-grid">

<view class="stats-item" data-type="orders" bindtap="onStatsTap">

<view class="stats-number">{{formatNumber(stats.totalOrders)}}</view>

<view class="stats-label">订单数量</view>

</view>

<view class="stats-item" data-type="amount" bindtap="onStatsTap">

<view class="stats-number">{{formatAmount(stats.totalAmount)}}</view>

<view class="stats-label">累计金额</view>

</view>

<view class="stats-item" data-type="savings" bindtap="onStatsTap">

<view class="stats-number">{{formatAmount(stats.totalSavings)}}</view>

<view class="stats-label">节省金额</view>

</view>

<view class="stats-item" data-type="contracts" bindtap="onStatsTap">

<view class="stats-number">{{formatNumber(stats.totalContracts)}}</view>

<view class="stats-label">有效合同</view>

</view>

</view>

</view>

</view>

<!-- 功能菜单 -->

<view class="menu-section" wx:if="{{!loading}}">

<!-- 订单相关 -->

<view class="menu-group">

<view class="group-title">订单管理</view>

<view class="menu-grid">

<view class="menu-item" data-url="/pages/orders/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-order"></text>

</view>

<view class="menu-text">我的订单</view>

<view class="menu-badge" wx:if="{{stats.pendingOrders > 0}}">

{{formatBadge(stats.pendingOrders)}}

</view>

</view>

<view class="menu-item" data-url="/pages/contracts/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-contract"></text>

</view>

<view class="menu-text">我的合同</view>

<view class="menu-badge" wx:if="{{stats.expiringSoonContracts > 0}}">

{{formatBadge(stats.expiringSoonContracts)}}

</view>

</view>

<view class="menu-item" data-url="/pages/power-data/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-chart"></text>

</view>

<view class="menu-text">用电数据</view>

</view>

<view class="menu-item" data-url="/pages/calculator/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-calculator"></text>

</view>

<view class="menu-text">节电计算</view>

</view>

</view>

</view>

<!-- 客户经理功能 (仅客户经理角色显示) -->

<view class="menu-group" wx:if="{{userInfo && userInfo.role === 'customer\_manager'}}">

<view class="group-title">客户经理</view>

<view class="menu-grid">

<view class="menu-item" data-url="/pages/customers/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-customers"></text>

</view>

<view class="menu-text">我的客户</view>

<view class="menu-badge" wx:if="{{stats.myCustomers > 0}}">

{{formatBadge(stats.myCustomers)}}

</view>

</view>

<view class="menu-item" data-url="/pages/follow-up/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-follow"></text>

</view>

<view class="menu-text">跟进管理</view>

<view class="menu-badge" wx:if="{{stats.todayFollowUp > 0}}">

{{formatBadge(stats.todayFollowUp)}}

</view>

</view>

<view class="menu-item" data-url="/pages/performance/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-performance"></text>

</view>

<view class="menu-text">我的业绩</view>

</view>

<view class="menu-item" data-url="/pages/ranking/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-ranking"></text>

</view>

<view class="menu-text">业绩排行</view>

</view>

</view>

</view>

<!-- 服务功能 -->

<view class="menu-group">

<view class="group-title">服务中心</view>

<view class="menu-grid">

<view class="menu-item" data-action="customer-service" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-service"></text>

</view>

<view class="menu-text">在线客服</view>

</view>

<view class="menu-item" data-url="/pages/feedback/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-feedback"></text>

</view>

<view class="menu-text">意见反馈</view>

</view>

<view class="menu-item" data-url="/pages/help/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-help"></text>

</view>

<view class="menu-text">帮助中心</view>

</view>

<view class="menu-item" data-url="/pages/about/index/index" bindtap="onMenuTap">

<view class="menu-icon">

<text class="iconfont icon-about"></text>

</view>

<view class="menu-text">关于我们</view>

</view>

</view>

</view>

<!-- 设置功能 -->

<view class="menu-group">

<view class="group-title">设置</view>

<view class="menu-list">

<view class="menu-list-item" data-url="/pages/profile/edit/edit" bindtap="onMenuTap">

<view class="list-item-content">

<view class="item-icon">

<text class="iconfont icon-user-edit"></text>

</view>

<view class="item-text">个人信息</view>

</view>

<view class="item-arrow">

<text class="iconfont icon-arrow-right"></text>

</view>

</view>

<view class="menu-list-item" data-url="/pages/profile/company/company" bindtap="onMenuTap" wx:if="{{userInfo}}">

<view class="list-item-content">

<view class="item-icon">

<text class="iconfont icon-company"></text>

</view>

<view class="item-text">企业信息</view>

</view>

<view class="item-arrow">

<text class="iconfont icon-arrow-right"></text>

</view>

</view>

<view class="menu-list-item" data-url="/pages/auth/verify/verify" bindtap="onMenuTap" wx:if="{{userInfo && userInfo.auth\_status !== 'verified'}}">

<view class="list-item-content">

<view class="item-icon">

<text class="iconfont icon-verify"></text>

</view>

<view class="item-text">身份认证</view>

</view>

<view class="item-arrow">

<text class="iconfont icon-arrow-right"></text>

</view>

</view>

<view class="menu-list-item" data-url="/pages/settings/index/index" bindtap="onMenuTap">

<view class="list-item-content">

<view class="item-icon">

<text class="iconfont icon-settings"></text>

</view>

<view class="item-text">系统设置</view>

</view>

<view class="item-arrow">

<text class="iconfont icon-arrow-right"></text>

</view>

</view>

</view>

</view>

</view>

<!-- 退出登录按钮 -->

<view class="logout-section" wx:if="{{userInfo && !loading}}">

<button class="logout-btn" bindtap="onLogout">退出登录</button>

</view>

<!-- 版本信息 -->

<view class="version-info">

<text class="version-text">版本号：{{version}}</text>

</view>

</view>

<!-- 头像选择弹窗 -->

<modal

title="更换头像"

hidden="{{!showAvatarModal}}"

confirm-text="确定"

cancel-text="取消"

bindconfirm="onAvatarConfirm"

bindcancel="onAvatarCancel"

>

<view class="avatar-options">

<view class="avatar-option" bindtap="onChooseFromAlbum">

<text class="iconfont icon-album"></text>

<text>从相册选择</text>

</view>

<view class="avatar-option" bindtap="onTakePhoto">

<text class="iconfont icon-camera"></text>

<text>拍照</text>

</view>

</view>

</modal>

现在继续开发下一个模块：**用户中心首页样式文件 - pages/profile/index/index.wxss**

用户中心首页样式文件 - pages/profile/index/index.wxss

/\* pages/profile/index/index.wxss \*/

@import '../../../styles/common.wxss';

.profile-container {

min-height: 100vh;

background-color: var(--bg-color-light);

padding-bottom: 40rpx;

}

/\* 用户信息区域 \*/

.user-section {

background: linear-gradient(135deg, var(--primary-color) 0%, var(--primary-dark) 100%);

padding: 60rpx 30rpx 40rpx;

color: #fff;

position: relative;

overflow: hidden;

}

.user-section::before {

content: '';

position: absolute;

top: -50%;

right: -20%;

width: 200%;

height: 200%;

background: radial-gradient(circle, rgba(255,255,255,0.1) 0%, transparent 70%);

pointer-events: none;

}

.user-header {

position: relative;

z-index: 2;

}

.user-info {

display: flex;

align-items: center;

margin-bottom: 40rpx;

}

.avatar-container {

position: relative;

margin-right: 30rpx;

}

.user-avatar {

width: 120rpx;

height: 120rpx;

border-radius: 50%;

border: 4rpx solid rgba(255,255,255,0.3);

box-shadow: 0 4rpx 12rpx rgba(0,0,0,0.1);

}

.edit-icon {

position: absolute;

bottom: 0;

right: 0;

width: 40rpx;

height: 40rpx;

background-color: var(--warning-color);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

border: 3rpx solid #fff;

font-size: 20rpx;

color: #fff;

}

.user-details {

flex: 1;

}

.user-name {

font-size: 36rpx;

font-weight: 600;

margin-bottom: 10rpx;

text-shadow: 0 2rpx 4rpx rgba(0,0,0,0.1);

}

.user-company {

font-size: 28rpx;

opacity: 0.9;

margin-bottom: 8rpx;

}

.login-tip {

font-size: 26rpx;

opacity: 0.8;

}

.auth-status {

display: flex;

justify-content: flex-end;

}

.status-item {

display: flex;

align-items: center;

padding: 12rpx 20rpx;

border-radius: 30rpx;

font-size: 24rpx;

background: rgba(255,255,255,0.2);

backdrop-filter: blur(10rpx);

}

.status-item.verified {

background: rgba(76, 175, 80, 0.3);

}

.status-item.pending {

background: rgba(255, 152, 0, 0.3);

}

.status-item .iconfont {

font-size: 28rpx;

margin-right: 8rpx;

}

/\* 数据统计 \*/

.stats-section {

margin-top: 20rpx;

}

.stats-grid {

display: grid;

grid-template-columns: repeat(4, 1fr);

gap: 20rpx;

}

.stats-item {

text-align: center;

padding: 20rpx 10rpx;

border-radius: 16rpx;

background: rgba(255,255,255,0.15);

backdrop-filter: blur(10rpx);

transition: all 0.3s ease;

}

.stats-item:active {

transform: scale(0.95);

background: rgba(255,255,255,0.25);

}

.stats-number {

font-size: 32rpx;

font-weight: 700;

margin-bottom: 8rpx;

text-shadow: 0 2rpx 4rpx rgba(0,0,0,0.1);

}

.stats-label {

font-size: 22rpx;

opacity: 0.9;

}

/\* 功能菜单 \*/

.menu-section {

padding: 30rpx;

}

.menu-group {

margin-bottom: 40rpx;

}

.group-title {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 20rpx;

padding-left: 10rpx;

position: relative;

}

.group-title::before {

content: '';

position: absolute;

left: 0;

top: 50%;

transform: translateY(-50%);

width: 6rpx;

height: 24rpx;

background: var(--primary-color);

border-radius: 3rpx;

}

/\* 网格菜单 \*/

.menu-grid {

display: grid;

grid-template-columns: repeat(4, 1fr);

gap: 30rpx;

background: #fff;

padding: 40rpx 30rpx;

border-radius: 20rpx;

box-shadow: 0 2rpx 12rpx rgba(0,0,0,0.08);

}

.menu-item {

display: flex;

flex-direction: column;

align-items: center;

position: relative;

padding: 20rpx 10rpx;

border-radius: 16rpx;

transition: all 0.3s ease;

}

.menu-item:active {

transform: scale(0.95);

background-color: var(--bg-color-light);

}

.menu-icon {

width: 80rpx;

height: 80rpx;

background: linear-gradient(135deg, var(--primary-color), var(--primary-light));

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

margin-bottom: 16rpx;

box-shadow: 0 4rpx 12rpx rgba(33, 150, 243, 0.3);

}

.menu-icon .iconfont {

font-size: 36rpx;

color: #fff;

}

.menu-text {

font-size: 24rpx;

color: var(--text-color-primary);

text-align: center;

line-height: 1.4;

}

.menu-badge {

position: absolute;

top: 15rpx;

right: 15rpx;

min-width: 32rpx;

height: 32rpx;

background: var(--error-color);

color: #fff;

font-size: 20rpx;

border-radius: 16rpx;

display: flex;

align-items: center;

justify-content: center;

padding: 0 8rpx;

font-weight: 600;

transform: scale(0.9);

}

/\* 列表菜单 \*/

.menu-list {

background: #fff;

border-radius: 20rpx;

overflow: hidden;

box-shadow: 0 2rpx 12rpx rgba(0,0,0,0.08);

}

.menu-list-item {

display: flex;

align-items: center;

justify-content: space-between;

padding: 32rpx 30rpx;

border-bottom: 1rpx solid var(--border-color);

transition: background-color 0.3s ease;

}

.menu-list-item:last-child {

border-bottom: none;

}

.menu-list-item:active {

background-color: var(--bg-color-light);

}

.list-item-content {

display: flex;

align-items: center;

flex: 1;

}

.item-icon {

width: 60rpx;

height: 60rpx;

background: var(--bg-color-light);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

margin-right: 24rpx;

}

.item-icon .iconfont {

font-size: 28rpx;

color: var(--primary-color);

}

.item-text {

font-size: 30rpx;

color: var(--text-color-primary);

}

.item-arrow .iconfont {

font-size: 24rpx;

color: var(--text-color-secondary);

}

/\* 退出登录按钮 \*/

.logout-section {

padding: 30rpx;

}

.logout-btn {

width: 100%;

background: #fff;

color: var(--error-color);

font-size: 32rpx;

font-weight: 500;

border: 2rpx solid var(--error-color);

border-radius: 16rpx;

padding: 24rpx;

transition: all 0.3s ease;

}

.logout-btn:active {

background: var(--error-color);

color: #fff;

}

/\* 版本信息 \*/

.version-info {

text-align: center;

padding: 20rpx;

}

.version-text {

font-size: 24rpx;

color: var(--text-color-secondary);

}

/\* 头像选择弹窗 \*/

.avatar-options {

display: flex;

justify-content: space-around;

padding: 40rpx 20rpx;

}

.avatar-option {

display: flex;

flex-direction: column;

align-items: center;

padding: 30rpx;

border-radius: 16rpx;

background: var(--bg-color-light);

transition: all 0.3s ease;

min-width: 120rpx;

}

.avatar-option:active {

transform: scale(0.95);

background: var(--primary-light);

}

.avatar-option .iconfont {

font-size: 48rpx;

color: var(--primary-color);

margin-bottom: 12rpx;

}

.avatar-option text:last-child {

font-size: 26rpx;

color: var(--text-color-primary);

}

/\* 响应式适配 \*/

@media screen and (max-width: 350px) {

.stats-grid {

grid-template-columns: repeat(2, 1fr);

gap: 15rpx;

}

.menu-grid {

grid-template-columns: repeat(3, 1fr);

gap: 20rpx;

padding: 30rpx 20rpx;

}

.user-info {

flex-direction: column;

text-align: center;

}

.avatar-container {

margin-right: 0;

margin-bottom: 20rpx;

}

}

/\* 深色模式适配 \*/

@media (prefers-color-scheme: dark) {

.profile-container {

background-color: var(--bg-color-dark);

}

.menu-grid,

.menu-list {

background: var(--card-bg-dark);

}

.logout-btn {

background: var(--card-bg-dark);

border-color: var(--error-color);

}

}

接下来开发：**用户中心首页配置文件 - pages/profile/index/index.json**

用户中心首页配置文件 - pages/profile/index/index.json

{

"navigationBarTitleText": "个人中心",

"navigationBarBackgroundColor": "#2196F3",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f5f5",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"usingComponents": {

"loading": "/components/loading/loading"

},

"style": "v2",

"singlePage": {

"navigationBarFit": "float"

}

}

现在开发下一个模块：**个人信息编辑页面逻辑 - pages/profile/edit/edit.js**

个人信息编辑页面逻辑 - pages/profile/edit/edit.js

// pages/profile/edit/edit.js

const api = require('../../../utils/api');

const auth = require('../../../utils/auth');

const utils = require('../../../utils/utils');

Page({

data: {

loading: false,

submitting: false,

userInfo: null,

// 表单数据

formData: {

name: '',

phone: '',

email: '',

company\_name: '',

industry\_type: '',

registered\_address: '',

contact\_person: '',

contact\_phone: ''

},

// 表单验证错误

errors: {},

// 行业类型选项

industryOptions: [

{ value: 'manufacturing', label: '制造业' },

{ value: 'commerce', label: '商业贸易' },

{ value: 'service', label: '服务业' },

{ value: 'real\_estate', label: '房地产' },

{ value: 'logistics', label: '物流运输' },

{ value: 'finance', label: '金融业' },

{ value: 'education', label: '教育培训' },

{ value: 'healthcare', label: '医疗健康' },

{ value: 'technology', label: '科技互联网' },

{ value: 'agriculture', label: '农林牧渔' },

{ value: 'mining', label: '采矿业' },

{ value: 'construction', label: '建筑业' },

{ value: 'other', label: '其他' }

],

// 选择器相关

industryIndex: -1,

showIndustryPicker: false,

// 企业搜索相关

companySearchResults: [],

showCompanySearch: false,

companySearchKeyword: '',

companySearchTimer: null

},

onLoad(options) {

this.loadUserInfo();

},

onShow() {

// 每次显示页面时检查登录状态

if (!auth.isLoggedIn()) {

wx.redirectTo({

url: '/pages/auth/login/login'

});

return;

}

},

onPullDownRefresh() {

this.loadUserInfo().finally(() => {

wx.stopPullDownRefresh();

});

},

// 加载用户信息

async loadUserInfo() {

if (!auth.isLoggedIn()) {

wx.redirectTo({

url: '/pages/auth/login/login'

});

return;

}

try {

this.setData({ loading: true });

const userInfo = await api.getUserInfo();

console.log('用户信息:', userInfo);

// 初始化表单数据

const formData = {

name: userInfo.name || '',

phone: userInfo.phone || '',

email: userInfo.email || '',

company\_name: userInfo.company\_name || '',

industry\_type: userInfo.industry\_type || '',

registered\_address: userInfo.registered\_address || '',

contact\_person: userInfo.contact\_person || '',

contact\_phone: userInfo.contact\_phone || ''

};

// 设置行业类型选择器索引

const industryIndex = this.data.industryOptions.findIndex(

option => option.value === userInfo.industry\_type

);

this.setData({

userInfo,

formData,

industryIndex: industryIndex >= 0 ? industryIndex : -1,

errors: {}

});

} catch (error) {

console.error('加载用户信息失败:', error);

wx.showToast({

title: '加载失败',

icon: 'error'

});

} finally {

this.setData({ loading: false });

}

},

// 表单输入处理

onInputChange(e) {

const { field } = e.currentTarget.dataset;

const { value } = e.detail;

this.setData({

[`formData.${field}`]: value,

[`errors.${field}`]: '' // 清除错误信息

});

// 企业名称搜索

if (field === 'company\_name') {

this.handleCompanySearch(value);

}

},

// 企业名称搜索

handleCompanySearch(keyword) {

// 清除之前的定时器

if (this.data.companySearchTimer) {

clearTimeout(this.data.companySearchTimer);

}

// 设置搜索关键词

this.setData({

companySearchKeyword: keyword,

showCompanySearch: keyword.length >= 2

});

// 如果关键词长度不够，直接返回

if (keyword.length < 2) {

this.setData({

companySearchResults: [],

showCompanySearch: false

});

return;

}

// 设置新的定时器，防抖搜索

const timer = setTimeout(() => {

this.searchCompanies(keyword);

}, 300);

this.setData({

companySearchTimer: timer

});

},

// 搜索企业

async searchCompanies(keyword) {

try {

const results = await api.searchCompanies(keyword);

console.log('企业搜索结果:', results);

this.setData({

companySearchResults: results || []

});

} catch (error) {

console.error('企业搜索失败:', error);

this.setData({

companySearchResults: []

});

}

},

// 选择企业

onCompanySelect(e) {

const { company } = e.currentTarget.dataset;

console.log('选择企业:', company);

// 自动填充企业相关信息

const updates = {

'formData.company\_name': company.company\_name,

'formData.registered\_address': company.registered\_address || '',

showCompanySearch: false,

companySearchResults: [],

'errors.company\_name': ''

};

// 如果有行业信息，也自动填充

if (company.industry\_type) {

const industryIndex = this.data.industryOptions.findIndex(

option => option.value === company.industry\_type

);

if (industryIndex >= 0) {

updates['industryIndex'] = industryIndex;

updates['formData.industry\_type'] = company.industry\_type;

}

}

this.setData(updates);

},

// 隐藏企业搜索结果

onHideCompanySearch() {

this.setData({

showCompanySearch: false,

companySearchResults: []

});

},

// 行业类型选择器

onIndustryChange(e) {

const index = e.detail.value;

const selectedOption = this.data.industryOptions[index];

this.setData({

industryIndex: index,

'formData.industry\_type': selectedOption.value,

'errors.industry\_type': ''

});

},

// 显示行业选择器

onShowIndustryPicker() {

this.setData({

showIndustryPicker: true

});

},

// 隐藏行业选择器

onHideIndustryPicker() {

this.setData({

showIndustryPicker: false

});

},

// 表单验证

validateForm() {

const { formData } = this.data;

const errors = {};

let isValid = true;

// 姓名验证

if (!formData.name || formData.name.trim().length === 0) {

errors.name = '请输入姓名';

isValid = false;

} else if (formData.name.length > 20) {

errors.name = '姓名长度不能超过20个字符';

isValid = false;

}

// 手机号验证

if (!formData.phone || formData.phone.trim().length === 0) {

errors.phone = '请输入手机号';

isValid = false;

} else if (!/^1[3-9]\d{9}$/.test(formData.phone)) {

errors.phone = '请输入正确的手机号';

isValid = false;

}

// 邮箱验证（可选）

if (formData.email && formData.email.trim().length > 0) {

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!emailRegex.test(formData.email)) {

errors.email = '请输入正确的邮箱地址';

isValid = false;

}

}

// 企业名称验证

if (!formData.company\_name || formData.company\_name.trim().length === 0) {

errors.company\_name = '请输入企业名称';

isValid = false;

} else if (formData.company\_name.length > 100) {

errors.company\_name = '企业名称长度不能超过100个字符';

isValid = false;

}

// 行业类型验证

if (!formData.industry\_type || formData.industry\_type.trim().length === 0) {

errors.industry\_type = '请选择行业类型';

isValid = false;

}

// 注册地址验证

if (!formData.registered\_address || formData.registered\_address.trim().length === 0) {

errors.registered\_address = '请输入注册地址';

isValid = false;

} else if (formData.registered\_address.length > 200) {

errors.registered\_address = '注册地址长度不能超过200个字符';

isValid = false;

}

// 联系人验证

if (!formData.contact\_person || formData.contact\_person.trim().length === 0) {

errors.contact\_person = '请输入联系人姓名';

isValid = false;

} else if (formData.contact\_person.length > 20) {

errors.contact\_person = '联系人姓名长度不能超过20个字符';

isValid = false;

}

// 联系电话验证

if (!formData.contact\_phone || formData.contact\_phone.trim().length === 0) {

errors.contact\_phone = '请输入联系电话';

isValid = false;

} else if (!/^1[3-9]\d{9}$/.test(formData.contact\_phone)) {

errors.contact\_phone = '请输入正确的联系电话';

isValid = false;

}

this.setData({ errors });

return isValid;

},

// 提交表单

async onSubmit() {

if (this.data.submitting) {

return;

}

// 表单验证

if (!this.validateForm()) {

wx.showToast({

title: '请检查输入信息',

icon: 'error'

});

return;

}

try {

this.setData({ submitting: true });

const updateData = { ...this.data.formData };

// 去除前后空格

Object.keys(updateData).forEach(key => {

if (typeof updateData[key] === 'string') {

updateData[key] = updateData[key].trim();

}

});

console.log('提交更新数据:', updateData);

const result = await api.updateUserInfo(updateData);

console.log('更新成功:', result);

wx.showToast({

title: '保存成功',

icon: 'success'

});

// 更新本地用户信息

this.setData({

userInfo: { ...this.data.userInfo, ...updateData }

});

// 延迟返回上一页

setTimeout(() => {

wx.navigateBack();

}, 1500);

} catch (error) {

console.error('更新用户信息失败:', error);

let errorMessage = '保存失败';

if (error.message) {

errorMessage = error.message;

} else if (error.data && error.data.message) {

errorMessage = error.data.message;

}

wx.showToast({

title: errorMessage,

icon: 'error',

duration: 2000

});

} finally {

this.setData({ submitting: false });

}

},

// 重置表单

onReset() {

wx.showModal({

title: '确认重置',

content: '确定要重置表单吗？所有修改将被清除',

success: (res) => {

if (res.confirm) {

this.loadUserInfo();

wx.showToast({

title: '已重置',

icon: 'success'

});

}

}

});

},

// 页面卸载时清理定时器

onUnload() {

if (this.data.companySearchTimer) {

clearTimeout(this.data.companySearchTimer);

}

},

// 页面隐藏时隐藏弹框

onHide() {

this.setData({

showCompanySearch: false,

showIndustryPicker: false

});

}

});

现在继续开发下一个模块：**个人信息编辑页面模板 - pages/profile/edit/edit.wxml**

个人信息编辑页面模板 - pages/profile/edit/edit.wxml

<!--pages/profile/edit/edit.wxml-->

<view class="container">

<!-- 加载状态 -->

<view wx:if="{{loading}}" class="loading-container">

<view class="loading-text">

<text class="loading-icon">⏳</text>

<text>加载中...</text>

</view>

</view>

<!-- 表单内容 -->

<view wx:else class="form-container">

<form bindsubmit="onSubmit">

<!-- 个人基本信息 -->

<view class="form-section">

<view class="section-title">个人基本信息</view>

<!-- 姓名 -->

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

<text>姓名</text>

</view>

<input

class="form-input {{errors.name ? 'error' : ''}}"

type="text"

placeholder="请输入姓名"

value="{{formData.name}}"

data-field="name"

bindinput="onInputChange"

maxlength="20"

/>

<view wx:if="{{errors.name}}" class="error-text">{{errors.name}}</view>

</view>

<!-- 手机号 -->

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

<text>手机号</text>

</view>

<input

class="form-input {{errors.phone ? 'error' : ''}}"

type="number"

placeholder="请输入手机号"

value="{{formData.phone}}"

data-field="phone"

bindinput="onInputChange"

maxlength="11"

/>

<view wx:if="{{errors.phone}}" class="error-text">{{errors.phone}}</view>

</view>

<!-- 邮箱 -->

<view class="form-item">

<view class="form-label">

<text>邮箱</text>

</view>

<input

class="form-input {{errors.email ? 'error' : ''}}"

type="text"

placeholder="请输入邮箱（可选）"

value="{{formData.email}}"

data-field="email"

bindinput="onInputChange"

/>

<view wx:if="{{errors.email}}" class="error-text">{{errors.email}}</view>

</view>

</view>

<!-- 企业信息 -->

<view class="form-section">

<view class="section-title">企业信息</view>

<!-- 企业名称 -->

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

<text>企业名称</text>

</view>

<view class="search-input-container">

<input

class="form-input {{errors.company\_name ? 'error' : ''}}"

type="text"

placeholder="请输入企业名称"

value="{{formData.company\_name}}"

data-field="company\_name"

bindinput="onInputChange"

maxlength="100"

/>

<!-- 搜索结果下拉框 -->

<view wx:if="{{showCompanySearch && companySearchResults.length > 0}}" class="search-results">

<view

wx:for="{{companySearchResults}}"

wx:key="id"

class="search-item"

data-company="{{item}}"

bindtap="onCompanySelect"

>

<view class="company-name">{{item.company\_name}}</view>

<view class="company-address">{{item.registered\_address}}</view>

</view>

</view>

<!-- 搜索结果为空 -->

<view wx:if="{{showCompanySearch && companySearchResults.length === 0 && companySearchKeyword.length >= 2}}" class="search-empty">

<text>未找到相关企业</text>

</view>

</view>

<view wx:if="{{errors.company\_name}}" class="error-text">{{errors.company\_name}}</view>

</view>

<!-- 行业类型 -->

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

<text>行业类型</text>

</view>

<view class="picker-container" bindtap="onShowIndustryPicker">

<input

class="form-input picker-input {{errors.industry\_type ? 'error' : ''}}"

type="text"

placeholder="请选择行业类型"

value="{{industryIndex >= 0 ? industryOptions[industryIndex].label : ''}}"

disabled

/>

<view class="picker-arrow">></view>

</view>

<view wx:if="{{errors.industry\_type}}" class="error-text">{{errors.industry\_type}}</view>

</view>

<!-- 注册地址 -->

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

<text>注册地址</text>

</view>

<textarea

class="form-textarea {{errors.registered\_address ? 'error' : ''}}"

placeholder="请输入注册地址"

value="{{formData.registered\_address}}"

data-field="registered\_address"

bindinput="onInputChange"

maxlength="200"

auto-height

/>

<view wx:if="{{errors.registered\_address}}" class="error-text">{{errors.registered\_address}}</view>

</view>

<!-- 联系人 -->

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

<text>联系人</text>

</view>

<input

class="form-input {{errors.contact\_person ? 'error' : ''}}"

type="text"

placeholder="请输入联系人姓名"

value="{{formData.contact\_person}}"

data-field="contact\_person"

bindinput="onInputChange"

maxlength="20"

/>

<view wx:if="{{errors.contact\_person}}" class="error-text">{{errors.contact\_person}}</view>

</view>

<!-- 联系电话 -->

<view class="form-item">

<view class="form-label">

<text class="required">\*</text>

<text>联系电话</text>

</view>

<input

class="form-input {{errors.contact\_phone ? 'error' : ''}}"

type="number"

placeholder="请输入联系电话"

value="{{formData.contact\_phone}}"

data-field="contact\_phone"

bindinput="onInputChange"

maxlength="11"

/>

<view wx:if="{{errors.contact\_phone}}" class="error-text">{{errors.contact\_phone}}</view>

</view>

</view>

<!-- 操作按钮 -->

<view class="form-actions">

<button

class="btn-reset"

type="button"

bindtap="onReset"

disabled="{{submitting}}"

>

重置

</button>

<button

class="btn-submit"

form-type="submit"

disabled="{{submitting}}"

loading="{{submitting}}"

>

{{submitting ? '保存中...' : '保存'}}

</button>

</view>

</form>

</view>

<!-- 行业类型选择器 -->

<view wx:if="{{showIndustryPicker}}" class="picker-overlay" bindtap="onHideIndustryPicker">

<view class="picker-container-popup" catchtap="">

<view class="picker-header">

<text class="picker-cancel" bindtap="onHideIndustryPicker">取消</text>

<text class="picker-title">选择行业类型</text>

<text class="picker-confirm" bindtap="onHideIndustryPicker">确定</text>

</view>

<picker-view

class="picker-view"

value="{{[industryIndex >= 0 ? industryIndex : 0]}}"

bindchange="onIndustryChange"

>

<picker-view-column>

<view wx:for="{{industryOptions}}" wx:key="value" class="picker-item">

{{item.label}}

</view>

</picker-view-column>

</picker-view>

</view>

</view>

<!-- 遮罩层 - 点击隐藏企业搜索结果 -->

<view

wx:if="{{showCompanySearch}}"

class="search-overlay"

bindtap="onHideCompanySearch"

></view>

</view>

现在继续开发下一个模块：**个人信息编辑页面样式文件 - pages/profile/edit/edit.wxss**

个人信息编辑页面样式文件 - pages/profile/edit/edit.wxss

/\* pages/profile/edit/edit.wxss \*/

.container {

min-height: 100vh;

background-color: var(--bg-color-light);

}

/\* 加载状态 \*/

.loading-container {

display: flex;

justify-content: center;

align-items: center;

height: 400rpx;

}

.loading-text {

display: flex;

flex-direction: column;

align-items: center;

color: var(--text-color-secondary);

font-size: 28rpx;

}

.loading-icon {

font-size: 48rpx;

margin-bottom: 16rpx;

}

/\* 表单容器 \*/

.form-container {

padding: 32rpx;

padding-bottom: 120rpx;

}

/\* 表单区块 \*/

.form-section {

background-color: #fff;

border-radius: 16rpx;

padding: 32rpx;

margin-bottom: 24rpx;

box-shadow: 0 2rpx 8rpx rgba(0, 0, 0, 0.06);

}

.section-title {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: 32rpx;

padding-bottom: 16rpx;

border-bottom: 2rpx solid var(--border-color);

position: relative;

}

.section-title::before {

content: '';

position: absolute;

left: 0;

bottom: -2rpx;

width: 60rpx;

height: 4rpx;

background: var(--primary-color);

border-radius: 2rpx;

}

/\* 表单项 \*/

.form-item {

margin-bottom: 32rpx;

}

.form-item:last-child {

margin-bottom: 0;

}

.form-label {

display: flex;

align-items: center;

font-size: 28rpx;

color: var(--text-color-primary);

margin-bottom: 16rpx;

font-weight: 500;

}

.required {

color: var(--error-color);

margin-right: 8rpx;

font-size: 28rpx;

}

/\* 输入框样式 \*/

.form-input {

width: 100%;

height: 80rpx;

padding: 0 24rpx;

border: 2rpx solid var(--border-color);

border-radius: 12rpx;

font-size: 28rpx;

color: var(--text-color-primary);

background-color: #fff;

transition: all 0.3s ease;

}

.form-input:focus {

border-color: var(--primary-color);

box-shadow: 0 0 0 4rpx rgba(79, 172, 254, 0.1);

}

.form-input.error {

border-color: var(--error-color);

box-shadow: 0 0 0 4rpx rgba(245, 108, 108, 0.1);

}

.form-input::placeholder {

color: var(--text-color-placeholder);

}

/\* 文本域样式 \*/

.form-textarea {

width: 100%;

min-height: 120rpx;

padding: 16rpx 24rpx;

border: 2rpx solid var(--border-color);

border-radius: 12rpx;

font-size: 28rpx;

color: var(--text-color-primary);

background-color: #fff;

transition: all 0.3s ease;

line-height: 1.6;

}

.form-textarea:focus {

border-color: var(--primary-color);

box-shadow: 0 0 0 4rpx rgba(79, 172, 254, 0.1);

}

.form-textarea.error {

border-color: var(--error-color);

box-shadow: 0 0 0 4rpx rgba(245, 108, 108, 0.1);

}

.form-textarea::placeholder {

color: var(--text-color-placeholder);

}

/\* 选择器样式 \*/

.picker-container {

position: relative;

}

.picker-input {

cursor: pointer;

user-select: none;

}

.picker-arrow {

position: absolute;

right: 24rpx;

top: 50%;

transform: translateY(-50%) rotate(90deg);

color: var(--text-color-secondary);

font-size: 28rpx;

transition: transform 0.3s ease;

}

/\* 搜索输入框容器 \*/

.search-input-container {

position: relative;

}

/\* 搜索结果样式 \*/

.search-results {

position: absolute;

top: 100%;

left: 0;

right: 0;

background-color: #fff;

border: 2rpx solid var(--border-color);

border-top: none;

border-radius: 0 0 12rpx 12rpx;

max-height: 400rpx;

overflow-y: auto;

z-index: 10;

box-shadow: 0 4rpx 12rpx rgba(0, 0, 0, 0.1);

}

.search-item {

padding: 24rpx;

border-bottom: 1rpx solid var(--border-color);

transition: background-color 0.3s ease;

}

.search-item:last-child {

border-bottom: none;

}

.search-item:hover {

background-color: var(--bg-color-light);

}

.company-name {

font-size: 28rpx;

color: var(--text-color-primary);

font-weight: 500;

margin-bottom: 8rpx;

}

.company-address {

font-size: 24rpx;

color: var(--text-color-secondary);

line-height: 1.4;

}

.search-empty {

padding: 40rpx 24rpx;

text-align: center;

color: var(--text-color-secondary);

font-size: 26rpx;

}

/\* 搜索遮罩层 \*/

.search-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

z-index: 5;

}

/\* 错误提示 \*/

.error-text {

color: var(--error-color);

font-size: 24rpx;

margin-top: 8rpx;

line-height: 1.4;

}

/\* 操作按钮 \*/

.form-actions {

position: fixed;

bottom: 0;

left: 0;

right: 0;

display: flex;

padding: 24rpx 32rpx;

background-color: #fff;

border-top: 1rpx solid var(--border-color);

box-shadow: 0 -4rpx 12rpx rgba(0, 0, 0, 0.06);

gap: 24rpx;

}

.btn-reset {

flex: 1;

height: 88rpx;

border: 2rpx solid var(--border-color);

background-color: #fff;

color: var(--text-color-primary);

font-size: 28rpx;

border-radius: 12rpx;

transition: all 0.3s ease;

}

.btn-reset:hover {

border-color: var(--text-color-secondary);

background-color: var(--bg-color-light);

}

.btn-reset[disabled] {

opacity: 0.6;

pointer-events: none;

}

.btn-submit {

flex: 2;

height: 88rpx;

background: linear-gradient(135deg, var(--primary-color), var(--primary-color-dark));

color: #fff;

font-size: 28rpx;

font-weight: 500;

border: none;

border-radius: 12rpx;

transition: all 0.3s ease;

}

.btn-submit:hover {

transform: translateY(-2rpx);

box-shadow: 0 8rpx 20rpx rgba(79, 172, 254, 0.3);

}

.btn-submit[disabled] {

opacity: 0.6;

transform: none;

box-shadow: none;

pointer-events: none;

}

/\* 行业选择器弹窗 \*/

.picker-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0, 0, 0, 0.5);

display: flex;

align-items: flex-end;

z-index: 1000;

}

.picker-container-popup {

width: 100%;

background-color: #fff;

border-radius: 24rpx 24rpx 0 0;

animation: slideUp 0.3s ease-out;

}

@keyframes slideUp {

from {

transform: translateY(100%);

}

to {

transform: translateY(0);

}

}

.picker-header {

display: flex;

align-items: center;

justify-content: space-between;

padding: 32rpx;

border-bottom: 1rpx solid var(--border-color);

}

.picker-cancel,

.picker-confirm {

font-size: 28rpx;

color: var(--primary-color);

padding: 8rpx;

}

.picker-title {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

}

.picker-view {

height: 400rpx;

width: 100%;

}

.picker-item {

display: flex;

align-items: center;

justify-content: center;

height: 80rpx;

font-size: 28rpx;

color: var(--text-color-primary);

}

现在继续开发下一个模块：**个人信息编辑页面配置文件 - pages/profile/edit/edit.json**

个人信

{

"navigationBarTitleText": "编辑个人信息",

"navigationBarBackgroundColor": "#4facfe",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f7fa",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"usingComponents": {}

}

现在继续开发下一个模块：**我的合同页面逻辑 - pages/profile/contracts/contracts.js**

我的合同页面逻辑 - pages/profile/contracts/contracts.js

// pages/profile/contracts/contracts.js

const app = getApp()

const api = require('../../../utils/api')

const util = require('../../../utils/util')

Page({

data: {

contracts: [],

loading: true,

refreshing: false,

hasMore: true,

page: 1,

pageSize: 10,

// 筛选条件

filterVisible: false,

statusOptions: [

{ label: '全部', value: '' },

{ label: '待签署', value: 'pending' },

{ label: '已签署', value: 'signed' },

{ label: '执行中', value: 'executing' },

{ label: '已完成', value: 'completed' },

{ label: '已取消', value: 'cancelled' }

],

selectedStatus: '',

// 日期筛选

dateRange: {

start: '',

end: ''

},

// 搜索

searchKeyword: '',

// 预览

previewContract: null,

previewVisible: false,

// 统计信息

statistics: {

total: 0,

signed: 0,

executing: 0,

nearExpiry: 0

}

},

onLoad(options) {

this.loadContracts()

this.loadStatistics()

},

onShow() {

// 页面显示时刷新数据

if (this.data.contracts.length > 0) {

this.refreshContracts()

}

},

onPullDownRefresh() {

this.refreshContracts()

},

onReachBottom() {

if (this.data.hasMore && !this.data.loading) {

this.loadMoreContracts()

}

},

// 加载合同列表

async loadContracts(reset = true) {

if (reset) {

this.setData({

loading: true,

page: 1,

contracts: [],

hasMore: true

})

}

try {

const params = {

page: this.data.page,

page\_size: this.data.pageSize,

status: this.data.selectedStatus,

keyword: this.data.searchKeyword,

start\_date: this.data.dateRange.start,

end\_date: this.data.dateRange.end

}

const response = await api.get('/user/contracts', params)

if (response.success) {

const newContracts = response.data.contracts || []

this.setData({

contracts: reset ? newContracts : [...this.data.contracts, ...newContracts],

hasMore: newContracts.length === this.data.pageSize,

loading: false

})

} else {

throw new Error(response.message || '加载失败')

}

} catch (error) {

console.error('加载合同列表失败:', error)

wx.showToast({

title: error.message || '加载失败',

icon: 'none'

})

this.setData({ loading: false })

}

wx.stopPullDownRefresh()

},

// 加载更多合同

loadMoreContracts() {

this.setData({

page: this.data.page + 1

})

this.loadContracts(false)

},

// 刷新合同列表

refreshContracts() {

this.setData({ refreshing: true })

this.loadContracts()

this.loadStatistics()

setTimeout(() => {

this.setData({ refreshing: false })

}, 1000)

},

// 加载统计信息

async loadStatistics() {

try {

const response = await api.get('/user/contracts/statistics')

if (response.success) {

this.setData({

statistics: response.data

})

}

} catch (error) {

console.error('加载统计信息失败:', error)

}

},

// 搜索合同

onSearchInput(e) {

this.setData({

searchKeyword: e.detail.value

})

},

onSearchConfirm() {

this.loadContracts()

},

onSearchClear() {

this.setData({

searchKeyword: ''

})

this.loadContracts()

},

// 显示筛选器

showFilter() {

this.setData({

filterVisible: true

})

},

// 隐藏筛选器

hideFilter() {

this.setData({

filterVisible: false

})

},

// 选择状态筛选

onStatusChange(e) {

const selectedStatus = this.data.statusOptions[e.detail.value].value

this.setData({

selectedStatus

})

},

// 选择日期范围

onStartDateChange(e) {

this.setData({

'dateRange.start': e.detail.value

})

},

onEndDateChange(e) {

this.setData({

'dateRange.end': e.detail.value

})

},

// 应用筛选

applyFilter() {

this.hideFilter()

this.loadContracts()

},

// 重置筛选

resetFilter() {

this.setData({

selectedStatus: '',

dateRange: {

start: '',

end: ''

},

searchKeyword: ''

})

this.loadContracts()

},

// 查看合同详情

onContractTap(e) {

const contractId = e.currentTarget.dataset.id

wx.navigateTo({

url: `/pages/profile/contract-detail/contract-detail?id=${contractId}`

})

},

// 预览合同

async previewContract(e) {

const contractId = e.currentTarget.dataset.id

wx.showLoading({

title: '加载中...'

})

try {

const response = await api.get(`/user/contracts/${contractId}/preview`)

if (response.success) {

this.setData({

previewContract: response.data,

previewVisible: true

})

} else {

throw new Error(response.message || '预览失败')

}

} catch (error) {

console.error('预览合同失败:', error)

wx.showToast({

title: error.message || '预览失败',

icon: 'none'

})

}

wx.hideLoading()

},

// 关闭预览

closePreview() {

this.setData({

previewVisible: false,

previewContract: null

})

},

// 下载合同

async downloadContract(e) {

const contractId = e.currentTarget.dataset.id

const contract = this.data.contracts.find(c => c.id === contractId)

if (!contract) {

wx.showToast({

title: '合同不存在',

icon: 'none'

})

return

}

wx.showLoading({

title: '准备下载...'

})

try {

const response = await api.get(`/user/contracts/${contractId}/download`)

if (response.success && response.data.download\_url) {

wx.hideLoading()

// 下载文件

wx.downloadFile({

url: response.data.download\_url,

success: (res) => {

if (res.statusCode === 200) {

const filePath = res.tempFilePath

// 保存到相册或打开文档

wx.showActionSheet({

itemList: ['保存到相册', '用其他应用打开'],

success: (actionRes) => {

if (actionRes.tapIndex === 0) {

// 保存到相册

wx.saveImageToPhotosAlbum({

filePath: filePath,

success: () => {

wx.showToast({

title: '保存成功',

icon: 'success'

})

},

fail: () => {

wx.showToast({

title: '保存失败',

icon: 'none'

})

}

})

} else if (actionRes.tapIndex === 1) {

// 用其他应用打开

wx.openDocument({

filePath: filePath,

fileType: 'pdf',

success: () => {

console.log('打开文档成功')

},

fail: () => {

wx.showToast({

title: '打开失败',

icon: 'none'

})

}

})

}

}

})

} else {

wx.showToast({

title: '下载失败',

icon: 'none'

})

}

},

fail: () => {

wx.showToast({

title: '下载失败',

icon: 'none'

})

}

})

} else {

throw new Error(response.message || '获取下载链接失败')

}

} catch (error) {

console.error('下载合同失败:', error)

wx.showToast({

title: error.message || '下载失败',

icon: 'none'

})

}

wx.hideLoading()

},

// 签署合同

async signContract(e) {

const contractId = e.currentTarget.dataset.id

const contract = this.data.contracts.find(c => c.id === contractId)

if (!contract) {

wx.showToast({

title: '合同不存在',

icon: 'none'

})

return

}

if (contract.status !== 'pending') {

wx.showToast({

title: '合同状态不允许签署',

icon: 'none'

})

return

}

wx.showModal({

title: '确认签署',

content: '确定要签署这份合同吗？签署后将不能修改。',

success: async (res) => {

if (res.confirm) {

wx.showLoading({

title: '签署中...'

})

try {

const response = await api.post(`/user/contracts/${contractId}/sign`)

if (response.success) {

wx.showToast({

title: '签署成功',

icon: 'success'

})

// 刷新合同列表

this.refreshContracts()

} else {

throw new Error(response.message || '签署失败')

}

} catch (error) {

console.error('签署合同失败:', error)

wx.showToast({

title: error.message || '签署失败',

icon: 'none'

})

}

wx.hideLoading()

}

}

})

},

// 续签合同

renewContract(e) {

const contractId = e.currentTarget.dataset.id

const contract = this.data.contracts.find(c => c.id === contractId)

if (!contract) {

wx.showToast({

title: '合同不存在',

icon: 'none'

})

return

}

wx.showModal({

title: '续签合同',

content: '是否要为这份合同申请续签？我们将安排客户经理与您联系。',

success: async (res) => {

if (res.confirm) {

wx.showLoading({

title: '申请中...'

})

try {

const response = await api.post(`/user/contracts/${contractId}/renew`)

if (response.success) {

wx.showToast({

title: '续签申请已提交',

icon: 'success'

})

// 刷新合同列表

this.refreshContracts()

} else {

throw new Error(response.message || '申请失败')

}

} catch (error) {

console.error('申请续签失败:', error)

wx.showToast({

title: error.message || '申请失败',

icon: 'none'

})

}

wx.hideLoading()

}

}

})

},

// 取消合同

cancelContract(e) {

const contractId = e.currentTarget.dataset.id

const contract = this.data.contracts.find(c => c.id === contractId)

if (!contract) {

wx.showToast({

title: '合同不存在',

icon: 'none'

})

return

}

if (!['pending', 'signed'].includes(contract.status)) {

wx.showToast({

title: '当前状态不允许取消',

icon: 'none'

})

return

}

wx.showModal({

title: '取消合同',

content: '确定要取消这份合同吗？取消后将无法恢复。',

success: async (res) => {

if (res.confirm) {

wx.showLoading({

title: '取消中...'

})

try {

const response = await api.post(`/user/contracts/${contractId}/cancel`)

if (response.success) {

wx.showToast({

title: '取消成功',

icon: 'success'

})

// 刷新合同列表

this.refreshContracts()

} else {

throw new Error(response.message || '取消失败')

}

} catch (error) {

console.error('取消合同失败:', error)

wx.showToast({

title: error.message || '取消失败',

icon: 'none'

})

}

wx.hideLoading()

}

}

})

},

// 联系客服

contactCustomerService(e) {

const contractId = e.currentTarget.dataset.id

wx.showActionSheet({

itemList: ['在线客服', '电话咨询', '提交问题'],

success: (res) => {

if (res.tapIndex === 0) {

// 跳转到在线客服页面

wx.navigateTo({

url: `/pages/customer-service/chat/chat?contractId=${contractId}`

})

} else if (res.tapIndex === 1) {

// 拨打客服电话

wx.makePhoneCall({

phoneNumber: '400-123-4567',

fail: () => {

wx.showToast({

title: '拨号失败',

icon: 'none'

})

}

})

} else if (res.tapIndex === 2) {

// 跳转到问题提交页面

wx.navigateTo({

url: `/pages/customer-service/feedback/feedback?contractId=${contractId}`

})

}

}

})

},

// 格式化合同状态

formatContractStatus(status) {

const statusMap = {

'pending': '待签署',

'signed': '已签署',

'executing': '执行中',

'completed': '已完成',

'cancelled': '已取消'

}

return statusMap[status] || '未知状态'

},

// 格式化日期

formatDate(dateString) {

return util.formatDate(dateString, 'YYYY-MM-DD')

},

// 格式化金额

formatAmount(amount) {

return util.formatCurrency(amount)

},

// 检查是否即将到期

isNearExpiry(endDate) {

const now = new Date()

const expiry = new Date(endDate)

const diffTime = expiry.getTime() - now.getTime()

const diffDays = Math.ceil(diffTime / (1000 \* 60 \* 60 \* 24))

return diffDays <= 30 && diffDays > 0

},

// 获取合同状态颜色

getStatusColor(status) {

const colorMap = {

'pending': '#ff9500',

'signed': '#007aff',

'executing': '#34c759',

'completed': '#8e8e93',

'cancelled': '#ff3b30'

}

return colorMap[status] || '#8e8e93'

}

})

现在我将继续创建我的合同页面的模板文件：

我的合同页面模板 - pages/profile/contracts/contracts.wxml

<!-- pages/profile/contracts/contracts.wxml -->

<view class="container">

<!-- 统计卡片 -->

<view class="statistics-card">

<view class="stat-item">

<text class="stat-number">{{statistics.total}}</text>

<text class="stat-label">总合同</text>

</view>

<view class="stat-item">

<text class="stat-number">{{statistics.signed}}</text>

<text class="stat-label">已签署</text>

</view>

<view class="stat-item">

<text class="stat-number">{{statistics.executing}}</text>

<text class="stat-label">执行中</text>

</view>

<view class="stat-item">

<text class="stat-number">{{statistics.nearExpiry}}</text>

<text class="stat-label">即将到期</text>

</view>

</view>

<!-- 搜索和筛选栏 -->

<view class="search-bar">

<view class="search-input-wrapper">

<icon class="search-icon" type="search" size="16" color="#999"></icon>

<input

class="search-input"

placeholder="搜索合同编号、名称"

value="{{searchKeyword}}"

bindinput="onSearchInput"

bindconfirm="onSearchConfirm"

/>

<view wx:if="{{searchKeyword}}" class="clear-icon" bindtap="onSearchClear">

<icon type="clear" size="16" color="#999"></icon>

</view>

</view>

<view class="filter-btn" bindtap="showFilter">

<icon type="filter" size="16" color="#007aff"></icon>

<text>筛选</text>

</view>

</view>

<!-- 刷新指示器 -->

<view wx:if="{{refreshing}}" class="refresh-indicator">

<text>刷新中...</text>

</view>

<!-- 合同列表 -->

<view class="contracts-list">

<view

wx:for="{{contracts}}"

wx:key="id"

class="contract-item"

bindtap="onContractTap"

data-id="{{item.id}}"

>

<!-- 合同基本信息 -->

<view class="contract-header">

<view class="contract-title">

<text class="contract-name">{{item.contract\_name}}</text>

<view class="contract-status" style="color: {{getStatusColor(item.status)}}">

{{formatContractStatus(item.status)}}

</view>

</view>

<view class="contract-number">合同编号：{{item.contract\_number}}</view>

</view>

<!-- 合同详情 -->

<view class="contract-details">

<view class="detail-row">

<text class="detail-label">签署日期：</text>

<text class="detail-value">{{item.signed\_date ? formatDate(item.signed\_date) : '-'}}</text>

</view>

<view class="detail-row">

<text class="detail-label">服务期限：</text>

<text class="detail-value">

{{formatDate(item.start\_date)}} 至 {{formatDate(item.end\_date)}}

</text>

</view>

<view class="detail-row">

<text class="detail-label">合同金额：</text>

<text class="detail-value amount">{{formatAmount(item.amount)}}</text>

</view>

<view wx:if="{{item.product\_name}}" class="detail-row">

<text class="detail-label">服务产品：</text>

<text class="detail-value">{{item.product\_name}}</text>

</view>

</view>

<!-- 到期提醒 -->

<view wx:if="{{isNearExpiry(item.end\_date)}}" class="expiry-notice">

<icon type="warn" size="16" color="#ff9500"></icon>

<text>合同即将到期，请及时续签</text>

</view>

<!-- 操作按钮 -->

<view class="contract-actions">

<button

wx:if="{{item.status === 'pending'}}"

class="action-btn primary"

catchtap="signContract"

data-id="{{item.id}}"

>

立即签署

</button>

<button

wx:if="{{item.status === 'signed' || item.status === 'executing'}}"

class="action-btn secondary"

catchtap="renewContract"

data-id="{{item.id}}"

>

申请续签

</button>

<button

class="action-btn secondary"

catchtap="previewContract"

data-id="{{item.id}}"

>

预览

</button>

<button

wx:if="{{item.status === 'signed' || item.status === 'executing' || item.status === 'completed'}}"

class="action-btn secondary"

catchtap="downloadContract"

data-id="{{item.id}}"

>

下载

</button>

<button

wx:if="{{item.status === 'pending' || item.status === 'signed'}}"

class="action-btn danger"

catchtap="cancelContract"

data-id="{{item.id}}"

>

取消

</button>

<button

class="action-btn secondary"

catchtap="contactCustomerService"

data-id="{{item.id}}"

>

联系客服

</button>

</view>

</view>

</view>

<!-- 加载更多 -->

<view wx:if="{{loading}}" class="loading">

<text>加载中...</text>

</view>

<!-- 无数据提示 -->

<view wx:if="{{!loading && contracts.length === 0}}" class="empty-state">

<icon type="contract" size="64" color="#ccc"></icon>

<text class="empty-text">暂无合同记录</text>

<text class="empty-hint">您还没有任何合同，去看看我们的产品吧</text>

<button class="empty-btn" bindtap="goToProducts">浏览产品</button>

</view>

<!-- 筛选弹窗 -->

<view wx:if="{{filterVisible}}" class="filter-modal">

<view class="filter-mask" bindtap="hideFilter"></view>

<view class="filter-content">

<view class="filter-header">

<text class="filter-title">筛选条件</text>

<icon class="close-icon" type="clear" size="20" bindtap="hideFilter"></icon>

</view>

<view class="filter-section">

<text class="section-title">合同状态</text>

<picker

bindchange="onStatusChange"

value="{{selectedStatusIndex}}"

range="{{statusOptions}}"

range-key="label"

>

<view class="picker-item">

<text>{{statusOptions[selectedStatusIndex].label || '选择状态'}}</text>

<icon type="arrow\_right" size="16" color="#999"></icon>

</view>

</picker>

</view>

<view class="filter-section">

<text class="section-title">签署日期</text>

<view class="date-range">

<picker mode="date" bindchange="onStartDateChange" value="{{dateRange.start}}">

<view class="date-picker">

<text>{{dateRange.start || '开始日期'}}</text>

<icon type="calendar" size="16" color="#999"></icon>

</view>

</picker>

<text class="date-separator">至</text>

<picker mode="date" bindchange="onEndDateChange" value="{{dateRange.end}}">

<view class="date-picker">

<text>{{dateRange.end || '结束日期'}}</text>

<icon type="calendar" size="16" color="#999"></icon>

</view>

</picker>

</view>

</view>

<view class="filter-actions">

<button class="filter-reset-btn" bindtap="resetFilter">重置</button>

<button class="filter-apply-btn" bindtap="applyFilter">应用</button>

</view>

</view>

</view>

<!-- 合同预览弹窗 -->

<view wx:if="{{previewVisible}}" class="preview-modal">

<view class="preview-mask" bindtap="closePreview"></view>

<view class="preview-content">

<view class="preview-header">

<text class="preview-title">合同预览</text>

<icon class="close-icon" type="clear" size="20" bindtap="closePreview"></icon>

</view>

<scroll-view scroll-y class="preview-body">

<view wx:if="{{previewContract}}" class="contract-preview">

<view class="preview-info">

<text class="preview-contract-name">{{previewContract.contract\_name}}</text>

<text class="preview-contract-number">{{previewContract.contract\_number}}</text>

</view>

<view class="preview-parties">

<view class="party">

<text class="party-title">甲方（服务方）</text>

<text class="party-name">{{previewContract.party\_a\_name}}</text>

<text class="party-address">{{previewContract.party\_a\_address}}</text>

</view>

<view class="party">

<text class="party-title">乙方（客户方）</text>

<text class="party-name">{{previewContract.party\_b\_name}}</text>

<text class="party-address">{{previewContract.party\_b\_address}}</text>

</view>

</view>

<view class="preview-terms">

<text class="terms-title">主要条款</text>

<view class="term-item">

<text class="term-label">服务内容：</text>

<text class="term-value">{{previewContract.service\_content}}</text>

</view>

<view class="term-item">

<text class="term-label">服务期限：</text>

<text class="term-value">

{{formatDate(previewContract.start\_date)}} 至 {{formatDate(previewContract.end\_date)}}

</text>

</view>

<view class="term-item">

<text class="term-label">合同金额：</text>

<text class="term-value">{{formatAmount(previewContract.amount)}}</text>

</view>

<view class="term-item">

<text class="term-label">付款方式：</text>

<text class="term-value">{{previewContract.payment\_method}}</text>

</view>

</view>

</view>

</scroll-view>

<view class="preview-actions">

<button class="preview-btn secondary" bindtap="closePreview">关闭</button>

<button

wx:if="{{previewContract.status === 'pending'}}"

class="preview-btn primary"

bindtap="signContract"

data-id="{{previewContract.id}}"

>

立即签署

</button>

</view>

</view>

</view>

</view>

现在我将创建我的合同页面的样式文件：

我的合同页面样式文件 - pages/profile/contracts/contracts.wxss

/\* 我的合同页面样式 \*/

.container {

padding: 0;

background-color: var(--bg-color-light);

min-height: 100vh;

}

/\* 筛选区域 \*/

.filter-section {

background: #fff;

padding: 30rpx;

border-bottom: 2rpx solid #f0f0f0;

position: sticky;

top: 0;

z-index: 100;

}

.filter-tabs {

display: flex;

justify-content: space-around;

}

.filter-tab {

flex: 1;

text-align: center;

padding: 20rpx;

font-size: 28rpx;

color: var(--text-color-secondary);

border-radius: 8rpx;

transition: all 0.3s ease;

}

.filter-tab.active {

color: var(--primary-color);

background-color: var(--primary-light);

font-weight: 500;

}

/\* 合同列表 \*/

.contracts-list {

padding: 20rpx;

}

.contract-item {

background: #fff;

border-radius: var(--border-radius);

margin-bottom: 20rpx;

padding: 30rpx;

box-shadow: var(--shadow-light);

position: relative;

overflow: hidden;

}

.contract-item::before {

content: '';

position: absolute;

left: 0;

top: 0;

width: 8rpx;

height: 100%;

background: linear-gradient(to bottom, var(--primary-color), var(--primary-light));

}

.contract-header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 20rpx;

}

.contract-title {

font-size: 32rpx;

font-weight: 500;

color: var(--text-color-primary);

flex: 1;

margin-right: 20rpx;

}

.contract-status {

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 24rpx;

font-weight: 500;

white-space: nowrap;

}

.status-signed {

background: var(--success-light);

color: var(--success-color);

}

.status-pending {

background: var(--warning-light);

color: var(--warning-color);

}

.status-expired {

background: var(--error-light);

color: var(--error-color);

}

.status-terminated {

background: #f5f5f5;

color: var(--text-color-secondary);

}

.contract-info {

margin-bottom: 20rpx;

}

.info-row {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 16rpx;

}

.info-row:last-child {

margin-bottom: 0;

}

.info-label {

font-size: 26rpx;

color: var(--text-color-secondary);

width: 160rpx;

flex-shrink: 0;

}

.info-value {

font-size: 26rpx;

color: var(--text-color-primary);

flex: 1;

text-align: right;

}

.info-value.highlight {

color: var(--primary-color);

font-weight: 500;

}

.info-value.warning {

color: var(--warning-color);

font-weight: 500;

}

.contract-actions {

display: flex;

justify-content: flex-end;

gap: 20rpx;

margin-top: 20rpx;

padding-top: 20rpx;

border-top: 2rpx solid #f0f0f0;

}

.action-btn {

padding: 16rpx 32rpx;

border-radius: 8rpx;

font-size: 26rpx;

border: none;

cursor: pointer;

transition: all 0.3s ease;

}

.action-btn.primary {

background: var(--primary-color);

color: #fff;

}

.action-btn.secondary {

background: #f8f9fa;

color: var(--text-color-primary);

border: 2rpx solid #e9ecef;

}

.action-btn.danger {

background: var(--error-color);

color: #fff;

}

.action-btn:hover {

opacity: 0.8;

}

.action-btn:active {

transform: scale(0.98);

}

/\* 空状态 \*/

.empty-state {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 120rpx 40rpx;

text-align: center;

}

.empty-icon {

width: 200rpx;

height: 200rpx;

opacity: 0.3;

margin-bottom: 40rpx;

}

.empty-text {

font-size: 28rpx;

color: var(--text-color-secondary);

margin-bottom: 20rpx;

}

.empty-tip {

font-size: 24rpx;

color: var(--text-color-light);

line-height: 1.6;

}

/\* 加载状态 \*/

.loading-container {

display: flex;

justify-content: center;

align-items: center;

padding: 80rpx;

}

.loading-text {

font-size: 26rpx;

color: var(--text-color-secondary);

margin-left: 20rpx;

}

/\* 续签提醒 \*/

.renewal-reminder {

background: linear-gradient(135deg, var(--warning-light), #fff5e6);

border: 2rpx solid var(--warning-color);

border-radius: var(--border-radius);

padding: 30rpx;

margin: 20rpx;

position: relative;

}

.reminder-icon {

position: absolute;

right: 20rpx;

top: 20rpx;

width: 40rpx;

height: 40rpx;

color: var(--warning-color);

}

.reminder-title {

font-size: 30rpx;

font-weight: 500;

color: var(--warning-color);

margin-bottom: 10rpx;

}

.reminder-content {

font-size: 26rpx;

color: var(--text-color-primary);

line-height: 1.5;

margin-bottom: 20rpx;

}

.reminder-action {

display: flex;

justify-content: flex-end;

}

.reminder-btn {

background: var(--warning-color);

color: #fff;

padding: 16rpx 32rpx;

border-radius: 8rpx;

font-size: 26rpx;

border: none;

}

/\* 底部安全区域 \*/

.safe-area-bottom {

height: env(safe-area-inset-bottom);

background: var(--bg-color-light);

}

/\* 下拉刷新样式 \*/

.refresh-container {

display: flex;

justify-content: center;

align-items: center;

padding: 40rpx;

}

.refresh-text {

font-size: 26rpx;

color: var(--text-color-secondary);

margin-left: 20rpx;

}

/\* 合同预览浮层样式 \*/

.contract-preview {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.5);

z-index: 999;

display: flex;

align-items: center;

justify-content: center;

}

.preview-content {

background: #fff;

width: 90%;

height: 80%;

border-radius: var(--border-radius);

display: flex;

flex-direction: column;

}

.preview-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 30rpx;

border-bottom: 2rpx solid #f0f0f0;

}

.preview-title {

font-size: 32rpx;

font-weight: 500;

color: var(--text-color-primary);

}

.preview-close {

width: 60rpx;

height: 60rpx;

display: flex;

align-items: center;

justify-content: center;

border-radius: 50%;

background: #f8f9fa;

color: var(--text-color-secondary);

}

.preview-body {

flex: 1;

overflow-y: auto;

padding: 30rpx;

}

/\* 响应式适配 \*/

@media screen and (max-width: 320px) {

.contract-item {

padding: 20rpx;

}

.contract-title {

font-size: 28rpx;

}

.filter-tab {

padding: 16rpx 10rpx;

font-size: 24rpx;

}

}

/\* 动画效果 \*/

.contract-item {

animation: slideInUp 0.3s ease-out;

}

@keyframes slideInUp {

from {

opacity: 0;

transform: translateY(30rpx);

}

to {

opacity: 1;

transform: translateY(0);

}

}

/\* 长按选择效果 \*/

.contract-item.selecting {

background: var(--primary-light);

border: 2rpx solid var(--primary-color);

}

.contract-item.selected {

background: var(--primary-light);

border: 2rpx solid var(--primary-color);

position: relative;

}

.contract-item.selected::after {

content: '✓';

position: absolute;

right: 20rpx;

top: 20rpx;

width: 40rpx;

height: 40rpx;

background: var(--primary-color);

color: #fff;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

font-size: 24rpx;

font-weight: bold;

}

现在继续开发下一个模块：我的合同页面配置文件。我的合同页面配置文件 - pages/profile/contracts/contracts.json

{

"navigationBarTitleText": "我的合同",

"navigationBarBackgroundColor": "#ffffff",

"navigationBarTextStyle": "black",

"backgroundColor": "#f8f9fa",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"usingComponents": {

"loading": "/components/loading/loading",

"empty": "/components/empty/empty"

}

}

接下来开发客户经理功能小程序页面模块。首先创建客户经理工作台首页：客户经理工作台首页逻辑 - pages/manager/index/index.js

// 客户经理工作台首页

const app = getApp()

const api = require('../../../utils/api')

Page({

data: {

userInfo: {},

workbenchData: {

todayData: {

newCustomers: 0,

followUpTasks: 0,

newOrders: 0,

orderAmount: 0

},

monthData: {

newCustomers: 0,

totalOrders: 0,

orderAmount: 0,

targetProgress: 0

},

statistics: {

totalCustomers: 0,

activeCustomers: 0,

completedOrders: 0,

satisfaction: 0

}

},

quickActions: [

{

icon: 'customers',

title: '我的客户',

subtitle: '客户管理',

url: '/pages/manager/customers/customers',

color: '#1890ff'

},

{

icon: 'follow',

title: '跟进任务',

subtitle: '待跟进客户',

url: '/pages/manager/follow/follow',

color: '#52c41a'

},

{

icon: 'performance',

title: '业绩统计',

subtitle: '我的业绩',

url: '/pages/manager/performance/performance',

color: '#faad14'

},

{

icon: 'orders',

title: '订单管理',

subtitle: '客户订单',

url: '/pages/manager/orders/orders',

color: '#722ed1'

}

],

recentCustomers: [],

urgentTasks: [],

loading: true,

refreshing: false

},

onLoad(options) {

this.checkManagerAuth()

},

onShow() {

this.loadWorkbenchData()

this.loadRecentData()

},

// 检查客户经理权限

checkManagerAuth() {

const userInfo = app.globalData.userInfo

if (!userInfo || userInfo.role !== 'manager') {

wx.showModal({

title: '权限不足',

content: '您没有客户经理权限，请联系管理员',

showCancel: false,

success: () => {

wx.switchTab({

url: '/pages/profile/index/index'

})

}

})

return false

}

this.setData({ userInfo })

return true

},

// 加载工作台数据

async loadWorkbenchData() {

try {

this.setData({ loading: true })

const result = await api.get('/manager/workbench')

if (result.success) {

this.setData({

workbenchData: result.data,

loading: false

})

} else {

throw new Error(result.message)

}

} catch (error) {

console.error('加载工作台数据失败:', error)

this.setData({ loading: false })

wx.showToast({

title: '加载失败',

icon: 'error'

})

}

},

// 加载最近数据

async loadRecentData() {

try {

const [customersResult, tasksResult] = await Promise.all([

api.get('/manager/recent-customers', { limit: 5 }),

api.get('/manager/urgent-tasks', { limit: 5 })

])

if (customersResult.success) {

this.setData({

recentCustomers: customersResult.data

})

}

if (tasksResult.success) {

this.setData({

urgentTasks: tasksResult.data

})

}

} catch (error) {

console.error('加载最近数据失败:', error)

}

},

// 下拉刷新

async onPullDownRefresh() {

this.setData({ refreshing: true })

try {

await Promise.all([

this.loadWorkbenchData(),

this.loadRecentData()

])

} catch (error) {

console.error('刷新失败:', error)

} finally {

this.setData({ refreshing: false })

wx.stopPullDownRefresh()

}

},

// 快捷操作点击

onQuickActionTap(e) {

const { url } = e.currentTarget.dataset

if (url.startsWith('/pages')) {

wx.navigateTo({ url })

} else {

wx.switchTab({ url })

}

},

// 查看今日数据详情

onTodayDataTap(e) {

const { type } = e.currentTarget.dataset

const routes = {

customers: '/pages/manager/customers/customers?filter=today',

tasks: '/pages/manager/follow/follow?filter=today',

orders: '/pages/manager/orders/orders?filter=today'

}

if (routes[type]) {

wx.navigateTo({

url: routes[type]

})

}

},

// 查看客户详情

onCustomerTap(e) {

const { customerId } = e.currentTarget.dataset

wx.navigateTo({

url: `/pages/manager/customer-detail/customer-detail?id=${customerId}`

})

},

// 立即跟进

onFollowTap(e) {

const { customerId, taskId } = e.currentTarget.dataset

wx.navigateTo({

url: `/pages/manager/follow-detail/follow-detail?customerId=${customerId}&taskId=${taskId}`

})

},

// 一键电话

onCallTap(e) {

const { phone } = e.currentTarget.dataset

if (!phone) {

wx.showToast({

title: '暂无电话号码',

icon: 'none'

})

return

}

wx.makePhoneCall({

phoneNumber: phone,

fail: (error) => {

console.error('拨打电话失败:', error)

wx.showToast({

title: '拨打失败',

icon: 'error'

})

}

})

},

// 查看更多客户

onMoreCustomersTap() {

wx.navigateTo({

url: '/pages/manager/customers/customers'

})

},

// 查看更多任务

onMoreTasksTap() {

wx.navigateTo({

url: '/pages/manager/follow/follow'

})

},

// 目标进度点击

onTargetTap() {

wx.navigateTo({

url: '/pages/manager/performance/performance?tab=target'

})

},

// 格式化金额

formatAmount(amount) {

if (!amount) return '0'

return (amount / 10000).toFixed(1) + '万'

},

// 格式化日期

formatDate(dateStr) {

if (!dateStr) return ''

const date = new Date(dateStr)

const now = new Date()

const diff = now.getTime() - date.getTime()

const days = Math.floor(diff / (1000 \* 60 \* 60 \* 24))

if (days === 0) return '今天'

if (days === 1) return '昨天'

if (days < 7) return `${days}天前`

return `${date.getMonth() + 1}/${date.getDate()}`

},

// 获取任务优先级颜色

getTaskPriorityColor(priority) {

const colors = {

high: '#ff4d4f',

medium: '#faad14',

low: '#52c41a'

}

return colors[priority] || '#d9d9d9'

},

// 分享工作台

onShareAppMessage() {

return {

title: '电力销售客户经理工作台',

path: '/pages/manager/index/index',

imageUrl: '/images/share-manager.png'

}

},

// 分享到朋友圈

onShareTimeline() {

return {

title: '电力销售客户经理工作台',

imageUrl: '/images/share-manager.png'

}

}

})

客户经理工作台首页模板 - pages/manager/index/index.wxml

<!--客户经理工作台首页-->

<view class="container">

<!-- 加载状态 -->

<loading wx:if="{{loading}}" />

<!-- 主要内容 -->

<view wx:else class="content">

<!-- 欢迎区域 -->

<view class="welcome-section">

<view class="welcome-header">

<view class="welcome-text">

<view class="greeting">

<text class="time">{{refreshing ? '刷新中...' : '早上好'}}</text>

<text class="name">{{userInfo.name || '客户经理'}}</text>

</view>

<view class="date">{{refreshing ? '' : '今天是个好日子，加油！'}}</view>

</view>

<view class="avatar">

<image src="{{userInfo.avatar\_url || '/images/default-avatar.png'}}" mode="aspectFill" />

</view>

</view>

</view>

<!-- 今日数据 -->

<view class="today-section">

<view class="section-title">今日概况</view>

<view class="today-grid">

<view class="today-item" bindtap="onTodayDataTap" data-type="customers">

<view class="today-number primary">{{workbenchData.todayData.newCustomers}}</view>

<view class="today-label">新增客户</view>

</view>

<view class="today-item" bindtap="onTodayDataTap" data-type="tasks">

<view class="today-number warning">{{workbenchData.todayData.followUpTasks}}</view>

<view class="today-label">待跟进</view>

</view>

<view class="today-item" bindtap="onTodayDataTap" data-type="orders">

<view class="today-number">{{workbenchData.todayData.newOrders}}</view>

<view class="today-label">新订单</view>

</view>

<view class="today-item">

<view class="today-number success">{{formatAmount(workbenchData.todayData.orderAmount)}}</view>

<view class="today-label">成交金额</view>

</view>

</view>

</view>

<!-- 快捷操作 -->

<view class="quick-section">

<view class="section-title">快捷操作</view>

<view class="quick-grid">

<view

wx:for="{{quickActions}}"

wx:key="title"

class="quick-item"

bindtap="onQuickActionTap"

data-url="{{item.url}}"

>

<view class="quick-icon" style="background-color: {{item.color}}20;">

<text class="icon-{{item.icon}}" style="color: {{item.color}};"></text>

</view>

<view class="quick-content">

<view class="quick-title">{{item.title}}</view>

<view class="quick-subtitle">{{item.subtitle}}</view>

</view>

<view class="quick-arrow">

<text class="icon-arrow-right"></text>

</view>

</view>

</view>

</view>

<!-- 本月业绩 -->

<view class="performance-section">

<view class="section-title">本月业绩</view>

<view class="performance-card">

<view class="performance-header">

<view class="performance-item">

<view class="performance-number">{{workbenchData.monthData.newCustomers}}</view>

<view class="performance-label">新增客户</view>

</view>

<view class="performance-item">

<view class="performance-number">{{workbenchData.monthData.totalOrders}}</view>

<view class="performance-label">成交订单</view>

</view>

<view class="performance-item">

<view class="performance-number">{{formatAmount(workbenchData.monthData.orderAmount)}}</view>

<view class="performance-label">成交金额</view>

</view>

</view>

<view class="target-progress" bindtap="onTargetTap">

<view class="target-header">

<view class="target-title">目标完成度</view>

<view class="target-percent">{{workbenchData.monthData.targetProgress}}%</view>

</view>

<view class="progress-bar">

<view class="progress-fill" style="width: {{workbenchData.monthData.targetProgress}}%;"></view>

</view>

</view>

</view>

</view>

<!-- 最近客户 -->

<view wx:if="{{recentCustomers.length > 0}}" class="recent-section">

<view class="section-header">

<view class="section-title">最近客户</view>

<view class="section-more" bindtap="onMoreCustomersTap">

<text>查看更多</text>

<text class="icon-arrow-right"></text>

</view>

</view>

<view class="recent-list">

<view

wx:for="{{recentCustomers}}"

wx:key="id"

class="recent-item"

bindtap="onCustomerTap"

data-customer-id="{{item.id}}"

>

<view class="customer-avatar">

<image src="{{item.avatar\_url || '/images/default-company.png'}}" mode="aspectFill" />

</view>

<view class="customer-info">

<view class="customer-name">{{item.company\_name || item.name}}</view>

<view class="customer-contact">{{item.phone}}</view>

<view class="customer-time">{{formatDate(item.last\_contact\_time)}}</view>

</view>

<view class="customer-status">

<view class="status-tag status-{{item.status}}">{{item.status\_text}}</view>

</view>

<view class="customer-actions">

<view class="action-btn" bindtap="onCallTap" data-phone="{{item.phone}}" catchtap="true">

<text class="icon-phone"></text>

</view>

</view>

</view>

</view>

</view>

<!-- 紧急任务 -->

<view wx:if="{{urgentTasks.length > 0}}" class="urgent-section">

<view class="section-header">

<view class="section-title">紧急任务</view>

<view class="section-more" bindtap="onMoreTasksTap">

<text>查看更多</text>

<text class="icon-arrow-right"></text>

</view>

</view>

<view class="urgent-list">

<view

wx:for="{{urgentTasks}}"

wx:key="id"

class="urgent-item"

>

<view class="task-priority" style="background-color: {{getTaskPriorityColor(item.priority)}};"></view>

<view class="task-content">

<view class="task-title">{{item.title}}</view>

<view class="task-customer">客户：{{item.customer\_name}}</view>

<view class="task-time">截止：{{item.deadline}}</view>

</view>

<view class="task-action">

<button

class="follow-btn"

size="mini"

bindtap="onFollowTap"

data-customer-id="{{item.customer\_id}}"

data-task-id="{{item.id}}"

>

立即跟进

</button>

</view>

</view>

</view>

</view>

<!-- 空状态提示 -->

<view wx:if="{{recentCustomers.length === 0 && urgentTasks.length === 0}}" class="empty-section">

<empty

icon="manager"

title="暂无待处理事项"

description="您可以主动添加客户或等待系统分配"

/>

</view>

</view>

<!-- 底部安全区域 -->

<view class="safe-area-bottom"></view>

</view>

<!-- 浮动操作按钮 (可选) -->

<view class="fab-container" wx:if="{{showFab}}">

<view class="fab-main" bindtap="onFabTap">

<text class="icon-add"></text>

</view>

<view class="fab-options" wx:if="{{fabExpanded}}">

<view class="fab-option" bindtap="onAddCustomerTap">

<text class="icon-user-add"></text>

<text>添加客户</text>

</view>

<view class="fab-option" bindtap="onQuickFollowTap">

<text class="icon-phone"></text>

<text>快速跟进</text>

</view>

</view>

</view>

<!-- 下拉刷新提示 -->

<view class="refresh-tip" wx:if="{{showRefreshTip}}">

<text class="icon-refresh"></text>

<text>数据已更新</text>

</view>

现在让我继续开发下一个模块：**客户经理工作台首页样式文件**。

客户经理工作台首页样式文件 - pages/manager/index/index.wxss

/\* 客户经理工作台首页样式 \*/

@import "/styles/common.wxss";

.container {

min-height: 100vh;

background-color: var(--bg-color-light);

}

/\* ==================== 欢迎区域 ==================== \*/

.welcome-section {

background: linear-gradient(135deg, var(--primary-color) 0%, #4facfe 100%);

padding: var(--spacing-lg) var(--spacing-md) var(--spacing-xl);

color: white;

}

.welcome-header {

display: flex;

justify-content: space-between;

align-items: center;

}

.welcome-text {

flex: 1;

}

.greeting {

display: flex;

align-items: baseline;

margin-bottom: var(--spacing-xs);

}

.time {

font-size: 32rpx;

font-weight: 600;

margin-right: var(--spacing-xs);

}

.name {

font-size: 28rpx;

opacity: 0.9;

}

.date {

font-size: 24rpx;

opacity: 0.8;

margin-top: var(--spacing-xs);

}

.avatar {

width: 120rpx;

height: 120rpx;

border-radius: 50%;

overflow: hidden;

border: 4rpx solid rgba(255, 255, 255, 0.3);

box-shadow: 0 8rpx 20rpx rgba(0, 0, 0, 0.1);

}

.avatar image {

width: 100%;

height: 100%;

}

/\* ==================== 今日数据 ==================== \*/

.today-section {

margin: var(--spacing-lg) var(--spacing-md);

}

.section-title {

font-size: 32rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: var(--spacing-md);

position: relative;

padding-left: var(--spacing-sm);

}

.section-title::before {

content: '';

position: absolute;

left: 0;

top: 50%;

transform: translateY(-50%);

width: 6rpx;

height: 32rpx;

background-color: var(--primary-color);

border-radius: 3rpx;

}

.today-grid {

display: flex;

background: white;

border-radius: var(--border-radius-lg);

box-shadow: var(--shadow-light);

overflow: hidden;

}

.today-item {

flex: 1;

padding: var(--spacing-lg) var(--spacing-md);

text-align: center;

position: relative;

transition: all 0.2s ease;

}

.today-item:active {

background-color: var(--bg-color-light);

}

.today-item:not(:last-child)::after {

content: '';

position: absolute;

right: 0;

top: 20%;

width: 2rpx;

height: 60%;

background-color: var(--border-color);

}

.today-number {

font-size: 48rpx;

font-weight: 700;

color: var(--text-color-primary);

margin-bottom: var(--spacing-xs);

line-height: 1;

}

.today-number.primary {

color: var(--primary-color);

}

.today-number.warning {

color: var(--warning-color);

}

.today-number.success {

color: var(--success-color);

}

.today-label {

font-size: 24rpx;

color: var(--text-color-secondary);

}

/\* ==================== 快捷操作 ==================== \*/

.quick-section {

margin: var(--spacing-lg) var(--spacing-md);

}

.quick-grid {

background: white;

border-radius: var(--border-radius-lg);

box-shadow: var(--shadow-light);

overflow: hidden;

}

.quick-item {

display: flex;

align-items: center;

padding: var(--spacing-lg) var(--spacing-md);

border-bottom: 2rpx solid var(--border-color);

transition: all 0.2s ease;

}

.quick-item:last-child {

border-bottom: none;

}

.quick-item:active {

background-color: var(--bg-color-light);

}

.quick-icon {

width: 80rpx;

height: 80rpx;

border-radius: var(--border-radius-md);

display: flex;

align-items: center;

justify-content: center;

margin-right: var(--spacing-md);

}

.quick-icon text {

font-size: 36rpx;

}

.quick-content {

flex: 1;

}

.quick-title {

font-size: 28rpx;

font-weight: 500;

color: var(--text-color-primary);

margin-bottom: var(--spacing-xs);

}

.quick-subtitle {

font-size: 24rpx;

color: var(--text-color-secondary);

}

.quick-arrow {

color: var(--text-color-placeholder);

font-size: 24rpx;

}

/\* ==================== 本月业绩 ==================== \*/

.performance-section {

margin: var(--spacing-lg) var(--spacing-md);

}

.performance-card {

background: white;

border-radius: var(--border-radius-lg);

box-shadow: var(--shadow-light);

padding: var(--spacing-lg);

}

.performance-header {

display: flex;

margin-bottom: var(--spacing-lg);

}

.performance-item {

flex: 1;

text-align: center;

}

.performance-number {

font-size: 36rpx;

font-weight: 600;

color: var(--text-color-primary);

margin-bottom: var(--spacing-xs);

}

.performance-label {

font-size: 24rpx;

color: var(--text-color-secondary);

}

.target-progress {

border-top: 2rpx solid var(--border-color);

padding-top: var(--spacing-lg);

}

.target-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: var(--spacing-md);

}

.target-title {

font-size: 28rpx;

color: var(--text-color-primary);

}

.target-percent {

font-size: 32rpx;

font-weight: 600;

color: var(--primary-color);

}

.progress-bar {

height: 12rpx;

background-color: var(--bg-color-light);

border-radius: 6rpx;

overflow: hidden;

}

.progress-fill {

height: 100%;

background: linear-gradient(90deg, var(--primary-color) 0%, #4facfe 100%);

border-radius: 6rpx;

transition: width 0.3s ease;

}

/\* ==================== 最近客户 ==================== \*/

.recent-section, .urgent-section {

margin: var(--spacing-lg) var(--spacing-md);

}

.section-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: var(--spacing-md);

}

.section-more {

display: flex;

align-items: center;

font-size: 24rpx;

color: var(--text-color-secondary);

}

.section-more text:last-child {

margin-left: var(--spacing-xs);

}

.recent-list, .urgent-list {

background: white;

border-radius: var(--border-radius-lg);

box-shadow: var(--shadow-light);

overflow: hidden;

}

.recent-item {

display: flex;

align-items: center;

padding: var(--spacing-md);

border-bottom: 2rpx solid var(--border-color);

transition: all 0.2s ease;

}

.recent-item:last-child {

border-bottom: none;

}

.recent-item:active {

background-color: var(--bg-color-light);

}

.customer-avatar {

width: 80rpx;

height: 80rpx;

border-radius: var(--border-radius-md);

overflow: hidden;

margin-right: var(--spacing-md);

background-color: var(--bg-color-light);

}

.customer-avatar image {

width: 100%;

height: 100%;

}

.customer-info {

flex: 1;

}

.customer-name {

font-size: 28rpx;

font-weight: 500;

color: var(--text-color-primary);

margin-bottom: var(--spacing-xs);

}

.customer-contact {

font-size: 24rpx;

color: var(--text-color-secondary);

margin-bottom: var(--spacing-xs);

}

.customer-time {

font-size: 22rpx;

color: var(--text-color-placeholder);

}

.customer-status {

margin-right: var(--spacing-md);

}

.status-tag {

padding: var(--spacing-xs) var(--spacing-sm);

border-radius: var(--border-radius-sm);

font-size: 22rpx;

color: white;

}

.status-potential {

background-color: var(--info-color);

}

.status-following {

background-color: var(--warning-color);

}

.status-negotiating {

background-color: var(--primary-color);

}

.status-deal {

background-color: var(--success-color);

}

.status-lost {

background-color: var(--error-color);

}

.customer-actions {

display: flex;

align-items: center;

}

.action-btn {

width: 60rpx;

height: 60rpx;

border-radius: 50%;

background-color: var(--primary-color);

color: white;

display: flex;

align-items: center;

justify-content: center;

font-size: 24rpx;

transition: all 0.2s ease;

}

.action-btn:active {

transform: scale(0.95);

opacity: 0.8;

}

/\* ==================== 紧急任务 ==================== \*/

.urgent-item {

display: flex;

align-items: center;

padding: var(--spacing-md);

border-bottom: 2rpx solid var(--border-color);

}

.urgent-item:last-child {

border-bottom: none;

}

.task-priority {

width: 8rpx;

height: 80rpx;

border-radius: 4rpx;

margin-right: var(--spacing-md);

}

.task-content {

flex: 1;

}

.task-title {

font-size: 28rpx;

font-weight: 500;

color: var(--text-color-primary);

margin-bottom: var(--spacing-xs);

}

.task-customer {

font-size: 24rpx;

color: var(--text-color-secondary);

margin-bottom: var(--spacing-xs);

}

.task-time {

font-size: 22rpx;

color: var(--warning-color);

}

.task-action {

margin-left: var(--spacing-md);

}

.follow-btn {

background-color: var(--primary-color) !important;

color: white !important;

border: none !important;

font-size: 24rpx !important;

padding: var(--spacing-sm) var(--spacing-md) !important;

border-radius: var(--border-radius-md) !important;

}

/\* ==================== 空状态 ==================== \*/

.empty-section {

margin: var(--spacing-xl) var(--spacing-md);

background: white;

border-radius: var(--border-radius-lg);

box-shadow: var(--shadow-light);

padding: var(--spacing-xl);

text-align: center;

}

/\* ==================== 浮动操作按钮 ==================== \*/

.fab-container {

position: fixed;

bottom: 120rpx;

right: var(--spacing-md);

z-index: 1000;

}

.fab-main {

width: 100rpx;

height: 100rpx;

border-radius: 50%;

background-color: var(--primary-color);

color: white;

display: flex;

align-items: center;

justify-content: center;

font-size: 36rpx;

box-shadow: 0 8rpx 20rpx rgba(0, 0, 0, 0.15);

transition: all 0.3s ease;

}

.fab-main:active {

transform: scale(0.95);

}

.fab-options {

position: absolute;

bottom: 120rpx;

right: 0;

background: white;

border-radius: var(--border-radius-lg);

box-shadow: var(--shadow-medium);

padding: var(--spacing-sm);

min-width: 200rpx;

}

.fab-option {

display: flex;

align-items: center;

padding: var(--spacing-md);

border-radius: var(--border-radius-md);

transition: all 0.2s ease;

}

.fab-option:active {

background-color: var(--bg-color-light);

}

.fab-option text:first-child {

font-size: 24rpx;

color: var(--primary-color);

margin-right: var(--spacing-sm);

}

.fab-option text:last-child {

font-size: 26rpx;

color: var(--text-color-primary);

}

/\* ==================== 下拉刷新提示 ==================== \*/

.refresh-tip {

position: fixed;

top: 20rpx;

left: 50%;

transform: translateX(-50%);

background-color: var(--success-color);

color: white;

padding: var(--spacing-sm) var(--spacing-md);

border-radius: var(--border-radius-lg);

font-size: 24rpx;

display: flex;

align-items: center;

z-index: 1000;

animation: slideDown 0.3s ease;

}

.refresh-tip text:first-child {

margin-right: var(--spacing-xs);

}

@keyframes slideDown {

0% {

transform: translateX(-50%) translateY(-100%);

opacity: 0;

}

100% {

transform: translateX(-50%) translateY(0);

opacity: 1;

}

}

/\* ==================== 响应式适配 ==================== \*/

/\* 小屏幕适配 \*/

@media (max-width: 375px) {

.today-number {

font-size: 40rpx;

}

.performance-number {

font-size: 32rpx;

}

.quick-icon {

width: 70rpx;

height: 70rpx;

}

.customer-avatar {

width: 70rpx;

height: 70rpx;

}

}

/\* 大屏幕适配 \*/

@media (min-width: 414px) {

.today-number {

font-size: 52rpx;

}

.performance-number {

font-size: 40rpx;

}

}

现在让我继续开发下一个模块：**客户经理工作台首页配置文件**。

客户经理工作台首页配置文件 - pages/manager/index/index.json

{

"navigationBarTitleText": "工作台",

"navigationBarBackgroundColor": "#2E7CF6",

"navigationBarTextStyle": "white",

"backgroundColor": "#f8f9fa",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 100,

"usingComponents": {

"loading": "/components/loading/loading",

"empty": "/components/empty/empty"

}

}

接下来开发：**我的客户页面逻辑 - pages/manager/customers/customers.js**

// 客户经理 - 我的客户页面

const api = require('../../../utils/api')

const util = require('../../../utils/util')

Page({

data: {

loading: false,

refreshing: false,

loadingMore: false,

hasMore: true,

// 筛选条件

currentTab: 'all',

searchKeyword: '',

statusFilter: '',

sortBy: 'created\_at',

sortOrder: 'desc',

// 标签页配置

tabs: [

{ key: 'all', name: '全部', count: 0 },

{ key: 'potential', name: '潜在客户', count: 0 },

{ key: 'following', name: '跟进中', count: 0 },

{ key: 'negotiating', name: '商务洽谈', count: 0 },

{ key: 'deal', name: '已成交', count: 0 },

{ key: 'lost', name: '已流失', count: 0 }

],

// 客户列表

customers: [],

page: 1,

pageSize: 20,

// 筛选面板

showFilterPanel: false,

filterOptions: {

industries: [

{ value: 'manufacturing', label: '制造业' },

{ value: 'commerce', label: '商贸业' },

{ value: 'service', label: '服务业' },

{ value: 'construction', label: '建筑业' },

{ value: 'other', label: '其他' }

],

scales: [

{ value: 'small', label: '小微企业' },

{ value: 'medium', label: '中型企业' },

{ value: 'large', label: '大型企业' }

]

},

selectedIndustry: '',

selectedScale: '',

// 客户状态配置

statusConfig: {

potential: { text: '潜在客户', color: '#909399' },

following: { text: '跟进中', color: '#E6A23C' },

negotiating: { text: '商务洽谈', color: '#409EFF' },

deal: { text: '已成交', color: '#67C23A' },

lost: { text: '已流失', color: '#F56C6C' }

}

},

onLoad(options) {

// 获取传入的状态参数

if (options.status) {

this.setData({

currentTab: options.status

})

}

this.loadCustomers(true)

this.loadStatistics()

},

onShow() {

// 页面显示时刷新数据

this.loadCustomers(true)

},

onPullDownRefresh() {

this.setData({

refreshing: true

})

this.loadCustomers(true).finally(() => {

this.setData({

refreshing: false

})

wx.stopPullDownRefresh()

})

},

onReachBottom() {

if (this.data.hasMore && !this.data.loadingMore) {

this.loadCustomers(false)

}

},

// 加载客户列表

async loadCustomers(refresh = false) {

const { currentTab, searchKeyword, selectedIndustry, selectedScale, sortBy, sortOrder, pageSize } = this.data

let { page } = this.data

if (refresh) {

page = 1

this.setData({

loading: page === 1,

customers: []

})

} else {

page += 1

this.setData({

loadingMore: true

})

}

try {

const params = {

page,

page\_size: pageSize,

sort\_by: sortBy,

sort\_order: sortOrder

}

// 添加筛选条件

if (currentTab !== 'all') {

params.status = currentTab

}

if (searchKeyword) {

params.keyword = searchKeyword

}

if (selectedIndustry) {

params.industry = selectedIndustry

}

if (selectedScale) {

params.scale = selectedScale

}

const result = await api.request('/manager/customers', {

method: 'GET',

data: params

})

const customers = result.data.items || []

const hasMore = customers.length === pageSize

// 处理客户数据

const processedCustomers = customers.map(customer => ({

...customer,

avatar\_url: customer.avatar\_url || '/images/default-company.png',

last\_contact\_time\_text: customer.last\_contact\_time ?

util.formatDate(customer.last\_contact\_time) : '暂无联系',

status\_config: this.data.statusConfig[customer.status] || { text: '未知', color: '#909399' },

contact\_count\_text: customer.contact\_count ? `已联系${customer.contact\_count}次` : '未联系',

order\_count\_text: customer.order\_count ? `${customer.order\_count}个订单` : '暂无订单',

total\_amount\_text: customer.total\_amount ? util.formatAmount(customer.total\_amount) : '￥0'

}))

this.setData({

customers: refresh ? processedCustomers : [...this.data.customers, ...processedCustomers],

page,

hasMore,

loading: false,

loadingMore: false

})

} catch (error) {

console.error('加载客户列表失败:', error)

wx.showToast({

title: '加载失败',

icon: 'none'

})

this.setData({

loading: false,

loadingMore: false

})

}

},

// 加载统计数据

async loadStatistics() {

try {

const result = await api.request('/manager/customers/statistics')

const stats = result.data || {}

// 更新标签页计数

const updatedTabs = this.data.tabs.map(tab => ({

...tab,

count: stats[tab.key] || 0

}))

this.setData({

tabs: updatedTabs

})

} catch (error) {

console.error('加载统计数据失败:', error)

}

},

// 标签页切换

onTabChange(e) {

const { key } = e.currentTarget.dataset

if (key === this.data.currentTab) return

this.setData({

currentTab: key

})

this.loadCustomers(true)

},

// 搜索输入

onSearchInput(e) {

const keyword = e.detail.value.trim()

this.setData({

searchKeyword: keyword

})

// 防抖搜索

clearTimeout(this.searchTimer)

this.searchTimer = setTimeout(() => {

this.loadCustomers(true)

}, 500)

},

// 搜索提交

onSearchSubmit() {

this.loadCustomers(true)

},

// 清除搜索

onSearchClear() {

this.setData({

searchKeyword: ''

})

this.loadCustomers(true)

},

// 显示筛选面板

onFilterTap() {

this.setData({

showFilterPanel: true

})

},

// 隐藏筛选面板

onFilterPanelClose() {

this.setData({

showFilterPanel: false

})

},

// 行业筛选

onIndustryChange(e) {

const industry = e.currentTarget.dataset.value

this.setData({

selectedIndustry: this.data.selectedIndustry === industry ? '' : industry

})

},

// 规模筛选

onScaleChange(e) {

const scale = e.currentTarget.dataset.value

this.setData({

selectedScale: this.data.selectedScale === scale ? '' : scale

})

},

// 重置筛选

onFilterReset() {

this.setData({

selectedIndustry: '',

selectedScale: '',

sortBy: 'created\_at',

sortOrder: 'desc'

})

},

// 应用筛选

onFilterApply() {

this.setData({

showFilterPanel: false

})

this.loadCustomers(true)

},

// 排序切换

onSortChange(e) {

const { sort } = e.currentTarget.dataset

let { sortBy, sortOrder } = this.data

if (sortBy === sort) {

sortOrder = sortOrder === 'desc' ? 'asc' : 'desc'

} else {

sortBy = sort

sortOrder = 'desc'

}

this.setData({

sortBy,

sortOrder

})

this.loadCustomers(true)

},

// 客户详情

onCustomerTap(e) {

const { customerId } = e.currentTarget.dataset

wx.navigateTo({

url: `/pages/manager/customer-detail/customer-detail?id=${customerId}`

})

},

// 拨打电话

onCallTap(e) {

e.stopPropagation()

const { phone } = e.currentTarget.dataset

if (!phone) {

wx.showToast({

title: '暂无电话号码',

icon: 'none'

})

return

}

wx.showActionSheet({

items: [

{

name: '拨打电话',

color: '#07C160'

},

{

name: '复制号码',

color: '#576B95'

}

],

success: (res) => {

if (res.tapIndex === 0) {

// 拨打电话

wx.makePhoneCall({

phoneNumber: phone,

fail: (err) => {

console.error('拨打电话失败:', err)

wx.showToast({

title: '拨打失败',

icon: 'none'

})

}

})

} else if (res.tapIndex === 1) {

// 复制号码

wx.setClipboardData({

data: phone,

success: () => {

wx.showToast({

title: '号码已复制',

icon: 'success'

})

}

})

}

}

})

},

// 微信聊天

onWechatTap(e) {

e.stopPropagation()

const { customerId, customerName } = e.currentTarget.dataset

wx.showModal({

title: '联系客户',

content: `确定要通过微信联系${customerName}吗？`,

confirmText: '确定',

cancelText: '取消',

success: (res) => {

if (res.confirm) {

// 这里可以跳转到聊天页面或记录联系记录

this.recordContact(customerId, 'wechat', '通过微信联系客户')

}

}

})

},

// 快速跟进

onQuickFollowTap(e) {

e.stopPropagation()

const { customerId, customerName } = e.currentTarget.dataset

wx.navigateTo({

url: `/pages/manager/follow/follow?customerId=${customerId}&customerName=${encodeURIComponent(customerName)}&type=quick`

})

},

// 记录联系记录

async recordContact(customerId, contactType, content) {

try {

await api.request('/manager/customers/contact-record', {

method: 'POST',

data: {

customer\_id: customerId,

contact\_type: contactType,

content: content,

contact\_time: new Date().toISOString()

}

})

wx.showToast({

title: '记录成功',

icon: 'success'

})

// 刷新客户列表

this.loadCustomers(true)

} catch (error) {

console.error('记录联系失败:', error)

wx.showToast({

title: '记录失败',

icon: 'none'

})

}

},

// 长按客户项

onCustomerLongPress(e) {

const { customerId, customerName, status } = e.currentTarget.dataset

const menuItems = ['查看详情', '添加跟进', '修改状态']

// 根据客户状态添加特定操作

if (status === 'potential') {

menuItems.push('转为跟进中')

} else if (status === 'following') {

menuItems.push('转为商务洽谈')

}

wx.showActionSheet({

itemList: menuItems,

success: (res) => {

const selectedItem = menuItems[res.tapIndex]

switch (selectedItem) {

case '查看详情':

this.onCustomerTap(e)

break

case '添加跟进':

this.onQuickFollowTap(e)

break

case '修改状态':

this.showStatusChangeModal(customerId, customerName, status)

break

case '转为跟进中':

this.changeCustomerStatus(customerId, 'following')

break

case '转为商务洽谈':

this.changeCustomerStatus(customerId, 'negotiating')

break

}

}

})

},

// 显示状态修改弹窗

showStatusChangeModal(customerId, customerName, currentStatus) {

const statusOptions = [

{ key: 'potential', name: '潜在客户' },

{ key: 'following', name: '跟进中' },

{ key: 'negotiating', name: '商务洽谈' },

{ key: 'deal', name: '已成交' },

{ key: 'lost', name: '已流失' }

]

const itemList = statusOptions

.filter(item => item.key !== currentStatus)

.map(item => item.name)

wx.showActionSheet({

itemList,

success: (res) => {

const selectedStatus = statusOptions

.filter(item => item.key !== currentStatus)[res.tapIndex]

wx.showModal({

title: '确认修改',

content: `确定将${customerName}的状态修改为"${selectedStatus.name}"吗？`,

success: (modalRes) => {

if (modalRes.confirm) {

this.changeCustomerStatus(customerId, selectedStatus.key)

}

}

})

}

})

},

// 修改客户状态

async changeCustomerStatus(customerId, newStatus) {

try {

await api.request(`/manager/customers/${customerId}/status`, {

method: 'PUT',

data: {

status: newStatus,

remark: `状态修改为${this.data.statusConfig[newStatus]?.text || newStatus}`

}

})

wx.showToast({

title: '状态已更新',

icon: 'success'

})

// 刷新数据

this.loadCustomers(true)

this.loadStatistics()

} catch (error) {

console.error('修改状态失败:', error)

wx.showToast({

title: '修改失败',

icon: 'none'

})

}

},

// 批量操作

onBatchOperationTap() {

// 这里可以实现批量操作功能

wx.showToast({

title: '功能开发中',

icon: 'none'

})

},

// 新增客户

onAddCustomerTap() {

wx.navigateTo({

url: '/pages/manager/customer-add/customer-add'

})

},

// 导出客户数据

onExportTap() {

wx.showLoading({

title: '导出中...'

})

api.request('/manager/customers/export', {

method: 'POST',

data: {

status: this.data.currentTab === 'all' ? '' : this.data.currentTab,

keyword: this.data.searchKeyword,

industry: this.data.selectedIndustry,

scale: this.data.selectedScale

}

}).then(result => {

wx.hideLoading()

if (result.data.download\_url) {

wx.showModal({

title: '导出成功',

content: '客户数据导出完成，是否立即下载？',

success: (res) => {

if (res.confirm) {

// 这里可以处理文件下载

wx.downloadFile({

url: result.data.download\_url,

success: (downloadRes) => {

wx.openDocument({

filePath: downloadRes.tempFilePath,

success: () => {

console.log('打开文档成功')

}

})

}

})

}

}

})

}

}).catch(error => {

wx.hideLoading()

console.error('导出失败:', error)

wx.showToast({

title: '导出失败',

icon: 'none'

})

})

},

// 页面卸载

onUnload() {

// 清除定时器

if (this.searchTimer) {

clearTimeout(this.searchTimer)

}

}

})

现在继续开发下一个模块：**我的客户页面模板 - pages/manager/customers/customers.wxml**

我的客户页面模板 - pages/manager/customers/customers.wxml

<!--pages/manager/customers/customers.wxml-->

<view class="customer-page">

<!-- 页面头部 -->

<view class="page-header">

<view class="header-title">我的客户</view>

<view class="header-actions">

<view class="action-btn" bindtap="onAddCustomerTap">

<text class="iconfont icon-add"></text>

<text class="action-text">新增</text>

</view>

<view class="action-btn" bindtap="onExportTap">

<text class="iconfont icon-export"></text>

<text class="action-text">导出</text>

</view>

</view>

</view>

<!-- 搜索栏 -->

<view class="search-section">

<view class="search-box">

<text class="iconfont icon-search search-icon"></text>

<input

class="search-input"

placeholder="搜索客户姓名、企业名称或手机号"

value="{{searchKeyword}}"

bindinput="onSearchInput"

bindconfirm="onSearchSubmit"

confirm-type="search"

/>

<view class="search-clear" wx:if="{{searchKeyword}}" bindtap="onSearchClear">

<text class="iconfont icon-close"></text>

</view>

</view>

<view class="filter-btn" bindtap="onFilterTap">

<text class="iconfont icon-filter"></text>

<text>筛选</text>

<view class="filter-dot" wx:if="{{selectedIndustry || selectedScale}}"></view>

</view>

</view>

<!-- 标签页 -->

<view class="tabs-container">

<scroll-view class="tabs-scroll" scroll-x enable-flex>

<view class="tabs-wrapper">

<view

class="tab-item {{currentTab === item.key ? 'active' : ''}}"

wx:for="{{tabs}}"

wx:key="key"

data-key="{{item.key}}"

bindtap="onTabChange"

>

<text class="tab-name">{{item.name}}</text>

<text class="tab-count" wx:if="{{item.count > 0}}">({{item.count}})</text>

</view>

</view>

</scroll-view>

</view>

<!-- 排序和筛选条件显示 -->

<view class="filter-display" wx:if="{{selectedIndustry || selectedScale || sortBy !== 'created\_at'}}">

<view class="filter-tags">

<view class="filter-tag" wx:if="{{selectedIndustry}}">

<text>{{selectedIndustry}}</text>

<text class="tag-close" data-type="industry" bindtap="onFilterTagClose">×</text>

</view>

<view class="filter-tag" wx:if="{{selectedScale}}">

<text>{{selectedScale}}</text>

<text class="tag-close" data-type="scale" bindtap="onFilterTagClose">×</text>

</view>

</view>

<view class="sort-display">

<text class="sort-text">按{{sortBy === 'created\_at' ? '注册时间' : sortBy === 'last\_contact\_time' ? '最后联系时间' : '客户价值'}}排序</text>

<text class="sort-order">{{sortOrder === 'desc' ? '↓' : '↑'}}</text>

</view>

</view>

<!-- 客户列表 -->

<view class="customer-list">

<!-- 加载状态 -->

<view class="loading-container" wx:if="{{loading}}">

<view class="loading-spinner"></view>

<text class="loading-text">加载中...</text>

</view>

<!-- 客户项 -->

<view

class="customer-item"

wx:for="{{customers}}"

wx:key="id"

data-customer-id="{{item.id}}"

data-customer-name="{{item.name}}"

data-status="{{item.status}}"

bindtap="onCustomerTap"

bindlongpress="onCustomerLongPress"

>

<!-- 客户头像和基本信息 -->

<view class="customer-avatar">

<image

class="avatar-img"

src="{{item.avatar\_url}}"

mode="aspectFill"

lazy-load

/>

<view class="status-dot {{item.status}}"></view>

</view>

<view class="customer-info">

<!-- 第一行：姓名和状态 -->

<view class="info-row">

<view class="customer-name">{{item.name}}</view>

<view class="customer-status" style="color: {{item.status\_config.color}}">

{{item.status\_config.text}}

</view>

</view>

<!-- 第二行：企业名称 -->

<view class="info-row">

<view class="company-name">{{item.company\_name || '未填写企业'}}</view>

<view class="customer-value">{{item.total\_amount\_text}}</view>

</view>

<!-- 第三行：联系信息 -->

<view class="info-row contact-info">

<view class="contact-item">

<text class="iconfont icon-phone"></text>

<text class="contact-text">{{item.phone || '未绑定手机'}}</text>

</view>

<view class="contact-item">

<text class="iconfont icon-time"></text>

<text class="contact-text">{{item.last\_contact\_time\_text}}</text>

</view>

</view>

<!-- 第四行：统计信息 -->

<view class="info-row stats-info">

<view class="stat-item">

<text class="stat-label">联系:</text>

<text class="stat-value">{{item.contact\_count || 0}}次</text>

</view>

<view class="stat-item">

<text class="stat-label">订单:</text>

<text class="stat-value">{{item.order\_count || 0}}个</text>

</view>

<view class="stat-item">

<text class="stat-label">注册:</text>

<text class="stat-value">{{item.created\_at\_text}}</text>

</view>

</view>

</view>

<!-- 快捷操作按钮 -->

<view class="customer-actions">

<view

class="action-btn call-btn"

data-phone="{{item.phone}}"

bindtap="onCallTap"

wx:if="{{item.phone}}"

>

<text class="iconfont icon-phone"></text>

</view>

<view

class="action-btn wechat-btn"

data-customer-id="{{item.id}}"

data-customer-name="{{item.name}}"

bindtap="onWechatTap"

>

<text class="iconfont icon-wechat"></text>

</view>

<view

class="action-btn follow-btn"

data-customer-id="{{item.id}}"

data-customer-name="{{item.name}}"

bindtap="onQuickFollowTap"

>

<text class="iconfont icon-edit"></text>

</view>

</view>

</view>

<!-- 空状态 -->

<view class="empty-state" wx:if="{{!loading && customers.length === 0}}">

<image class="empty-image" src="/images/empty-customer.png" />

<text class="empty-text">暂无客户数据</text>

<text class="empty-desc">快去开发新客户吧！</text>

<button class="empty-btn" bindtap="onAddCustomerTap">立即添加</button>

</view>

<!-- 加载更多 -->

<view class="load-more" wx:if="{{!loading && customers.length > 0}}">

<view class="load-more-text" wx:if="{{loadingMore}}">

<view class="loading-spinner small"></view>

<text>加载中...</text>

</view>

<text class="load-more-text" wx:elif="{{!hasMore}}">没有更多数据了</text>

</view>

</view>

<!-- 筛选面板 -->

<view class="filter-panel {{showFilterPanel ? 'show' : ''}}" catchtap="onFilterPanelClose">

<view class="filter-content" catchtap="">

<view class="filter-header">

<text class="filter-title">筛选条件</text>

<view class="filter-close" bindtap="onFilterPanelClose">

<text class="iconfont icon-close"></text>

</view>

</view>

<view class="filter-body">

<!-- 行业筛选 -->

<view class="filter-group">

<view class="filter-group-title">行业类型</view>

<view class="filter-options">

<view

class="filter-option {{selectedIndustry === item.value ? 'active' : ''}}"

wx:for="{{filterOptions.industries}}"

wx:key="value"

data-value="{{item.value}}"

bindtap="onIndustryChange"

>

{{item.label}}

</view>

</view>

</view>

<!-- 规模筛选 -->

<view class="filter-group">

<view class="filter-group-title">企业规模</view>

<view class="filter-options">

<view

class="filter-option {{selectedScale === item.value ? 'active' : ''}}"

wx:for="{{filterOptions.scales}}"

wx:key="value"

data-value="{{item.value}}"

bindtap="onScaleChange"

>

{{item.label}}

</view>

</view>

</view>

<!-- 排序选项 -->

<view class="filter-group">

<view class="filter-group-title">排序方式</view>

<view class="sort-options">

<view

class="sort-option {{sortBy === 'created\_at' ? 'active' : ''}}"

data-sort="created\_at"

bindtap="onSortChange"

>

<text>注册时间</text>

<text class="sort-arrow" wx:if="{{sortBy === 'created\_at'}}">

{{sortOrder === 'desc' ? '↓' : '↑'}}

</text>

</view>

<view

class="sort-option {{sortBy === 'last\_contact\_time' ? 'active' : ''}}"

data-sort="last\_contact\_time"

bindtap="onSortChange"

>

<text>最后联系时间</text>

<text class="sort-arrow" wx:if="{{sortBy === 'last\_contact\_time'}}">

{{sortOrder === 'desc' ? '↓' : '↑'}}

</text>

</view>

<view

class="sort-option {{sortBy === 'total\_amount' ? 'active' : ''}}"

data-sort="total\_amount"

bindtap="onSortChange"

>

<text>客户价值</text>

<text class="sort-arrow" wx:if="{{sortBy === 'total\_amount'}}">

{{sortOrder === 'desc' ? '↓' : '↑'}}

</text>

</view>

</view>

</view>

</view>

<view class="filter-footer">

<button class="reset-btn" bindtap="onFilterReset">重置</button>

<button class="apply-btn" bindtap="onFilterApply">应用</button>

</view>

</view>

</view>

<!-- 浮动操作按钮 -->

<view class="fab-container" wx:if="{{customers.length > 0}}">

<view class="fab-btn" bindtap="onBatchOperationTap">

<text class="iconfont icon-menu"></text>

</view>

</view>

</view>

继续开发下一个模块：**我的客户页面样式文件 - pages/manager/customers/customers.wxss**

我的客户页面样式文件 - pages/manager/customers/customers.wxss

/\* pages/manager/customers/customers.wxss \*/

.customer-page {

background-color: #f5f5f5;

min-height: 100vh;

padding-bottom: 100rpx;

}

/\* 页面头部 \*/

.page-header {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

padding: 30rpx 30rpx 40rpx;

display: flex;

justify-content: space-between;

align-items: center;

}

.header-title {

color: #fff;

font-size: 36rpx;

font-weight: 600;

}

.header-actions {

display: flex;

gap: 20rpx;

}

.action-btn {

display: flex;

flex-direction: column;

align-items: center;

gap: 8rpx;

padding: 10rpx;

color: #fff;

font-size: 20rpx;

opacity: 0.9;

transition: all 0.3s;

}

.action-btn:active {

transform: scale(0.95);

opacity: 0.7;

}

.action-text {

font-size: 20rpx;

}

/\* 搜索栏 \*/

.search-section {

background: #fff;

padding: 30rpx;

display: flex;

gap: 20rpx;

box-shadow: 0 2rpx 10rpx rgba(0, 0, 0, 0.05);

}

.search-box {

flex: 1;

position: relative;

background: #f8f8f8;

border-radius: 50rpx;

padding: 20rpx 30rpx;

display: flex;

align-items: center;

gap: 15rpx;

}

.search-icon {

font-size: 32rpx;

color: #999;

}

.search-input {

flex: 1;

font-size: 28rpx;

color: #333;

}

.search-clear {

padding: 10rpx;

color: #999;

font-size: 24rpx;

}

.filter-btn {

position: relative;

background: #667eea;

color: #fff;

padding: 20rpx 30rpx;

border-radius: 50rpx;

display: flex;

align-items: center;

gap: 10rpx;

font-size: 26rpx;

transition: all 0.3s;

}

.filter-btn:active {

transform: scale(0.95);

background: #5a6fd8;

}

.filter-dot {

position: absolute;

top: 10rpx;

right: 10rpx;

width: 12rpx;

height: 12rpx;

background: #ff4757;

border-radius: 50%;

}

/\* 标签页 \*/

.tabs-container {

background: #fff;

padding: 0 30rpx;

border-bottom: 1rpx solid #f0f0f0;

}

.tabs-scroll {

white-space: nowrap;

}

.tabs-wrapper {

display: flex;

gap: 40rpx;

}

.tab-item {

position: relative;

padding: 30rpx 0;

display: flex;

align-items: center;

gap: 8rpx;

font-size: 28rpx;

color: #666;

transition: all 0.3s;

white-space: nowrap;

}

.tab-item.active {

color: #667eea;

font-weight: 600;

}

.tab-item.active::after {

content: '';

position: absolute;

bottom: 0;

left: 0;

right: 0;

height: 4rpx;

background: #667eea;

border-radius: 2rpx;

}

.tab-count {

font-size: 20rpx;

color: #999;

background: #f0f0f0;

padding: 4rpx 8rpx;

border-radius: 10rpx;

min-width: 30rpx;

text-align: center;

}

.tab-item.active .tab-count {

background: #667eea;

color: #fff;

}

/\* 筛选条件显示 \*/

.filter-display {

background: #fff;

padding: 20rpx 30rpx;

display: flex;

justify-content: space-between;

align-items: center;

border-bottom: 1rpx solid #f0f0f0;

}

.filter-tags {

display: flex;

gap: 15rpx;

}

.filter-tag {

background: #667eea;

color: #fff;

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 24rpx;

display: flex;

align-items: center;

gap: 10rpx;

}

.tag-close {

font-size: 28rpx;

opacity: 0.8;

}

.sort-display {

display: flex;

align-items: center;

gap: 10rpx;

font-size: 24rpx;

color: #666;

}

.sort-order {

font-size: 20rpx;

color: #667eea;

}

/\* 客户列表 \*/

.customer-list {

background: #fff;

margin-top: 20rpx;

}

.customer-item {

position: relative;

padding: 30rpx;

border-bottom: 1rpx solid #f0f0f0;

display: flex;

gap: 20rpx;

align-items: flex-start;

transition: all 0.3s;

}

.customer-item:active {

background: #f8f8f8;

}

.customer-item:last-child {

border-bottom: none;

}

/\* 客户头像 \*/

.customer-avatar {

position: relative;

flex-shrink: 0;

}

.avatar-img {

width: 80rpx;

height: 80rpx;

border-radius: 50%;

background: #f0f0f0;

}

.status-dot {

position: absolute;

bottom: 0;

right: 0;

width: 20rpx;

height: 20rpx;

border-radius: 50%;

border: 2rpx solid #fff;

}

.status-dot.potential {

background: #909399;

}

.status-dot.following {

background: #E6A23C;

}

.status-dot.negotiating {

background: #409EFF;

}

.status-dot.deal {

background: #67C23A;

}

.status-dot.lost {

background: #F56C6C;

}

/\* 客户信息 \*/

.customer-info {

flex: 1;

min-width: 0;

}

.info-row {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 10rpx;

}

.info-row:last-child {

margin-bottom: 0;

}

.customer-name {

font-size: 32rpx;

font-weight: 600;

color: #333;

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap;

flex: 1;

}

.customer-status {

font-size: 22rpx;

padding: 6rpx 12rpx;

border-radius: 15rpx;

background: rgba(0, 0, 0, 0.05);

white-space: nowrap;

}

.customer-status.potential {

background: rgba(144, 147, 153, 0.1);

color: #909399;

}

.customer-status.following {

background: rgba(230, 162, 60, 0.1);

color: #E6A23C;

}

.customer-status.negotiating {

background: rgba(64, 158, 255, 0.1);

color: #409EFF;

}

.customer-status.deal {

background: rgba(103, 194, 58, 0.1);

color: #67C23A;

}

.customer-status.lost {

background: rgba(245, 108, 108, 0.1);

color: #F56C6C;

}

/\* 客户详情信息 \*/

.customer-company {

font-size: 24rpx;

color: #666;

margin-bottom: 8rpx;

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap;

}

.customer-phone {

font-size: 24rpx;

color: #999;

display: flex;

align-items: center;

gap: 8rpx;

margin-bottom: 8rpx;

}

.customer-capacity {

font-size: 24rpx;

color: #666;

display: flex;

align-items: center;

gap: 8rpx;

}

.capacity-value {

font-weight: 600;

color: #333;

}

/\* 操作按钮区域 \*/

.customer-actions {

display: flex;

flex-direction: column;

gap: 15rpx;

align-items: flex-end;

min-width: 120rpx;

}

.action-button {

padding: 12rpx 20rpx;

border-radius: 20rpx;

font-size: 22rpx;

text-align: center;

min-width: 80rpx;

transition: all 0.3s;

}

.action-button:active {

transform: scale(0.95);

}

.call-btn {

background: #67C23A;

color: #fff;

}

.call-btn:active {

background: #5daf34;

}

.follow-btn {

background: #409EFF;

color: #fff;

}

.follow-btn:active {

background: #3a8ee6;

}

.assign-btn {

background: #E6A23C;

color: #fff;

}

.assign-btn:active {

background: #cf9236;

}

/\* 客户标签 \*/

.customer-tags {

display: flex;

gap: 10rpx;

margin-top: 15rpx;

flex-wrap: wrap;

}

.customer-tag {

font-size: 18rpx;

padding: 4rpx 8rpx;

border-radius: 8rpx;

background: #f0f0f0;

color: #666;

}

.tag-vip {

background: #ffeaa7;

color: #d63031;

}

.tag-potential {

background: #e1f5fe;

color: #0277bd;

}

.tag-high-value {

background: #f3e5f5;

color: #7b1fa2;

}

/\* 最后跟进信息 \*/

.last-follow {

margin-top: 15rpx;

padding-top: 15rpx;

border-top: 1rpx solid #f0f0f0;

font-size: 22rpx;

color: #999;

display: flex;

justify-content: space-between;

align-items: center;

}

.follow-time {

font-size: 20rpx;

}

.follow-content {

flex: 1;

margin: 0 15rpx;

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap;

}

/\* 空状态 \*/

.empty-state {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 100rpx 30rpx;

color: #999;

}

.empty-icon {

font-size: 100rpx;

color: #e0e0e0;

margin-bottom: 30rpx;

}

.empty-text {

font-size: 28rpx;

margin-bottom: 40rpx;

}

.empty-action {

background: #667eea;

color: #fff;

padding: 20rpx 40rpx;

border-radius: 50rpx;

font-size: 26rpx;

}

/\* 加载状态 \*/

.loading-more {

display: flex;

align-items: center;

justify-content: center;

padding: 40rpx;

color: #999;

font-size: 24rpx;

gap: 15rpx;

}

.loading-icon {

width: 30rpx;

height: 30rpx;

border: 3rpx solid #f0f0f0;

border-top: 3rpx solid #667eea;

border-radius: 50%;

animation: spin 1s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

/\* 筛选弹窗 \*/

.filter-popup {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.5);

z-index: 1000;

display: flex;

align-items: flex-end;

}

.filter-content {

background: #fff;

border-radius: 20rpx 20rpx 0 0;

padding: 40rpx 30rpx 80rpx;

width: 100%;

max-height: 80vh;

overflow-y: auto;

}

.filter-title {

font-size: 32rpx;

font-weight: 600;

color: #333;

text-align: center;

margin-bottom: 40rpx;

}

.filter-section {

margin-bottom: 40rpx;

}

.filter-section-title {

font-size: 28rpx;

font-weight: 600;

color: #333;

margin-bottom: 20rpx;

}

.filter-options {

display: flex;

flex-wrap: wrap;

gap: 15rpx;

}

.filter-option {

padding: 15rpx 25rpx;

border: 2rpx solid #e0e0e0;

border-radius: 50rpx;

font-size: 24rpx;

color: #666;

transition: all 0.3s;

}

.filter-option.active {

border-color: #667eea;

background: #667eea;

color: #fff;

}

.filter-actions {

display: flex;

gap: 20rpx;

margin-top: 40rpx;

}

.filter-reset {

flex: 1;

padding: 25rpx;

border: 2rpx solid #e0e0e0;

border-radius: 50rpx;

text-align: center;

font-size: 28rpx;

color: #666;

background: #fff;

}

.filter-confirm {

flex: 2;

padding: 25rpx;

background: #667eea;

color: #fff;

border-radius: 50rpx;

text-align: center;

font-size: 28rpx;

}

/\* 响应式适配 \*/

@media screen and (max-width: 320px) {

.customer-item {

padding: 20rpx;

}

.customer-name {

font-size: 28rpx;

}

.action-button {

padding: 8rpx 15rpx;

font-size: 20rpx;

min-width: 70rpx;

}

}

/\* 深色模式适配 \*/

@media (prefers-color-scheme: dark) {

.customer-page {

background-color: #1a1a1a;

}

.customer-list {

background: #2d2d2d;

}

.customer-item {

border-bottom-color: #3d3d3d;

}

.customer-item:active {

background: #3d3d3d;

}

.customer-name {

color: #fff;

}

.customer-company,

.customer-capacity {

color: #ccc;

}

.customer-phone {

color: #999;

}

}

现在继续开发下一个模块：**我的客户页面配置文件 - pages/manager/customers/customers.json**

我的客户页面配置文件 - pages/manager/customers/customers.json

{

"navigationBarTitleText": "我的客户",

"navigationBarBackgroundColor": "#667eea",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f5f5",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"usingComponents": {

"van-search": "@vant/weapp/search/index",

"van-tabs": "@vant/weapp/tabs/index",

"van-tab": "@vant/weapp/tab/index",

"van-tag": "@vant/weapp/tag/index",

"van-empty": "@vant/weapp/empty/index",

"van-loading": "@vant/weapp/loading/index",

"van-popup": "@vant/weapp/popup/index",

"van-button": "@vant/weapp/button/index",

"van-icon": "@vant/weapp/icon/index",

"van-cell": "@vant/weapp/cell/index",

"van-cell-group": "@vant/weapp/cell-group/index",

"van-radio": "@vant/weapp/radio/index",

"van-radio-group": "@vant/weapp/radio-group/index",

"van-checkbox": "@vant/weapp/checkbox/index",

"van-checkbox-group": "@vant/weapp/checkbox-group/index"

},

"style": "v2",

"singlePage": {

"navigationBarFit": "squeezed"

}

}

接下来开发：**客户详情页面逻辑 - pages/manager/customers/detail/detail.js**

客户详情页面逻辑 - pages/manager/customers/detail/detail.js

// pages/manager/customers/detail/detail.js

const api = require('../../../../utils/api')

const util = require('../../../../utils/util')

Page({

data: {

customerId: '',

customerInfo: {},

followRecords: [],

orders: [],

contracts: [],

tabs: [

{ key: 'info', title: '基本信息' },

{ key: 'follow', title: '跟进记录' },

{ key: 'orders', title: '订单记录' },

{ key: 'contracts', title: '合同记录' }

],

activeTab: 'info',

loading: false,

refreshing: false,

// 跟进记录相关

followPage: 1,

followHasMore: true,

followLoading: false,

// 订单记录相关

orderPage: 1,

orderHasMore: true,

orderLoading: false,

// 添加跟进记录弹窗

showFollowDialog: false,

followForm: {

type: 'phone',

content: '',

nextFollowDate: '',

remark: ''

},

followTypes: [

{ value: 'phone', label: '电话跟进' },

{ value: 'visit', label: '上门拜访' },

{ value: 'wechat', label: '微信沟通' },

{ value: 'email', label: '邮件联系' },

{ value: 'meeting', label: '会议洽谈' },

{ value: 'other', label: '其他方式' }

],

// 客户状态更新

showStatusDialog: false,

statusOptions: [

{ value: 'potential', label: '潜在客户', color: '#909399' },

{ value: 'following', label: '跟进中', color: '#E6A23C' },

{ value: 'negotiating', label: '洽谈中', color: '#409EFF' },

{ value: 'deal', label: '已成交', color: '#67C23A' },

{ value: 'lost', label: '已流失', color: '#F56C6C' }

],

selectedStatus: '',

// 标签管理

showTagDialog: false,

availableTags: [],

selectedTags: [],

// 分配客户经理

showAssignDialog: false,

employees: [],

selectedEmployee: ''

},

onLoad(options) {

console.log('客户详情页面加载', options)

if (options.id) {

this.setData({

customerId: options.id

})

this.loadCustomerDetail()

}

},

onShow() {

// 页面显示时刷新数据

if (this.data.customerId) {

this.loadCustomerDetail()

}

},

onPullDownRefresh() {

this.setData({ refreshing: true })

this.loadCustomerDetail().finally(() => {

this.setData({ refreshing: false })

wx.stopPullDownRefresh()

})

},

onReachBottom() {

const { activeTab } = this.data

if (activeTab === 'follow') {

this.loadMoreFollowRecords()

} else if (activeTab === 'orders') {

this.loadMoreOrders()

}

},

// 加载客户详情

async loadCustomerDetail() {

this.setData({ loading: true })

try {

const customerInfo = await api.getCustomerDetail(this.data.customerId)

this.setData({ customerInfo })

// 根据当前激活的标签页加载对应数据

await this.loadTabData(this.data.activeTab)

} catch (error) {

console.error('加载客户详情失败:', error)

wx.showToast({

title: '加载失败',

icon: 'none'

})

} finally {

this.setData({ loading: false })

}

},

// 加载标签页数据

async loadTabData(tab) {

switch (tab) {

case 'follow':

await this.loadFollowRecords()

break

case 'orders':

await this.loadOrders()

break

case 'contracts':

await this.loadContracts()

break

}

},

// 加载跟进记录

async loadFollowRecords(reset = true) {

if (this.data.followLoading) return

const page = reset ? 1 : this.data.followPage

this.setData({ followLoading: true })

try {

const response = await api.getCustomerFollowRecords(this.data.customerId, {

page,

limit: 10

})

const records = response.data || []

const followRecords = reset ? records : [...this.data.followRecords, ...records]

this.setData({

followRecords,

followPage: page + 1,

followHasMore: records.length >= 10

})

} catch (error) {

console.error('加载跟进记录失败:', error)

} finally {

this.setData({ followLoading: false })

}

},

// 加载更多跟进记录

loadMoreFollowRecords() {

if (this.data.followHasMore && !this.data.followLoading) {

this.loadFollowRecords(false)

}

},

// 加载订单记录

async loadOrders(reset = true) {

if (this.data.orderLoading) return

const page = reset ? 1 : this.data.orderPage

this.setData({ orderLoading: true })

try {

const response = await api.getCustomerOrders(this.data.customerId, {

page,

limit: 10

})

const orderList = response.data || []

const orders = reset ? orderList : [...this.data.orders, ...orderList]

this.setData({

orders,

orderPage: page + 1,

orderHasMore: orderList.length >= 10

})

} catch (error) {

console.error('加载订单记录失败:', error)

} finally {

this.setData({ orderLoading: false })

}

},

// 加载更多订单

loadMoreOrders() {

if (this.data.orderHasMore && !this.data.orderLoading) {

this.loadOrders(false)

}

},

// 加载合同记录

async loadContracts() {

try {

const contracts = await api.getCustomerContracts(this.data.customerId)

this.setData({ contracts })

} catch (error) {

console.error('加载合同记录失败:', error)

}

},

// 标签页切换

onTabChange(e) {

const activeTab = e.detail.name

this.setData({ activeTab })

this.loadTabData(activeTab)

},

// 拨打电话

onCallCustomer() {

const phone = this.data.customerInfo.phone

if (phone) {

wx.makePhoneCall({

phoneNumber: phone,

success: () => {

console.log('拨打电话成功')

},

fail: (error) => {

console.error('拨打电话失败:', error)

}

})

}

},

// 发送微信消息

onSendMessage() {

wx.showToast({

title: '功能开发中',

icon: 'none'

})

},

// 显示添加跟进记录弹窗

showAddFollowDialog() {

this.setData({

showFollowDialog: true,

followForm: {

type: 'phone',

content: '',

nextFollowDate: '',

remark: ''

}

})

},

// 关闭跟进记录弹窗

hideFollowDialog() {

this.setData({ showFollowDialog: false })

},

// 跟进类型选择

onFollowTypeChange(e) {

this.setData({

'followForm.type': e.detail.value

})

},

// 跟进内容输入

onFollowContentInput(e) {

this.setData({

'followForm.content': e.detail.value

})

},

// 下次跟进时间选择

onNextFollowDateChange(e) {

this.setData({

'followForm.nextFollowDate': e.detail.value

})

},

// 跟进备注输入

onFollowRemarkInput(e) {

this.setData({

'followForm.remark': e.detail.value

})

},

// 提交跟进记录

async submitFollowRecord() {

const { followForm } = this.data

if (!followForm.content.trim()) {

wx.showToast({

title: '请输入跟进内容',

icon: 'none'

})

return

}

try {

await api.addFollowRecord({

customerId: this.data.customerId,

...followForm

})

wx.showToast({

title: '添加成功',

icon: 'success'

})

this.hideFollowDialog()

this.loadFollowRecords()

} catch (error) {

console.error('添加跟进记录失败:', error)

wx.showToast({

title: '添加失败',

icon: 'none'

})

}

},

// 显示状态更新弹窗

showUpdateStatusDialog() {

this.setData({

showStatusDialog: true,

selectedStatus: this.data.customerInfo.status || 'potential'

})

},

// 关闭状态更新弹窗

hideStatusDialog() {

this.setData({ showStatusDialog: false })

},

// 状态选择

onStatusChange(e) {

this.setData({

selectedStatus: e.detail.value

})

},

// 更新客户状态

async updateCustomerStatus() {

const { selectedStatus } = this.data

try {

await api.updateCustomerStatus(this.data.customerId, selectedStatus)

wx.showToast({

title: '更新成功',

icon: 'success'

})

// 更新本地数据

this.setData({

'customerInfo.status': selectedStatus,

showStatusDialog: false

})

} catch (error) {

console.error('更新客户状态失败:', error)

wx.showToast({

title: '更新失败',

icon: 'none'

})

}

},

// 显示标签管理弹窗

async showTagManageDialog() {

try {

const availableTags = await api.getAvailableTags()

const customerTags = this.data.customerInfo.tags || []

this.setData({

showTagDialog: true,

availableTags,

selectedTags: customerTags.map(tag => tag.id)

})

} catch (error) {

console.error('加载标签失败:', error)

}

},

// 关闭标签管理弹窗

hideTagDialog() {

this.setData({ showTagDialog: false })

},

// 标签选择

onTagChange(e) {

this.setData({

selectedTags: e.detail.value

})

},

// 更新客户标签

async updateCustomerTags() {

const { selectedTags } = this.data

try {

await api.updateCustomerTags(this.data.customerId, selectedTags)

wx.showToast({

title: '更新成功',

icon: 'success'

})

this.hideTagDialog()

this.loadCustomerDetail()

} catch (error) {

console.error('更新标签失败:', error)

wx.showToast({

title: '更新失败',

icon: 'none'

})

}

},

// 显示分配客户经理弹窗

async showAssignEmployeeDialog() {

try {

const employees = await api.getEmployeeList()

this.setData({

showAssignDialog: true,

employees,

selectedEmployee: this.data.customerInfo.assigned\_employee\_id || ''

})

} catch (error) {

console.error('加载员工列表失败:', error)

}

},

// 关闭分配弹窗

hideAssignDialog() {

this.setData({ showAssignDialog: false })

},

// 员工选择

onEmployeeChange(e) {

this.setData({

selectedEmployee: e.detail.value

})

},

// 分配客户经理

async assignCustomer() {

const { selectedEmployee } = this.data

if (!selectedEmployee) {

wx.showToast({

title: '请选择客户经理',

icon: 'none'

})

return

}

try {

await api.assignCustomer(this.data.customerId, selectedEmployee)

wx.showToast({

title: '分配成功',

icon: 'success'

})

this.hideAssignDialog()

this.loadCustomerDetail()

} catch (error) {

console.error('分配客户失败:', error)

wx.showToast({

title: '分配失败',

icon: 'none'

})

}

},

// 查看订单详情

viewOrderDetail(e) {

const orderId = e.currentTarget.dataset.orderId

wx.navigateTo({

url: `/pages/orders/detail/detail?id=${orderId}`

})

},

// 查看合同详情

viewContractDetail(e) {

const contractId = e.currentTarget.dataset.contractId

wx.navigateTo({

url: `/pages/profile/contracts/detail/detail?id=${contractId}`

})

},

// 编辑客户信息

editCustomerInfo() {

wx.navigateTo({

url: `/pages/manager/customers/edit/edit?id=${this.data.customerId}`

})

},

// 格式化时间

formatTime(time) {

return util.formatTime(new Date(time))

},

// 格式化跟进类型

formatFollowType(type) {

const typeMap = {

'phone': '电话跟进',

'visit': '上门拜访',

'wechat': '微信沟通',

'email': '邮件联系',

'meeting': '会议洽谈',

'other': '其他方式'

}

return typeMap[type] || type

},

// 格式化状态

formatStatus(status) {

const statusMap = {

'potential': '潜在客户',

'following': '跟进中',

'negotiating': '洽谈中',

'deal': '已成交',

'lost': '已流失'

}

return statusMap[status] || status

},

// 格式化订单状态

formatOrderStatus(status) {

const statusMap = {

'pending': '待确认',

'confirmed': '已确认',

'processing': '处理中',

'completed': '已完成',

'cancelled': '已取消'

}

return statusMap[status] || status

}

})

现在这些模块都已完成：

1. **我的客户页面样式文件补充** (CSS) - 完成了剩余的样式定义
2. **我的客户页面配置文件** (JSON) - 页面配置和组件引用
3. **客户详情页面逻辑** (JavaScript) - 完整的客户详情功能

接下来继续开发下一个模块：**客户详情页面模板 - pages/manager/customers/detail/detail.wxml**

客户详情页面模板 - pages/manager/customers/detail/detail.wxml

<!-- pages/manager/customers/detail/detail.wxml -->

<view class="customer-detail-page">

<!-- 页面头部 -->

<view class="page-header">

<view class="customer-basic">

<view class="customer-avatar">

<image class="avatar-img" src="{{customerInfo.avatar\_url || '/images/default-avatar.png'}}" mode="aspectFill" />

<view class="status-dot {{customerInfo.status}}"></view>

</view>

<view class="basic-info">

<view class="customer-name">{{customerInfo.name || '未设置'}}</view>

<view class="customer-company">{{customerInfo.company\_name || '未设置'}}</view>

<view class="customer-phone">

<text class="phone-text">{{customerInfo.phone}}</text>

<text class="copy-btn" bindtap="copyPhone" data-phone="{{customerInfo.phone}}">复制</text>

</view>

</view>

</view>

<!-- 快捷操作 -->

<view class="quick-actions">

<view class="action-btn" bindtap="makePhoneCall">

<image class="action-icon" src="/images/icons/phone.png" />

<text class="action-text">拨打电话</text>

</view>

<view class="action-btn" bindtap="sendMessage">

<image class="action-icon" src="/images/icons/message.png" />

<text class="action-text">发送消息</text>

</view>

<view class="action-btn" bindtap="addFollowRecord">

<image class="action-icon" src="/images/icons/follow.png" />

<text class="action-text">添加跟进</text>

</view>

</view>

</view>

<!-- 标签页切换 -->

<view class="tab-container">

<view class="tab-list">

<view class="tab-item {{currentTab === 0 ? 'active' : ''}}" bindtap="switchTab" data-index="0">

基本信息

</view>

<view class="tab-item {{currentTab === 1 ? 'active' : ''}}" bindtap="switchTab" data-index="1">

跟进记录

</view>

<view class="tab-item {{currentTab === 2 ? 'active' : ''}}" bindtap="switchTab" data-index="2">

订单记录

</view>

<view class="tab-item {{currentTab === 3 ? 'active' : ''}}" bindtap="switchTab" data-index="3">

用电信息

</view>

</view>

</view>

<!-- 标签页内容 -->

<view class="tab-content">

<!-- 基本信息标签页 -->

<view class="tab-panel {{currentTab === 0 ? 'active' : ''}}">

<view class="info-section">

<view class="section-title">客户信息</view>

<view class="info-list">

<view class="info-item">

<text class="info-label">姓名</text>

<text class="info-value">{{customerInfo.name || '未设置'}}</text>

</view>

<view class="info-item">

<text class="info-label">企业名称</text>

<text class="info-value">{{customerInfo.company\_name || '未设置'}}</text>

</view>

<view class="info-item">

<text class="info-label">联系电话</text>

<text class="info-value">{{customerInfo.phone}}</text>

</view>

<view class="info-item">

<text class="info-label">注册时间</text>

<text class="info-value">{{customerInfo.created\_at}}</text>

</view>

<view class="info-item">

<text class="info-label">认证状态</text>

<text class="info-value status-{{customerInfo.auth\_status}}">

{{customerInfo.auth\_status === 'verified' ? '已认证' : customerInfo.auth\_status === 'pending' ? '待认证' : '未认证'}}

</text>

</view>

<view class="info-item">

<text class="info-label">客户标签</text>

<view class="customer-tags">

<view class="tag-item" wx:for="{{customerInfo.tags}}" wx:key="index">

{{item}}

</view>

<view class="tag-add" bindtap="addTag">+</view>

</view>

</view>

</view>

</view>

<view class="info-section">

<view class="section-title">归属信息</view>

<view class="info-list">

<view class="info-item">

<text class="info-label">客户经理</text>

<text class="info-value">{{customerInfo.manager\_name || '未分配'}}</text>

</view>

<view class="info-item">

<text class="info-label">分配时间</text>

<text class="info-value">{{customerInfo.assigned\_at || '未分配'}}</text>

</view>

<view class="info-item">

<text class="info-label">所属部门</text>

<text class="info-value">{{customerInfo.department\_name || '未分配'}}</text>

</view>

</view>

</view>

</view>

<!-- 跟进记录标签页 -->

<view class="tab-panel {{currentTab === 1 ? 'active' : ''}}">

<view class="follow-list">

<view class="follow-item" wx:for="{{followRecords}}" wx:key="id">

<view class="follow-header">

<view class="follow-type">{{item.follow\_type}}</view>

<view class="follow-time">{{item.created\_at}}</view>

</view>

<view class="follow-content">{{item.content}}</view>

<view class="follow-footer">

<text class="follower">{{item.employee\_name}}</text>

<view class="next-follow" wx:if="{{item.next\_follow\_date}}">

下次跟进：{{item.next\_follow\_date}}

</view>

</view>

</view>

<view class="empty-state" wx:if="{{followRecords.length === 0}}">

<image class="empty-icon" src="/images/empty-follow.png" />

<text class="empty-text">暂无跟进记录</text>

</view>

</view>

<!-- 添加跟进按钮 -->

<view class="add-follow-btn" bindtap="addFollowRecord">

<image class="add-icon" src="/images/icons/add.png" />

<text>添加跟进记录</text>

</view>

</view>

<!-- 订单记录标签页 -->

<view class="tab-panel {{currentTab === 2 ? 'active' : ''}}">

<view class="order-list">

<view class="order-item" wx:for="{{orderRecords}}" wx:key="id" bindtap="viewOrderDetail" data-id="{{item.id}}">

<view class="order-header">

<view class="order-number">订单号：{{item.order\_number}}</view>

<view class="order-status status-{{item.status}}">{{item.status\_text}}</view>

</view>

<view class="order-info">

<view class="product-name">{{item.product\_name}}</view>

<view class="order-amount">¥{{item.amount}}</view>

</view>

<view class="order-time">{{item.created\_at}}</view>

</view>

<view class="empty-state" wx:if="{{orderRecords.length === 0}}">

<image class="empty-icon" src="/images/empty-order.png" />

<text class="empty-text">暂无订单记录</text>

</view>

</view>

</view>

<!-- 用电信息标签页 -->

<view class="tab-panel {{currentTab === 3 ? 'active' : ''}}">

<view class="power-section">

<view class="section-title">用电基本信息</view>

<view class="info-list">

<view class="info-item">

<text class="info-label">用电容量</text>

<text class="info-value">{{powerInfo.capacity || '未设置'}} kVA</text>

</view>

<view class="info-item">

<text class="info-label">月均用电量</text>

<text class="info-value">{{powerInfo.monthly\_usage || '未设置'}} kWh</text>

</view>

<view class="info-item">

<text class="info-label">当前电价</text>

<text class="info-value">{{powerInfo.current\_price || '未设置'}} 元/kWh</text>

</view>

<view class="info-item">

<text class="info-label">行业类型</text>

<text class="info-value">{{powerInfo.industry\_type || '未设置'}}</text>

</view>

<view class="info-item">

<text class="info-label">用电模式</text>

<text class="info-value">{{powerInfo.usage\_pattern || '未设置'}}</text>

</view>

</view>

</view>

<view class="power-section">

<view class="section-title">用电量趋势</view>

<view class="chart-container">

<canvas class="usage-chart" canvas-id="usageChart"></canvas>

</view>

</view>

<view class="power-section">

<view class="section-title">节省潜力分析</view>

<view class="saving-analysis">

<view class="saving-item">

<text class="saving-label">预估年节省金额</text>

<text class="saving-value highlight">¥{{savingAnalysis.yearly\_saving || 0}}</text>

</view>

<view class="saving-item">

<text class="saving-label">节省比例</text>

<text class="saving-value">{{savingAnalysis.saving\_rate || 0}}%</text>

</view>

<view class="saving-item">

<text class="saving-label">推荐产品</text>

<text class="saving-value">{{savingAnalysis.recommended\_product || '暂无'}}</text>

</view>

</view>

</view>

</view>

</view>

</view>

<!-- 添加跟进记录弹窗 -->

<view class="modal {{showFollowModal ? 'show' : ''}}" catchtap="hideFollowModal">

<view class="modal-content" catchtap="stopPropagation">

<view class="modal-header">

<text class="modal-title">添加跟进记录</text>

<image class="close-btn" src="/images/icons/close.png" bindtap="hideFollowModal" />

</view>

<view class="modal-body">

<view class="form-group">

<text class="form-label">跟进方式</text>

<picker class="form-picker" range="{{followTypes}}" range-key="text" value="{{followForm.type\_index}}" bindchange="onFollowTypeChange">

<view class="picker-text">{{followTypes[followForm.type\_index].text}}</view>

</picker>

</view>

<view class="form-group">

<text class="form-label">跟进内容</text>

<textarea class="form-textarea" placeholder="请输入跟进内容" value="{{followForm.content}}" bindinput="onFollowContentChange" maxlength="500"></textarea>

</view>

<view class="form-group">

<text class="form-label">下次跟进时间</text>

<picker class="form-picker" mode="date" value="{{followForm.next\_date}}" bindchange="onNextDateChange">

<view class="picker-text">{{followForm.next\_date || '选择日期'}}</view>

</picker>

</view>

</view>

<view class="modal-footer">

<button class="cancel-btn" bindtap="hideFollowModal">取消</button>

<button class="confirm-btn" bindtap="submitFollowRecord" disabled="{{submitting}}">

{{submitting ? '提交中...' : '确认'}}

</button>

</view>

</view>

</view>

<!-- 添加标签弹窗 -->

<view class="modal {{showTagModal ? 'show' : ''}}" catchtap="hideTagModal">

<view class="modal-content" catchtap="stopPropagation">

<view class="modal-header">

<text class="modal-title">添加客户标签</text>

<image class="close-btn" src="/images/icons/close.png" bindtap="hideTagModal" />

</view>

<view class="modal-body">

<view class="tag-input-group">

<input class="tag-input" placeholder="输入标签名称" value="{{newTag}}" bindinput="onTagInput" maxlength="10" />

<button class="add-tag-btn" bindtap="confirmAddTag" disabled="{{!newTag}}">添加</button>

</view>

<view class="common-tags">

<text class="section-title">常用标签</text>

<view class="tag-list">

<view class="common-tag" wx:for="{{commonTags}}" wx:key="index" bindtap="selectCommonTag" data-tag="{{item}}">

{{item}}

</view>

</view>

</view>

</view>

</view>

</view>

现在继续开发下一个模块：**客户详情页面样式文件 - pages/manager/customers/detail/detail.wxss**

客户详情页面样式文件 - detail.wxss

/\* pages/manager/customers/detail/detail.wxss \*/

.customer-detail-page {

background-color: #f5f5f5;

min-height: 100vh;

}

/\* 页面头部 \*/

.page-header {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

padding: 30rpx;

color: white;

}

.customer-basic {

display: flex;

align-items: center;

margin-bottom: 30rpx;

}

.customer-avatar {

position: relative;

margin-right: 30rpx;

}

.avatar-img {

width: 120rpx;

height: 120rpx;

border-radius: 60rpx;

border: 4rpx solid rgba(255, 255, 255, 0.3);

}

.status-dot {

position: absolute;

bottom: 0;

right: 0;

width: 24rpx;

height: 24rpx;

border-radius: 12rpx;

border: 3rpx solid white;

}

.status-dot.active {

background-color: #52c41a;

}

.status-dot.inactive {

background-color: #f5222d;

}

.status-dot.pending {

background-color: #faad14;

}

.basic-info {

flex: 1;

}

.customer-name {

font-size: 36rpx;

font-weight: bold;

margin-bottom: 10rpx;

}

.customer-company {

font-size: 28rpx;

opacity: 0.9;

margin-bottom: 10rpx;

}

.customer-phone {

font-size: 26rpx;

display: flex;

align-items: center;

}

.phone-text {

margin-right: 20rpx;

}

.copy-btn {

background-color: rgba(255, 255, 255, 0.2);

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 24rpx;

}

/\* 快捷操作 \*/

.quick-actions {

display: flex;

justify-content: space-around;

}

.action-btn {

display: flex;

flex-direction: column;

align-items: center;

padding: 20rpx;

border-radius: 16rpx;

background-color: rgba(255, 255, 255, 0.1);

backdrop-filter: blur(10rpx);

}

.action-icon {

width: 48rpx;

height: 48rpx;

margin-bottom: 10rpx;

}

.action-text {

font-size: 24rpx;

}

/\* 标签页 \*/

.tab-container {

background-color: white;

padding: 0 30rpx;

border-bottom: 1rpx solid #eee;

}

.tab-list {

display: flex;

}

.tab-item {

flex: 1;

text-align: center;

padding: 30rpx 0;

font-size: 28rpx;

color: #666;

border-bottom: 4rpx solid transparent;

transition: all 0.3s;

}

.tab-item.active {

color: #1890ff;

border-bottom-color: #1890ff;

font-weight: bold;

}

/\* 标签页内容 \*/

.tab-content {

flex: 1;

overflow: hidden;

}

.tab-panel {

display: none;

padding: 30rpx;

}

.tab-panel.active {

display: block;

}

/\* 信息区块 \*/

.info-section {

background-color: white;

border-radius: 16rpx;

padding: 30rpx;

margin-bottom: 20rpx;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.section-title {

font-size: 32rpx;

font-weight: bold;

color: #333;

margin-bottom: 20rpx;

border-left: 6rpx solid #1890ff;

padding-left: 16rpx;

}

.info-list {

/\* 信息列表样式 \*/

}

.info-item {

display: flex;

justify-content: space-between;

align-items: center;

padding: 20rpx 0;

border-bottom: 1rpx solid #f0f0f0;

}

.info-item:last-child {

border-bottom: none;

}

.info-label {

font-size: 28rpx;

color: #666;

min-width: 160rpx;

}

.info-value {

font-size: 28rpx;

color: #333;

flex: 1;

text-align: right;

}

.info-value.status-verified {

color: #52c41a;

}

.info-value.status-pending {

color: #faad14;

}

.info-value.status-unverified {

color: #f5222d;

}

/\* 客户标签 \*/

.customer-tags {

display: flex;

flex-wrap: wrap;

align-items: center;

gap: 10rpx;

}

.tag-item {

background-color: #e6f7ff;

color: #1890ff;

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 24rpx;

border: 1rpx solid #91d5ff;

}

.tag-add {

background-color: #f0f0f0;

color: #666;

padding: 8rpx 12rpx;

border-radius: 20rpx;

font-size: 24rpx;

border: 1rpx dashed #d9d9d9;

}

/\* 跟进记录 \*/

.follow-list {

/\* 跟进记录列表样式 \*/

}

.follow-item {

background-color: white;

border-radius: 16rpx;

padding: 30rpx;

margin-bottom: 20rpx;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.follow-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 15rpx;

}

.follow-type {

background-color: #e6f7ff;

color: #1890ff;

padding: 6rpx 12rpx;

border-radius: 12rpx;

font-size: 24rpx;

}

.follow-time {

font-size: 24rpx;

color: #999;

}

.follow-content {

font-size: 28rpx;

color: #333;

line-height: 1.6;

margin-bottom: 15rpx;

}

.follow-footer {

display: flex;

justify-content: space-between;

align-items: center;

padding-top: 15rpx;

border-top: 1rpx solid #f0f0f0;

}

.follower {

font-size: 24rpx;

color: #666;

}

.next-follow {

font-size: 24rpx;

color: #1890ff;

}

/\* 添加跟进按钮 \*/

.add-follow-btn {

position: fixed;

bottom: 30rpx;

right: 30rpx;

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

color: white;

padding: 20rpx 30rpx;

border-radius: 50rpx;

box-shadow: 0 4rpx 16rpx rgba(102, 126, 234, 0.4);

display: flex;

align-items: center;

font-size: 28rpx;

}

.add-icon {

width: 32rpx;

height: 32rpx;

margin-right: 10rpx;

}

/\* 订单记录 \*/

.order-list {

/\* 订单列表样式 \*/

}

.order-item {

background-color: white;

border-radius: 16rpx;

padding: 30rpx;

margin-bottom: 20rpx;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.order-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 15rpx;

}

.order-number {

font-size: 28rpx;

color: #333;

}

.order-status {

padding: 6rpx 12rpx;

border-radius: 12rpx;

font-size: 24rpx;

}

.order-status.status-pending {

background-color: #fff7e6;

color: #faad14;

}

.order-status.status-processing {

background-color: #e6f7ff;

color: #1890ff;

}

.order-status.status-completed {

background-color: #f6ffed;

color: #52c41a;

}

.order-status.status-cancelled {

background-color: #fff2f0;

color: #f5222d;

}

.order-info {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 15rpx;

}

.product-name {

font-size: 30rpx;

color: #333;

font-weight: 500;

}

.order-amount {

font-size: 32rpx;

color: #f5222d;

font-weight: bold;

}

.order-time {

font-size: 24rpx;

color: #999;

text-align: right;

}

/\* 用电信息 \*/

.power-section {

background-color: white;

border-radius: 16rpx;

padding: 30rpx;

margin-bottom: 20rpx;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.chart-container {

height: 400rpx;

display: flex;

align-items: center;

justify-content: center;

}

.usage-chart {

width: 100%;

height: 100%;

}

.saving-analysis {

/\* 节省分析样式 \*/

}

.saving-item {

display: flex;

justify-content: space-between;

align-items: center;

padding: 20rpx 0;

border-bottom: 1rpx solid #f0f0f0;

}

.saving-item:last-child {

border-bottom: none;

}

.saving-label {

font-size: 28rpx;

color: #666;

}

.saving-value {

font-size: 28rpx;

color: #333;

font-weight: 500;

}

.saving-value.highlight {

color: #f5222d;

font-size: 32rpx;

font-weight: bold;

}

/\* 空状态 \*/

.empty-state {

display: flex;

flex-direction: column;

align-items: center;

padding: 100rpx 0;

color: #999;

}

.empty-icon {

width: 120rpx;

height: 120rpx;

opacity: 0.5;

margin-bottom: 20rpx;

}

.empty-text {

font-size: 28rpx;

}

/\* 弹窗样式 \*/

.modal {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

display: flex;

align-items: center;

justify-content: center;

opacity: 0;

visibility: hidden;

transition: all 0.3s;

z-index: 1000;

}

.modal.show {

opacity: 1;

visibility: visible;

}

.modal-content {

background-color: white;

border-radius: 20rpx;

width: 90%;

max-width: 600rpx;

max-height: 80%;

overflow: hidden;

transform: scale(0.8);

transition: transform 0.3s;

}

.modal.show .modal-content {

transform: scale(1);

}

.modal-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 30rpx;

border-bottom: 1rpx solid #f0f0f0;

}

.modal-title {

font-size: 32rpx;

font-weight: bold;

color: #333;

}

.close-btn {

width: 32rpx;

height: 32rpx;

}

.modal-body {

padding: 30rpx;

max-height: 500rpx;

overflow-y: auto;

}

.modal-footer {

display: flex;

padding: 20rpx 30rpx;

border-top: 1rpx solid #f0f0f0;

gap: 20rpx;

}

.cancel-btn {

flex: 1;

background-color: #f5f5f5;

color: #666;

border: none;

border-radius: 8rpx;

padding: 20rpx;

font-size: 28rpx;

}

.confirm-btn {

flex: 1;

background-color: #1890ff;

color: white;

border: none;

border-radius: 8rpx;

padding: 20rpx;

font-size: 28rpx;

}

.confirm-btn[disabled] {

background-color: #d9d9d9;

color: #aaa;

}

/\* 表单样式 \*/

.form-group {

margin-bottom: 30rpx;

}

.form-label {

display: block;

font-size: 28rpx;

color: #333;

margin-bottom: 15rpx;

}

.form-picker {

width: 100%;

background-color: #f5f5f5;

border-radius: 8rpx;

padding: 20rpx;

}

.picker-text {

font-size: 28rpx;

color: #333;

}

.form-textarea {

width: 100%;

min-height: 200rpx;

background-color: #f5f5f5;

border-radius: 8rpx;

padding: 20rpx;

font-size: 28rpx;

border: none;

}

/\* 标签输入 \*/

.tag-input-group {

display: flex;

gap: 20rpx;

align-items: center;

margin-bottom: 30rpx;

}

.tag-input {

flex: 1;

background-color: #f5f5f5;

border-radius: 8rpx;

padding: 20rpx;

font-size: 28rpx;

border: none;

}

.add-tag-btn {

background-color: #1890ff;

color: white;

border: none;

border-radius: 8rpx;

padding: 20rpx 30rpx;

font-size: 26rpx;

}

.add-tag-btn[disabled] {

background-color: #d9d9d9;

color: #aaa;

}

/\* 常见标签 \*/

.common-tags {

margin-top: 20rpx;

}

.common-tags-title {

font-size: 26rpx;

color: #666;

margin-bottom: 15rpx;

}

.common-tag-item {

display: inline-block;

background-color: #f0f0f0;

color: #333;

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 24rpx;

margin-right: 15rpx;

margin-bottom: 10rpx;

border: 1rpx solid #d9d9d9;

transition: all 0.2s;

}

.common-tag-item:active {

background-color: #e6f7ff;

color: #1890ff;

border-color: #91d5ff;

}

/\* 提醒设置 \*/

.reminder-section {

background-color: white;

border-radius: 16rpx;

padding: 30rpx;

margin-bottom: 20rpx;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.reminder-item {

display: flex;

justify-content: space-between;

align-items: center;

padding: 20rpx 0;

border-bottom: 1rpx solid #f0f0f0;

}

.reminder-item:last-child {

border-bottom: none;

}

.reminder-label {

font-size: 28rpx;

color: #333;

}

.reminder-switch {

transform: scale(0.8);

}

.reminder-time {

font-size: 24rpx;

color: #666;

margin-top: 5rpx;

}

/\* 加载状态 \*/

.loading-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 100rpx 0;

}

.loading-icon {

width: 60rpx;

height: 60rpx;

border: 4rpx solid #f0f0f0;

border-top: 4rpx solid #1890ff;

border-radius: 50%;

animation: spin 1s linear infinite;

}

.loading-text {

font-size: 28rpx;

color: #666;

margin-top: 20rpx;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

/\* 错误状态 \*/

.error-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 100rpx 0;

}

.error-icon {

width: 120rpx;

height: 120rpx;

opacity: 0.5;

margin-bottom: 20rpx;

}

.error-text {

font-size: 28rpx;

color: #999;

margin-bottom: 30rpx;

}

.retry-btn {

background-color: #1890ff;

color: white;

padding: 15rpx 30rpx;

border-radius: 8rpx;

font-size: 26rpx;

}

/\* 操作按钮组 \*/

.action-buttons {

display: flex;

gap: 20rpx;

padding: 30rpx;

background-color: white;

border-top: 1rpx solid #f0f0f0;

}

.action-button {

flex: 1;

padding: 20rpx;

border-radius: 8rpx;

font-size: 28rpx;

text-align: center;

}

.action-button.primary {

background-color: #1890ff;

color: white;

}

.action-button.secondary {

background-color: #f5f5f5;

color: #666;

}

.action-button.danger {

background-color: #f5222d;

color: white;

}

/\* 统计卡片 \*/

.stats-cards {

display: flex;

gap: 20rpx;

padding: 30rpx;

background-color: #f5f5f5;

}

.stat-card {

flex: 1;

background-color: white;

border-radius: 16rpx;

padding: 30rpx 20rpx;

text-align: center;

box-shadow: 0 2rpx 12rpx rgba(0, 0, 0, 0.1);

}

.stat-number {

font-size: 36rpx;

font-weight: bold;

color: #1890ff;

margin-bottom: 10rpx;

}

.stat-label {

font-size: 24rpx;

color: #666;

}

/\* 时间线样式 \*/

.timeline {

position: relative;

padding: 20rpx 0;

}

.timeline-item {

position: relative;

padding-left: 60rpx;

padding-bottom: 40rpx;

}

.timeline-item:last-child {

padding-bottom: 0;

}

.timeline-item::before {

content: '';

position: absolute;

left: 20rpx;

top: 0;

bottom: -40rpx;

width: 2rpx;

background-color: #f0f0f0;

}

.timeline-item:last-child::before {

display: none;

}

.timeline-dot {

position: absolute;

left: 12rpx;

top: 10rpx;

width: 16rpx;

height: 16rpx;

border-radius: 50%;

background-color: #1890ff;

border: 3rpx solid white;

box-shadow: 0 0 0 2rpx #1890ff;

}

.timeline-content {

background-color: #f8f9fa;

border-radius: 12rpx;

padding: 20rpx;

}

.timeline-time {

font-size: 24rpx;

color: #999;

margin-bottom: 10rpx;

}

.timeline-text {

font-size: 28rpx;

color: #333;

line-height: 1.5;

}

/\* 响应式适配 \*/

@media (max-width: 750rpx) {

.customer-basic {

flex-direction: column;

align-items: flex-start;

}

.customer-avatar {

margin-right: 0;

margin-bottom: 20rpx;

}

.quick-actions {

flex-wrap: wrap;

}

.action-btn {

min-width: 160rpx;

margin-bottom: 20rpx;

}

.stats-cards {

flex-direction: column;

}

.action-buttons {

flex-direction: column;

}

}

/\* 深色模式支持 \*/

@media (prefers-color-scheme: dark) {

.customer-detail-page {

background-color: #1a1a1a;

}

.info-section {

background-color: #2d2d2d;

color: #fff;

}

.section-title {

color: #fff;

}

.info-label {

color: #ccc;

}

.info-value {

color: #fff;

}

}

/\* 打印样式 \*/

@media print {

.add-follow-btn,

.action-buttons,

.quick-actions {

display: none !important;

}

.customer-detail-page {

background-color: white !important;

}

.info-section {

box-shadow: none !important;

border: 1rpx solid #ddd !important;

}

}

现在继续开发下一个模块：**客户详情页面配置文件 - pages/manager/customers/detail/detail.json**

客户详情页面配置文件 - detail.json

{

"navigationBarTitleText": "客户详情",

"navigationBarBackgroundColor": "#667eea",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f5f5",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"usingComponents": {

"van-popup": "@vant/weapp/popup/index",

"van-overlay": "@vant/weapp/overlay/index",

"van-picker": "@vant/weapp/picker/index",

"van-datetime-picker": "@vant/weapp/datetime-picker/index",

"van-field": "@vant/weapp/field/index",

"van-button": "@vant/weapp/button/index",

"van-tag": "@vant/weapp/tag/index",

"van-switch": "@vant/weapp/switch/index",

"van-toast": "@vant/weapp/toast/index",

"van-dialog": "@vant/weapp/dialog/index",

"van-loading": "@vant/weapp/loading/index",

"van-empty": "@vant/weapp/empty/index",

"van-divider": "@vant/weapp/divider/index"

},

"disableScroll": false

}

接下来开发：**跟进管理页面逻辑 - pages/manager/follow/follow.js**

跟进管理页面逻辑 - pages/manager/follow/follow.js

// pages/manager/follow/follow.js

const api = require('../../../utils/api')

const util = require('../../../utils/util')

Page({

data: {

// 页面状态

loading: false,

refreshing: false,

loadingMore: false,

hasMore: true,

// 当前标签页

activeTab: 0,

tabs: ['待跟进', '已跟进', '预约跟进', '逾期未跟进'],

// 跟进列表数据

followList: [],

pagination: {

page: 1,

pageSize: 10,

total: 0

},

// 筛选条件

filterParams: {

status: '', // 跟进状态

priority: '', // 优先级

dateRange: '', // 时间范围

keyword: '' // 搜索关键词

},

// 筛选选项

priorityOptions: [

{ value: 'high', text: '高优先级', color: '#f5222d' },

{ value: 'medium', text: '中优先级', color: '#faad14' },

{ value: 'low', text: '低优先级', color: '#52c41a' }

],

dateRangeOptions: [

{ value: 'today', text: '今天' },

{ value: 'tomorrow', text: '明天' },

{ value: 'week', text: '本周' },

{ value: 'month', text: '本月' },

{ value: 'overdue', text: '已逾期' }

],

// 搜索

showSearch: false,

searchKeyword: '',

// 筛选面板

showFilter: false,

tempFilterParams: {},

// 批量操作

showBatchActions: false,

selectedItems: [],

selectMode: false,

// 跟进详情弹窗

showFollowDetail: false,

currentFollow: null,

// 添加跟进弹窗

showAddFollow: false,

followForm: {

customerId: '',

customerName: '',

type: 'call',

content: '',

nextFollowDate: '',

priority: 'medium',

reminder: false

},

followTypes: [

{ value: 'call', text: '电话沟通', icon: 'phone' },

{ value: 'visit', text: '拜访客户', icon: 'location' },

{ value: 'email', text: '邮件联系', icon: 'email' },

{ value: 'wechat', text: '微信沟通', icon: 'wechat' },

{ value: 'meeting', text: '会议洽谈', icon: 'meeting' }

],

// 统计数据

statistics: {

todayFollow: 0,

overdueFollow: 0,

weeklyTarget: 0,

completionRate: 0

}

},

onLoad(options) {

// 从参数中获取初始状态

if (options.tab) {

this.setData({ activeTab: parseInt(options.tab) })

}

if (options.customerId) {

this.setData({

'followForm.customerId': options.customerId,

'followForm.customerName': options.customerName || ''

})

if (options.action === 'add') {

this.setData({ showAddFollow: true })

}

}

this.loadStatistics()

this.loadFollowList()

},

onShow() {

// 页面显示时刷新数据

if (this.data.followList.length > 0) {

this.refreshFollowList()

}

},

onPullDownRefresh() {

this.refreshFollowList()

},

onReachBottom() {

if (!this.data.loadingMore && this.data.hasMore) {

this.loadMoreFollowList()

}

},

// 切换标签页

onTabChange(e) {

const index = e.currentTarget.dataset.index

if (index !== this.data.activeTab) {

this.setData({

activeTab: index,

'pagination.page': 1,

followList: []

})

this.updateFilterByTab(index)

this.loadFollowList()

}

},

// 根据标签页更新筛选条件

updateFilterByTab(tabIndex) {

const statusMap = {

0: 'pending', // 待跟进

1: 'completed', // 已跟进

2: 'scheduled', // 预约跟进

3: 'overdue' // 逾期未跟进

}

this.setData({

'filterParams.status': statusMap[tabIndex] || ''

})

},

// 加载跟进统计

async loadStatistics() {

try {

const result = await api.getFollowStatistics()

if (result.success) {

this.setData({ statistics: result.data })

}

} catch (error) {

console.error('加载统计数据失败:', error)

}

},

// 加载跟进列表

async loadFollowList(showLoading = true) {

if (showLoading) {

this.setData({ loading: true })

}

try {

const params = {

page: this.data.pagination.page,

pageSize: this.data.pagination.pageSize,

...this.data.filterParams

}

const result = await api.getFollowList(params)

if (result.success) {

const newList = this.data.pagination.page === 1

? result.data.list

: [...this.data.followList, ...result.data.list]

this.setData({

followList: newList,

'pagination.total': result.data.total,

hasMore: newList.length < result.data.total

})

}

} catch (error) {

console.error('加载跟进列表失败:', error)

wx.showToast({

title: '加载失败',

icon: 'none'

})

} finally {

this.setData({ loading: false, refreshing: false, loadingMore: false })

wx.stopPullDownRefresh()

}

},

// 刷新跟进列表

refreshFollowList() {

this.setData({

refreshing: true,

'pagination.page': 1,

followList: []

})

this.loadFollowList(false)

},

// 加载更多

loadMoreFollowList() {

this.setData({

loadingMore: true,

'pagination.page': this.data.pagination.page + 1

})

this.loadFollowList(false)

},

// 显示/隐藏搜索

toggleSearch() {

this.setData({

showSearch: !this.data.showSearch,

searchKeyword: ''

})

if (!this.data.showSearch && this.data.filterParams.keyword) {

this.setData({

'filterParams.keyword': ''

})

this.refreshFollowList()

}

},

// 搜索输入

onSearchInput(e) {

this.setData({

searchKeyword: e.detail.value

})

},

// 搜索确认

onSearchConfirm() {

this.setData({

'filterParams.keyword': this.data.searchKeyword.trim()

})

this.refreshFollowList()

},

// 显示筛选面板

showFilterPanel() {

this.setData({

showFilter: true,

tempFilterParams: { ...this.data.filterParams }

})

},

// 筛选条件改变

onFilterChange(e) {

const { field, value } = e.currentTarget.dataset

this.setData({

[`tempFilterParams.${field}`]: value

})

},

// 重置筛选

resetFilter() {

const defaultFilter = {

status: this.data.filterParams.status, // 保持当前标签页的状态

priority: '',

dateRange: '',

keyword: ''

}

this.setData({

tempFilterParams: defaultFilter

})

},

// 应用筛选

applyFilter() {

this.setData({

filterParams: { ...this.data.tempFilterParams },

showFilter: false

})

this.refreshFollowList()

},

// 关闭筛选面板

closeFilter() {

this.setData({ showFilter: false })

},

// 跟进项点击

onFollowItemTap(e) {

const follow = e.currentTarget.dataset.follow

if (this.data.selectMode) {

this.toggleSelectItem(follow.id)

} else {

this.showFollowDetail(follow)

}

},

// 显示跟进详情

showFollowDetail(follow) {

this.setData({

currentFollow: follow,

showFollowDetail: true

})

},

// 关闭跟进详情

closeFollowDetail() {

this.setData({

showFollowDetail: false,

currentFollow: null

})

},

// 快速操作

onQuickAction(e) {

const { action, follow } = e.currentTarget.dataset

switch (action) {

case 'call':

this.makePhoneCall(follow.customer.phone)

break

case 'complete':

this.completeFollow(follow.id)

break

case 'reschedule':

this.rescheduleFollow(follow)

break

case 'detail':

this.goToCustomerDetail(follow.customer.id)

break

}

},

// 拨打电话

makePhoneCall(phoneNumber) {

wx.makePhoneCall({

phoneNumber: phoneNumber,

fail: (error) => {

wx.showToast({

title: '拨号失败',

icon: 'none'

})

}

})

},

// 完成跟进

async completeFollow(followId) {

try {

wx.showLoading({ title: '处理中...' })

const result = await api.updateFollowStatus({

id: followId,

status: 'completed',

completedAt: new Date().toISOString()

})

if (result.success) {

wx.showToast({

title: '操作成功',

icon: 'success'

})

this.refreshFollowList()

this.loadStatistics()

}

} catch (error) {

console.error('完成跟进失败:', error)

wx.showToast({

title: '操作失败',

icon: 'none'

})

} finally {

wx.hideLoading()

}

},

// 重新安排跟进

rescheduleFollow(follow) {

this.setData({

followForm: {

id: follow.id,

customerId: follow.customer.id,

customerName: follow.customer.name,

type: follow.type,

content: follow.content,

nextFollowDate: follow.nextFollowDate,

priority: follow.priority,

reminder: follow.reminder

},

showAddFollow: true

})

},

// 进入客户详情

goToCustomerDetail(customerId) {

wx.navigateTo({

url: `/pages/manager/customers/detail/detail?id=${customerId}`

})

},

// 显示添加跟进

showAddFollowDialog() {

const tomorrow = new Date()

tomorrow.setDate(tomorrow.getDate() + 1)

this.setData({

followForm: {

customerId: '',

customerName: '',

type: 'call',

content: '',

nextFollowDate: util.formatDate(tomorrow, 'YYYY-MM-DD'),

priority: 'medium',

reminder: true

},

showAddFollow: true

})

},

// 跟进表单输入

onFollowFormInput(e) {

const { field } = e.currentTarget.dataset

const { value } = e.detail

this.setData({

[`followForm.${field}`]: value

})

},

// 选择跟进类型

onFollowTypeChange(e) {

this.setData({

'followForm.type': e.detail.value

})

},

// 选择优先级

onPriorityChange(e) {

this.setData({

'followForm.priority': e.detail.value

})

},

// 选择日期

onDateChange(e) {

this.setData({

'followForm.nextFollowDate': e.detail.value

})

},

// 提醒开关

onReminderChange(e) {

this.setData({

'followForm.reminder': e.detail.value

})

},

// 选择客户

selectCustomer() {

wx.navigateTo({

url: '/pages/manager/customers/select/select'

})

},

// 保存跟进

async saveFollow() {

const form = this.data.followForm

// 表单验证

if (!form.customerId) {

wx.showToast({

title: '请选择客户',

icon: 'none'

})

return

}

if (!form.content.trim()) {

wx.showToast({

title: '请填写跟进内容',

icon: 'none'

})

return

}

try {

wx.showLoading({ title: '保存中...' })

const result = form.id

? await api.updateFollow(form.id, form)

: await api.createFollow(form)

if (result.success) {

wx.showToast({

title: '保存成功',

icon: 'success'

})

this.setData({ showAddFollow: false })

this.refreshFollowList()

this.loadStatistics()

}

} catch (error) {

console.error('保存跟进失败:', error)

wx.showToast({

title: '保存失败',

icon: 'none'

})

} finally {

wx.hideLoading()

}

},

// 关闭添加跟进

closeAddFollow() {

this.setData({ showAddFollow: false })

},

// 切换选择模式

toggleSelectMode() {

this.setData({

selectMode: !this.data.selectMode,

selectedItems: [],

showBatchActions: false

})

},

// 切换选择项

toggleSelectItem(itemId) {

const selectedItems = [...this.data.selectedItems]

const index = selectedItems.indexOf(itemId)

if (index > -1) {

selectedItems.splice(index, 1)

} else {

selectedItems.push(itemId)

}

this.setData({

selectedItems,

showBatchActions: selectedItems.length > 0

})

},

// 全选/取消全选

toggleSelectAll() {

const allSelected = this.data.selectedItems.length === this.data.followList.length

const selectedItems = allSelected ? [] : this.data.followList.map(item => item.id)

this.setData({

selectedItems,

showBatchActions: selectedItems.length > 0

})

},

// 批量操作

async batchAction(e) {

const { action } = e.currentTarget.dataset

const selectedIds = this.data.selectedItems

if (selectedIds.length === 0) {

wx.showToast({

title: '请选择项目',

icon: 'none'

})

return

}

let title = ''

let apiMethod = null

switch (action) {

case 'complete':

title = '批量完成跟进'

apiMethod = api.batchCompleteFollow

break

case 'delete':

title = '批量删除跟进'

apiMethod = api.batchDeleteFollow

break

default:

return

}

wx.showModal({

title: '确认操作',

content: `确定要${title}吗？`,

success: async (res) => {

if (res.confirm) {

try {

wx.showLoading({ title: '处理中...' })

const result = await apiMethod(selectedIds)

if (result.success) {

wx.showToast({

title: '操作成功',

icon: 'success'

})

this.setData({

selectMode: false,

selectedItems: [],

showBatchActions: false

})

this.refreshFollowList()

this.loadStatistics()

}

} catch (error) {

console.error('批量操作失败:', error)

wx.showToast({

title: '操作失败',

icon: 'none'

})

} finally {

wx.hideLoading()

}

}

}

})

},

// 格式化日期显示

formatFollowDate(dateStr) {

return util.formatRelativeTime(dateStr)

},

// 获取优先级样式

getPriorityStyle(priority) {

const styles = {

high: { color: '#f5222d', bgColor: '#fff2f0' },

medium: { color: '#faad14', bgColor: '#fffbe6' },

low: { color: '#52c41a', bgColor: '#f6ffed' }

}

return styles[priority] || styles.medium

},

// 获取跟进类型图标

getFollowTypeIcon(type) {

const icons = {

call: '/images/icons/phone.png',

visit: '/images/icons/location.png',

email: '/images/icons/email.png',

wechat: '/images/icons/wechat.png',

meeting: '/images/icons/meeting.png'

}

return icons[type] || '/images/icons/phone.png'

}

})

继续开发：**跟进管理页面模板 - pages/manager/follow/follow.wxml**

跟进管理页面模板 - pages/manager/follow/follow.wxml

<!--pages/manager/follow/follow.wxml-->

<view class="container">

<!-- 顶部筛选栏 -->

<view class="filter-bar">

<view class="filter-item">

<picker

bindchange="onStatusChange"

value="{{statusIndex}}"

range="{{statusList}}"

range-key="label"

>

<view class="picker-text">

{{statusList[statusIndex].label}}

<text class="iconfont icon-arrow-down"></text>

</view>

</picker>

</view>

<view class="filter-item">

<picker

mode="date"

value="{{filterDate}}"

bindchange="onDateChange"

>

<view class="picker-text">

{{filterDate || '选择日期'}}

<text class="iconfont icon-calendar"></text>

</view>

</picker>

</view>

<view class="filter-item search-item">

<input

class="search-input"

placeholder="搜索客户姓名"

value="{{searchKeyword}}"

bindinput="onSearchInput"

confirm-type="search"

bindconfirm="onSearch"

/>

<text class="iconfont icon-search" bindtap="onSearch"></text>

</view>

</view>

<!-- 统计信息 -->

<view class="stats-bar">

<view class="stat-item">

<text class="stat-number">{{stats.todayFollow}}</text>

<text class="stat-label">今日跟进</text>

</view>

<view class="stat-item">

<text class="stat-number">{{stats.weekFollow}}</text>

<text class="stat-label">本周跟进</text>

</view>

<view class="stat-item">

<text class="stat-number">{{stats.pendingFollow}}</text>

<text class="stat-label">待跟进</text>

</view>

</view>

<!-- 快速操作栏 -->

<view class="quick-actions">

<view class="action-btn primary" bindtap="onAddFollow">

<text class="iconfont icon-add"></text>

<text>添加跟进</text>

</view>

<view class="action-btn" bindtap="onBatchOperation">

<text class="iconfont icon-batch"></text>

<text>批量操作</text>

</view>

<view class="action-btn" bindtap="onExport">

<text class="iconfont icon-export"></text>

<text>导出记录</text>

</view>

</view>

<!-- 跟进列表 -->

<view class="follow-list">

<view

class="follow-item {{item.isUrgent ? 'urgent' : ''}}"

wx:for="{{followList}}"

wx:key="id"

bindtap="onItemTap"

data-item="{{item}}"

>

<!-- 客户信息头部 -->

<view class="item-header">

<view class="customer-info">

<text class="customer-name">{{item.customerName}}</text>

<text class="company-name">{{item.companyName}}</text>

<view class="status-badges">

<text class="badge status-{{item.customerStatus}}">{{item.customerStatusText}}</text>

<text class="badge priority-{{item.priority}}" wx:if="{{item.isUrgent}}">紧急</text>

</view>

</view>

<view class="item-actions">

<text class="iconfont icon-phone" bindtap="onCallCustomer" data-phone="{{item.phone}}"></text>

<text class="iconfont icon-wechat" bindtap="onContactWechat" data-id="{{item.id}}"></text>

</view>

</view>

<!-- 跟进内容 -->

<view class="follow-content">

<view class="follow-type">

<text class="type-icon {{item.followType}}"></text>

<text class="type-text">{{item.followTypeText}}</text>

</view>

<view class="follow-desc">{{item.content}}</view>

<view class="follow-result" wx:if="{{item.result}}">

<text class="result-label">跟进结果：</text>

<text class="result-text">{{item.result}}</text>

</view>

</view>

<!-- 时间和下次跟进 -->

<view class="item-footer">

<view class="time-info">

<view class="follow-time">

<text class="iconfont icon-time"></text>

<text>{{item.followTime}}</text>

</view>

<view class="next-follow" wx:if="{{item.nextFollowDate}}">

<text class="iconfont icon-calendar"></text>

<text class="{{item.isOverdue ? 'overdue' : ''}}">

下次跟进：{{item.nextFollowDate}}

</text>

</view>

</view>

<view class="operation-btns">

<text class="op-btn" bindtap="onEditFollow" data-id="{{item.id}}">编辑</text>

<text class="op-btn" bindtap="onAddNextFollow" data-customer="{{item}}">继续跟进</text>

</view>

</view>

<!-- 附件显示 -->

<view class="attachments" wx:if="{{item.attachments && item.attachments.length > 0}}">

<text class="attach-label">附件：</text>

<view class="attach-list">

<view

class="attach-item"

wx:for="{{item.attachments}}"

wx:key="id"

wx:for-item="file"

bindtap="onPreviewFile"

data-file="{{file}}"

>

<text class="iconfont icon-{{file.type}}"></text>

<text class="file-name">{{file.name}}</text>

</view>

</view>

</view>

</view>

<!-- 空状态 -->

<view class="empty-state" wx:if="{{followList.length === 0 && !loading}}">

<image class="empty-icon" src="/images/empty-follow.png"></image>

<text class="empty-text">暂无跟进记录</text>

<view class="empty-action" bindtap="onAddFollow">

<text>添加第一条跟进记录</text>

</view>

</view>

<!-- 加载更多 -->

<view class="load-more" wx:if="{{hasMore}}">

<text class="loading-text" wx:if="{{loading}}">加载中...</text>

<text class="load-text" wx:else bindtap="onLoadMore">点击加载更多</text>

</view>

</view>

<!-- 添加跟进浮动按钮 -->

<view class="fab" bindtap="onAddFollow">

<text class="iconfont icon-add"></text>

</view>

</view>

<!-- 添加/编辑跟进弹窗 -->

<view class="modal-overlay" wx:if="{{showFollowModal}}" bindtap="onCloseModal">

<view class="modal-content" catchtap="stopPropagation">

<view class="modal-header">

<text class="modal-title">{{isEditMode ? '编辑跟进记录' : '添加跟进记录'}}</text>

<text class="iconfont icon-close" bindtap="onCloseModal"></text>

</view>

<view class="modal-body">

<!-- 客户选择 -->

<view class="form-group" wx:if="{{!isEditMode}}">

<label class="form-label">选择客户</label>

<picker

bindchange="onCustomerChange"

value="{{customerIndex}}"

range="{{customerList}}"

range-key="name"

>

<view class="form-input picker-input">

{{selectedCustomer ? selectedCustomer.name + ' - ' + selectedCustomer.companyName : '请选择客户'}}

<text class="iconfont icon-arrow-down"></text>

</view>

</picker>

</view>

<!-- 跟进类型 -->

<view class="form-group">

<label class="form-label">跟进类型</label>

<picker

bindchange="onFollowTypeChange"

value="{{followTypeIndex}}"

range="{{followTypeList}}"

range-key="label"

>

<view class="form-input picker-input">

{{followForm.type ? followTypeList[followTypeIndex].label : '请选择类型'}}

<text class="iconfont icon-arrow-down"></text>

</view>

</picker>

</view>

<!-- 跟进内容 -->

<view class="form-group">

<label class="form-label">跟进内容</label>

<textarea

class="form-textarea"

placeholder="请输入跟进内容..."

value="{{followForm.content}}"

bindinput="onContentInput"

maxlength="500"

></textarea>

<text class="char-count">{{followForm.content.length}}/500</text>

</view>

<!-- 跟进结果 -->

<view class="form-group">

<label class="form-label">跟进结果</label>

<textarea

class="form-textarea"

placeholder="请输入跟进结果..."

value="{{followForm.result}}"

bindinput="onResultInput"

maxlength="300"

></textarea>

</view>

<!-- 下次跟进时间 -->

<view class="form-group">

<label class="form-label">下次跟进时间</label>

<picker

mode="multiSelector"

value="{{nextFollowIndex}}"

range="{{nextFollowRange}}"

bindchange="onNextFollowChange"

>

<view class="form-input picker-input">

{{followForm.nextFollowDate || '请选择时间'}}

<text class="iconfont icon-calendar"></text>

</view>

</picker>

</view>

<!-- 优先级 -->

<view class="form-group">

<label class="form-label">优先级</label>

<radio-group bindchange="onPriorityChange">

<view class="radio-group">

<label class="radio-item" wx:for="{{priorityList}}" wx:key="value">

<radio value="{{item.value}}" checked="{{followForm.priority === item.value}}"/>

<text class="radio-text">{{item.label}}</text>

</label>

</view>

</radio-group>

</view>

<!-- 附件上传 -->

<view class="form-group">

<label class="form-label">附件</label>

<view class="upload-area">

<view class="upload-btn" bindtap="onUploadAttachment">

<text class="iconfont icon-upload"></text>

<text>上传附件</text>

</view>

<view class="file-list" wx:if="{{followForm.attachments.length > 0}}">

<view

class="file-item"

wx:for="{{followForm.attachments}}"

wx:key="id"

wx:for-item="file"

>

<text class="iconfont icon-{{file.type}}"></text>

<text class="file-name">{{file.name}}</text>

<text class="iconfont icon-delete" bindtap="onDeleteFile" data-index="{{index}}"></text>

</view>

</view>

</view>

</view>

</view>

<view class="modal-footer">

<button class="btn btn-default" bindtap="onCloseModal">取消</button>

<button class="btn btn-primary" bindtap="onSaveFollow" loading="{{saving}}">

{{saving ? '保存中...' : '保存'}}

</button>

</view>

</view>

</view>

<!-- 客户选择弹窗 -->

<view class="modal-overlay" wx:if="{{showCustomerModal}}" bindtap="onCloseCustomerModal">

<view class="modal-content customer-modal" catchtap="stopPropagation">

<view class="modal-header">

<text class="modal-title">选择客户</text>

<text class="iconfont icon-close" bindtap="onCloseCustomerModal"></text>

</view>

<view class="modal-body">

<view class="search-box">

<input

class="search-input"

placeholder="搜索客户名称或公司"

value="{{customerSearchKeyword}}"

bindinput="onCustomerSearchInput"

/>

<text class="iconfont icon-search"></text>

</view>

<view class="customer-list">

<view

class="customer-item {{selectedCustomerId === item.id ? 'selected' : ''}}"

wx:for="{{availableCustomers}}"

wx:key="id"

bindtap="onSelectCustomer"

data-customer="{{item}}"

>

<view class="customer-info">

<text class="customer-name">{{item.name}}</text>

<text class="company-name">{{item.companyName}}</text>

<text class="customer-phone">{{item.phone}}</text>

</view>

<view class="customer-status">

<text class="badge status-{{item.status}}">{{item.statusText}}</text>

</view>

</view>

</view>

</view>

</view>

</view>

<!-- Toast提示 -->

<view class="toast {{showToast ? 'show' : ''}}" wx:if="{{toastMessage}}">

<text>{{toastMessage}}</text>

</view>

接下来开发跟进管理页面样式文件：

跟进管理页面样式文件 - pages/manager/follow/follow.wxss

/\* pages/manager/follow/follow.wxss \*/

.container {

background-color: var(--bg-color-light);

min-height: 100vh;

padding-bottom: 120rpx;

}

/\* 筛选栏样式 \*/

.filter-bar {

display: flex;

padding: 20rpx;

background-color: var(--bg-color-white);

border-bottom: 2rpx solid var(--border-color-light);

margin-bottom: 20rpx;

}

.filter-item {

flex: 1;

margin-right: 20rpx;

}

.filter-item:last-child {

margin-right: 0;

}

.picker-text {

display: flex;

align-items: center;

justify-content: space-between;

padding: 16rpx 20rpx;

background-color: var(--bg-color-light);

border-radius: 8rpx;

font-size: 28rpx;

color: var(--text-color-primary);

}

.search-item {

flex: 2;

position: relative;

}

.search-input {

width: 100%;

height: 68rpx;

padding: 0 60rpx 0 20rpx;

background-color: var(--bg-color-light);

border-radius: 8rpx;

font-size: 28rpx;

}

.search-item .icon-search {

position: absolute;

right: 20rpx;

top: 50%;

transform: translateY(-50%);

color: var(--text-color-secondary);

font-size: 32rpx;

}

/\* 统计信息栏 \*/

.stats-bar {

display: flex;

padding: 20rpx;

background-color: var(--bg-color-white);

margin-bottom: 20rpx;

}

.stat-item {

flex: 1;

text-align: center;

}

.stat-number {

display: block;

font-size: 48rpx;

font-weight: bold;

color: var(--primary-color);

margin-bottom: 8rpx;

}

.stat-label {

font-size: 24rpx;

color: var(--text-color-secondary);

}

/\* 快速操作栏 \*/

.quick-actions {

display: flex;

padding: 20rpx;

background-color: var(--bg-color-white);

margin-bottom: 20rpx;

}

.action-btn {

display: flex;

align-items: center;

flex: 1;

padding: 20rpx;

background-color: var(--bg-color-light);

border-radius: 8rpx;

margin-right: 20rpx;

text-align: center;

justify-content: center;

transition: all 0.3s ease;

}

.action-btn:last-child {

margin-right: 0;

}

.action-btn:active {

transform: scale(0.95);

opacity: 0.8;

}

.action-btn.primary {

background-color: var(--primary-color);

color: var(--bg-color-white);

}

.action-btn text {

font-size: 28rpx;

}

.action-btn .iconfont {

margin-right: 8rpx;

font-size: 32rpx;

}

/\* 跟进列表 \*/

.follow-list {

padding: 0 20rpx;

}

.follow-item {

background-color: var(--bg-color-white);

border-radius: 12rpx;

padding: 30rpx;

margin-bottom: 20rpx;

border-left: 8rpx solid var(--border-color-light);

transition: all 0.3s ease;

}

.follow-item.urgent {

border-left-color: var(--error-color);

}

.follow-item:active {

transform: scale(0.98);

}

/\* 项目头部 \*/

.item-header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 20rpx;

}

.customer-info {

flex: 1;

}

.customer-name {

font-size: 32rpx;

font-weight: bold;

color: var(--text-color-primary);

margin-right: 16rpx;

}

.company-name {

display: block;

font-size: 26rpx;

color: var(--text-color-secondary);

margin: 8rpx 0;

}

.status-badges {

display: flex;

gap: 12rpx;

margin-top: 12rpx;

}

.badge {

padding: 6rpx 12rpx;

border-radius: 20rpx;

font-size: 22rpx;

color: var(--bg-color-white);

}

.badge.status-potential {

background-color: var(--warning-color);

}

.badge.status-intention {

background-color: var(--info-color);

}

.badge.status-following {

background-color: var(--primary-color);

}

.badge.status-signed {

background-color: var(--success-color);

}

.badge.priority-high {

background-color: var(--error-color);

}

.item-actions {

display: flex;

gap: 20rpx;

}

.item-actions .iconfont {

width: 60rpx;

height: 60rpx;

line-height: 60rpx;

text-align: center;

border-radius: 50%;

font-size: 32rpx;

color: var(--primary-color);

background-color: rgba(74, 144, 226, 0.1);

transition: all 0.3s ease;

}

.item-actions .iconfont:active {

transform: scale(0.9);

}

/\* 跟进内容 \*/

.follow-content {

margin-bottom: 20rpx;

}

.follow-type {

display: flex;

align-items: center;

margin-bottom: 16rpx;

}

.type-icon {

width: 12rpx;

height: 12rpx;

border-radius: 50%;

margin-right: 12rpx;

}

.type-icon.phone {

background-color: var(--success-color);

}

.type-icon.visit {

background-color: var(--primary-color);

}

.type-icon.wechat {

background-color: var(--warning-color);

}

.type-icon.email {

background-color: var(--info-color);

}

.type-text {

font-size: 26rpx;

color: var(--text-color-secondary);

}

.follow-desc {

font-size: 30rpx;

color: var(--text-color-primary);

line-height: 1.6;

margin-bottom: 12rpx;

}

.follow-result {

background-color: var(--bg-color-light);

padding: 16rpx;

border-radius: 8rpx;

}

.result-label {

font-size: 26rpx;

color: var(--text-color-secondary);

}

.result-text {

font-size: 28rpx;

color: var(--text-color-primary);

}

/\* 项目底部 \*/

.item-footer {

display: flex;

justify-content: space-between;

align-items: flex-end;

}

.time-info {

flex: 1;

}

.follow-time,

.next-follow {

display: flex;

align-items: center;

font-size: 24rpx;

color: var(--text-color-secondary);

margin-bottom: 8rpx;

}

.follow-time:last-child,

.next-follow:last-child {

margin-bottom: 0;

}

.next-follow.overdue {

color: var(--error-color);

}

.time-info .iconfont {

margin-right: 8rpx;

font-size: 24rpx;

}

.operation-btns {

display: flex;

gap: 20rpx;

}

.op-btn {

padding: 12rpx 24rpx;

border-radius: 6rpx;

font-size: 24rpx;

color: var(--primary-color);

background-color: rgba(74, 144, 226, 0.1);

transition: all 0.3s ease;

}

.op-btn:active {

transform: scale(0.95);

opacity: 0.8;

}

/\* 附件显示 \*/

.attachments {

margin-top: 20rpx;

padding-top: 20rpx;

border-top: 2rpx solid var(--border-color-light);

}

.attach-label {

font-size: 26rpx;

color: var(--text-color-secondary);

margin-bottom: 12rpx;

}

.attach-list {

display: flex;

flex-wrap: wrap;

gap: 12rpx;

}

.attach-item {

display: flex;

align-items: center;

padding: 8rpx 16rpx;

background-color: var(--bg-color-light);

border-radius: 6rpx;

font-size: 24rpx;

color: var(--text-color-secondary);

}

.attach-item .iconfont {

margin-right: 6rpx;

font-size: 24rpx;

}

.file-name {

max-width: 200rpx;

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap;

}

/\* 空状态 \*/

.empty-state {

text-align: center;

padding: 120rpx 40rpx;

}

.empty-icon {

width: 200rpx;

height: 200rpx;

margin-bottom: 40rpx;

}

.empty-text {

display: block;

font-size: 32rpx;

color: var(--text-color-secondary);

margin-bottom: 40rpx;

}

.empty-action {

padding: 20rpx 40rpx;

background-color: var(--primary-color);

color: var(--bg-color-white);

border-radius: 8rpx;

font-size: 28rpx;

display: inline-block;

}

/\* 加载更多 \*/

.load-more {

text-align: center;

padding: 40rpx;

font-size: 28rpx;

color: var(--text-color-secondary);

}

.load-text {

color: var(--primary-color);

}

/\* 浮动按钮 \*/

.fab {

position: fixed;

right: 40rpx;

bottom: 120rpx;

width: 100rpx;

height: 100rpx;

background-color: var(--primary-color);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 4rpx 12rpx rgba(74, 144, 226, 0.3);

z-index: 100;

transition: all 0.3s ease;

}

.fab:active {

transform: scale(0.9);

}

.fab .iconfont {

font-size: 40rpx;

color: var(--bg-color-white);

}

/\* 弹窗样式 \*/

.modal-overlay {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

display: flex;

align-items: center;

justify-content: center;

z-index: 1000;

}

.modal-content {

width: 90%;

max-width: 600rpx;

max-height: 80vh;

background-color: var(--bg-color-white);

border-radius: 16rpx;

overflow: hidden;

}

.customer-modal {

max-height: 70vh;

}

.modal-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 30rpx;

border-bottom: 2rpx solid var(--border-color-light);

}

.modal-title {

font-size: 36rpx;

font-weight: bold;

color: var(--text-color-primary);

}

.modal-header .icon-close {

font-size: 40rpx;

color: var(--text-color-secondary);

}

.modal-body {

padding: 30rpx;

max-height: 60vh;

overflow-y: auto;

}

.modal-footer {

display: flex;

justify-content: flex-end;

gap: 20rpx;

padding: 30rpx;

border-top: 2rpx solid var(--border-color-light);

}

/\* 表单样式 \*/

.form-group {

margin-bottom: 40rpx;

}

.form-label {

display: block;

font-size: 28rpx;

color: var(--text-color-primary);

margin-bottom: 12rpx;

font-weight: 500;

}

.form-input {

width: 100%;

padding: 20rpx;

border: 2rpx solid var(--border-color-light);

border-radius: 8rpx;

font-size: 28rpx;

color: var(--text-color-primary);

}

.form-input:focus {

border-color: var(--primary-color);

}

.picker-input {

display: flex;

align-items: center;

justify-content: space-between;

background-color: var(--bg-color-light);

}

.form-textarea {

width: 100%;

min-height: 120rpx;

padding: 20rpx;

border: 2rpx solid var(--border-color-light);

border-radius: 8rpx;

font-size: 28rpx;

color: var(--text-color-primary);

resize: none;

}

.char-count {

display: block;

text-align: right;

font-size: 24rpx;

color: var(--text-color-secondary);

margin-top: 8rpx;

}

.radio-group {

display: flex;

flex-wrap: wrap;

gap: 30rpx;

}

.radio-item {

display: flex;

align-items: center;

margin-right: 30rpx;

}

.radio-circle {

width: 36rpx;

height: 36rpx;

border: 4rpx solid var(--border-color-light);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

margin-right: 12rpx;

transition: all 0.3s ease;

}

.radio-circle.active {

border-color: var(--primary-color);

background-color: var(--primary-color);

}

.radio-circle.active::after {

content: '';

width: 16rpx;

height: 16rpx;

background-color: var(--bg-color-white);

border-radius: 50%;

}

.radio-text {

font-size: 28rpx;

color: var(--text-color-primary);

}

/\* 文件上传样式 \*/

.upload-area {

display: flex;

flex-wrap: wrap;

gap: 20rpx;

margin-top: 20rpx;

}

.upload-item {

position: relative;

width: 160rpx;

height: 160rpx;

border: 2rpx dashed var(--border-color-light);

border-radius: 12rpx;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

background-color: var(--bg-color-light);

transition: all 0.3s ease;

}

.upload-item:active {

transform: scale(0.95);

}

.upload-placeholder {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

width: 100%;

height: 100%;

}

.upload-icon {

font-size: 48rpx;

color: var(--text-color-secondary);

margin-bottom: 8rpx;

}

.upload-text {

font-size: 24rpx;

color: var(--text-color-secondary);

text-align: center;

}

.uploaded-image {

width: 100%;

height: 100%;

border-radius: 12rpx;

object-fit: cover;

}

.delete-btn {

position: absolute;

top: -10rpx;

right: -10rpx;

width: 40rpx;

height: 40rpx;

background-color: var(--error-color);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

font-size: 24rpx;

color: var(--bg-color-white);

z-index: 10;

}

/\* 按钮样式 \*/

.btn {

display: inline-flex;

align-items: center;

justify-content: center;

padding: 24rpx 48rpx;

border-radius: 8rpx;

font-size: 28rpx;

font-weight: 500;

text-align: center;

border: none;

cursor: pointer;

transition: all 0.3s ease;

min-width: 160rpx;

}

.btn:active {

transform: scale(0.95);

}

.btn-primary {

background-color: var(--primary-color);

color: var(--bg-color-white);

}

.btn-primary:active {

background-color: #3a7bc8;

}

.btn-secondary {

background-color: var(--bg-color-light);

color: var(--text-color-primary);

border: 2rpx solid var(--border-color-light);

}

.btn-secondary:active {

background-color: #e8e8e8;

}

.btn-danger {

background-color: var(--error-color);

color: var(--bg-color-white);

}

.btn-danger:active {

background-color: #d63031;

}

.btn-outline {

background-color: transparent;

color: var(--primary-color);

border: 2rpx solid var(--primary-color);

}

.btn-outline:active {

background-color: rgba(74, 144, 226, 0.1);

}

.btn:disabled {

opacity: 0.5;

cursor: not-allowed;

}

.btn:disabled:active {

transform: none;

}

/\* 标签样式 \*/

.tag {

display: inline-flex;

align-items: center;

padding: 8rpx 16rpx;

border-radius: 20rpx;

font-size: 24rpx;

font-weight: 500;

margin-right: 12rpx;

margin-bottom: 12rpx;

}

.tag-primary {

background-color: rgba(74, 144, 226, 0.1);

color: var(--primary-color);

}

.tag-success {

background-color: rgba(0, 184, 148, 0.1);

color: var(--success-color);

}

.tag-warning {

background-color: rgba(253, 203, 110, 0.1);

color: var(--warning-color);

}

.tag-danger {

background-color: rgba(255, 107, 107, 0.1);

color: var(--error-color);

}

.tag-info {

background-color: rgba(116, 185, 255, 0.1);

color: var(--info-color);

}

/\* 工具提示 \*/

.tooltip {

position: relative;

display: inline-block;

}

.tooltip-content {

visibility: hidden;

position: absolute;

bottom: 100%;

left: 50%;

transform: translateX(-50%);

background-color: rgba(0, 0, 0, 0.8);

color: var(--bg-color-white);

padding: 16rpx 20rpx;

border-radius: 8rpx;

font-size: 24rpx;

white-space: nowrap;

z-index: 1000;

margin-bottom: 8rpx;

}

.tooltip:hover .tooltip-content {

visibility: visible;

}

/\* 分割线 \*/

.divider {

width: 100%;

height: 2rpx;

background-color: var(--border-color-light);

margin: 40rpx 0;

}

.divider-vertical {

width: 2rpx;

height: 60rpx;

background-color: var(--border-color-light);

margin: 0 20rpx;

}

/\* 进度条 \*/

.progress {

width: 100%;

height: 8rpx;

background-color: var(--bg-color-light);

border-radius: 4rpx;

overflow: hidden;

}

.progress-bar {

height: 100%;

background-color: var(--primary-color);

border-radius: 4rpx;

transition: width 0.3s ease;

}

.progress-text {

font-size: 24rpx;

color: var(--text-color-secondary);

margin-top: 8rpx;

text-align: right;

}

/\* 步骤指示器 \*/

.steps {

display: flex;

align-items: center;

padding: 40rpx 20rpx;

}

.step {

display: flex;

flex-direction: column;

align-items: center;

flex: 1;

position: relative;

}

.step-circle {

width: 60rpx;

height: 60rpx;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

font-size: 24rpx;

font-weight: bold;

background-color: var(--bg-color-light);

color: var(--text-color-secondary);

border: 4rpx solid var(--border-color-light);

z-index: 2;

}

.step.active .step-circle {

background-color: var(--primary-color);

color: var(--bg-color-white);

border-color: var(--primary-color);

}

.step.completed .step-circle {

background-color: var(--success-color);

color: var(--bg-color-white);

border-color: var(--success-color);

}

.step-title {

font-size: 24rpx;

color: var(--text-color-secondary);

margin-top: 12rpx;

text-align: center;

}

.step.active .step-title {

color: var(--primary-color);

font-weight: bold;

}

.step-line {

position: absolute;

top: 30rpx;

left: 50%;

right: -50%;

height: 4rpx;

background-color: var(--border-color-light);

z-index: 1;

}

.step:last-child .step-line {

display: none;

}

.step.completed .step-line {

background-color: var(--success-color);

}

/\* 响应式适配 \*/

@media screen and (max-width: 375px) {

.filter-bar {

flex-direction: column;

gap: 20rpx;

}

.filter-item {

margin-right: 0;

margin-bottom: 20rpx;

}

.filter-item:last-child {

margin-bottom: 0;

}

.stats-bar {

flex-wrap: wrap;

}

.stat-item {

flex: 1 1 50%;

margin-bottom: 20rpx;

}

.quick-actions {

flex-direction: column;

}

.action-btn {

margin-right: 0;

margin-bottom: 20rpx;

}

.action-btn:last-child {

margin-bottom: 0;

}

.modal-content {

width: 95%;

}

.upload-area {

justify-content: center;

}

.upload-item {

width: 120rpx;

height: 120rpx;

}

}

/\* 动画效果 \*/

@keyframes fadeIn {

from {

opacity: 0;

transform: translateY(20rpx);

}

to {

opacity: 1;

transform: translateY(0);

}

}

@keyframes slideIn {

from {

transform: translateX(-100%);

}

to {

transform: translateX(0);

}

}

@keyframes pulse {

0% {

transform: scale(1);

}

50% {

transform: scale(1.05);

}

100% {

transform: scale(1);

}

}

.fade-in {

animation: fadeIn 0.3s ease-out;

}

.slide-in {

animation: slideIn 0.3s ease-out;

}

.pulse {

animation: pulse 2s infinite;

}

/\* 深色模式支持 \*/

@media (prefers-color-scheme: dark) {

.tooltip-content {

background-color: rgba(255, 255, 255, 0.9);

color: var(--text-color-primary);

}

}

跟进管理页面样式文件补充完成！现在继续开发下一个模块：**跟进管理页面配置文件**

跟进管理页面配置文件 - pages/manager/follow/follow.json

{

"navigationBarTitleText": "跟进管理",

"navigationBarBackgroundColor": "#4A90E2",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f5f5",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"disableScroll": false,

"usingComponents": {

"van-picker": "@vant/weapp/picker/index",

"van-popup": "@vant/weapp/popup/index",

"van-datetime-picker": "@vant/weapp/datetime-picker/index",

"van-field": "@vant/weapp/field/index",

"van-button": "@vant/weapp/button/index",

"van-cell": "@vant/weapp/cell/index",

"van-cell-group": "@vant/weapp/cell-group/index",

"van-radio": "@vant/weapp/radio/index",

"van-radio-group": "@vant/weapp/radio-group/index",

"van-uploader": "@vant/weapp/uploader/index",

"van-loading": "@vant/weapp/loading/index",

"van-empty": "@vant/weapp/empty/index",

"van-divider": "@vant/weapp/divider/index",

"van-tag": "@vant/weapp/tag/index",

"van-toast": "@vant/weapp/toast/index",

"van-dialog": "@vant/weapp/dialog/index",

"van-action-sheet": "@vant/weapp/action-sheet/index",

"van-sticky": "@vant/weapp/sticky/index",

"van-dropdown-menu": "@vant/weapp/dropdown-menu/index",

"van-dropdown-item": "@vant/weapp/dropdown-item/index",

"van-search": "@vant/weapp/search/index",

"van-tabs": "@vant/weapp/tabs/index",

"van-tab": "@vant/weapp/tab/index",

"van-steps": "@vant/weapp/steps/index",

"van-step": "@vant/weapp/step/index"

},

"style": "v2",

"singlePage": {

"navigationBarFit": "squeezed"

}

}

现在继续开发下一个模块：**业绩统计页面逻辑**

业绩统计页面逻辑 - pages/manager/performance/performance.js

// pages/manager/performance/performance.js

const api = require('../../../utils/api');

const util = require('../../../utils/util');

Page({

data: {

userInfo: null,

loading: false,

refreshing: false,

// 时间筛选

currentPeriod: 'month', // month, quarter, year

periodOptions: [

{ label: '本月', value: 'month' },

{ label: '本季度', value: 'quarter' },

{ label: '本年度', value: 'year' }

],

customDateRange: {

startDate: '',

endDate: ''

},

showDatePicker: false,

datePickerType: 'start', // start, end

// 业绩数据

performanceData: {

// 个人业绩概览

overview: {

totalCustomers: 0,

newCustomers: 0,

activeCustomers: 0,

signedCustomers: 0,

totalOrders: 0,

totalAmount: 0,

completionRate: 0,

ranking: 0,

targetAmount: 0,

achievementRate: 0

},

// 月度趋势

monthlyTrend: [],

// 客户分布

customerDistribution: {

byStatus: [],

byIndustry: [],

byRegion: []

},

// 订单分析

orderAnalysis: {

byStatus: [],

byProduct: [],

avgOrderValue: 0,

conversionRate: 0

},

// 跟进效率

followEfficiency: {

totalFollows: 0,

avgResponseTime: 0,

customerSatisfaction: 0,

completedTasks: 0,

pendingTasks: 0

}

},

// 排行榜数据

rankings: {

personal: [],

team: []

},

// 图表配置

chartOptions: {

trend: {

tooltip: {

trigger: 'axis'

},

legend: {

data: ['新增客户', '成交订单', '订单金额']

},

xAxis: {

type: 'category',

data: []

},

yAxis: [

{

type: 'value',

name: '数量',

position: 'left'

},

{

type: 'value',

name: '金额(万元)',

position: 'right'

}

],

series: []

},

distribution: {

tooltip: {

trigger: 'item'

},

legend: {

orient: 'vertical',

left: 'left'

},

series: []

}

},

// UI状态

activeTab: 0,

tabs: [

{ title: '业绩概览', key: 'overview' },

{ title: '数据分析', key: 'analysis' },

{ title: '排行榜', key: 'ranking' }

],

showShareModal: false,

shareOptions: [

{ name: '分享给微信好友', icon: 'wechat', openType: 'share' },

{ name: '保存到相册', icon: 'save', action: 'saveToAlbum' },

{ name: '生成报告', icon: 'report', action: 'generateReport' }

]

},

onLoad(options) {

console.log('业绩统计页面加载', options);

this.getUserInfo();

this.initializeData();

},

onShow() {

this.loadPerformanceData();

},

onPullDownRefresh() {

this.setData({ refreshing: true });

this.loadPerformanceData().finally(() => {

this.setData({ refreshing: false });

wx.stopPullDownRefresh();

});

},

onReachBottom() {

// 如果有分页数据，可以在这里加载更多

},

// 获取用户信息

getUserInfo() {

const userInfo = wx.getStorageSync('userInfo');

if (userInfo) {

this.setData({ userInfo });

}

},

// 初始化数据

initializeData() {

const now = new Date();

const startOfMonth = new Date(now.getFullYear(), now.getMonth(), 1);

const endOfMonth = new Date(now.getFullYear(), now.getMonth() + 1, 0);

this.setData({

'customDateRange.startDate': util.formatDate(startOfMonth),

'customDateRange.endDate': util.formatDate(endOfMonth)

});

},

// 加载业绩数据

async loadPerformanceData() {

if (this.data.loading) return;

this.setData({ loading: true });

try {

const { currentPeriod, customDateRange } = this.data;

const params = {

period: currentPeriod,

startDate: customDateRange.startDate,

endDate: customDateRange.endDate

};

const [overviewRes, trendRes, distributionRes, rankingRes] = await Promise.all([

api.getPerformanceOverview(params),

api.getPerformanceTrend(params),

api.getCustomerDistribution(params),

api.getPerformanceRanking(params)

]);

this.setData({

'performanceData.overview': overviewRes.data,

'performanceData.monthlyTrend': trendRes.data,

'performanceData.customerDistribution': distributionRes.data,

'rankings': rankingRes.data

});

this.updateCharts();

} catch (error) {

console.error('加载业绩数据失败:', error);

wx.showToast({

title: '加载失败',

icon: 'none'

});

} finally {

this.setData({ loading: false });

}

},

// 更新图表

updateCharts() {

const { monthlyTrend, customerDistribution } = this.data.performanceData;

// 更新趋势图

if (monthlyTrend && monthlyTrend.length > 0) {

const trendChart = {

...this.data.chartOptions.trend,

xAxis: {

...this.data.chartOptions.trend.xAxis,

data: monthlyTrend.map(item => item.month)

},

series: [

{

name: '新增客户',

type: 'line',

yAxisIndex: 0,

data: monthlyTrend.map(item => item.newCustomers)

},

{

name: '成交订单',

type: 'line',

yAxisIndex: 0,

data: monthlyTrend.map(item => item.signedOrders)

},

{

name: '订单金额',

type: 'line',

yAxisIndex: 1,

data: monthlyTrend.map(item => (item.orderAmount / 10000).toFixed(2))

}

]

};

this.setData({

'chartOptions.trend': trendChart

});

}

// 更新分布图

if (customerDistribution && customerDistribution.byStatus) {

const distributionChart = {

...this.data.chartOptions.distribution,

series: [{

name: '客户状态分布',

type: 'pie',

radius: '50%',

data: customerDistribution.byStatus.map(item => ({

value: item.count,

name: item.status

}))

}]

};

this.setData({

'chartOptions.distribution': distributionChart

});

}

},

// 切换时间周期

onPeriodChange(e) {

const period = e.currentTarget.dataset.period;

this.setData({ currentPeriod: period });

this.updateDateRange(period);

this.loadPerformanceData();

},

// 更新日期范围

updateDateRange(period) {

const now = new Date();

let startDate, endDate;

switch (period) {

case 'month':

startDate = new Date(now.getFullYear(), now.getMonth(), 1);

endDate = new Date(now.getFullYear(), now.getMonth() + 1, 0);

break;

case 'quarter':

const quarter = Math.floor(now.getMonth() / 3);

startDate = new Date(now.getFullYear(), quarter \* 3, 1);

endDate = new Date(now.getFullYear(), quarter \* 3 + 3, 0);

break;

case 'year':

startDate = new Date(now.getFullYear(), 0, 1);

endDate = new Date(now.getFullYear(), 11, 31);

break;

}

this.setData({

'customDateRange.startDate': util.formatDate(startDate),

'customDateRange.endDate': util.formatDate(endDate)

});

},

// 显示日期选择器

showDatePicker(e) {

const type = e.currentTarget.dataset.type;

this.setData({

datePickerType: type,

showDatePicker: true

});

},

// 日期选择确认

onDateConfirm(e) {

const { datePickerType } = this.data;

const selectedDate = util.formatDate(new Date(e.detail));

this.setData({

[`customDateRange.${datePickerType}Date`]: selectedDate,

showDatePicker: false

});

// 如果开始日期和结束日期都已选择，重新加载数据

const { startDate, endDate } = this.data.customDateRange;

if (startDate && endDate) {

this.loadPerformanceData();

}

},

// 取消日期选择

onDateCancel() {

this.setData({ showDatePicker: false });

},

// 切换标签页

onTabChange(e) {

const index = e.detail.index || e.currentTarget.dataset.index;

this.setData({ activeTab: index });

},

// 查看客户详情

viewCustomerDetail(e) {

const customerId = e.currentTarget.dataset.id;

wx.navigateTo({

url: `/pages/manager/customers/detail/detail?id=${customerId}`

});

},

// 查看订单详情

viewOrderDetail(e) {

const orderId = e.currentTarget.dataset.id;

wx.navigateTo({

url: `/pages/orders/detail/detail?id=${orderId}`

});

},

// 显示分享弹窗

showShareModal() {

this.setData({ showShareModal: true });

},

// 隐藏分享弹窗

hideShareModal() {

this.setData({ showShareModal: false });

},

// 分享操作

onShareAction(e) {

const action = e.currentTarget.dataset.action;

switch (action) {

case 'saveToAlbum':

this.saveToAlbum();

break;

case 'generateReport':

this.generateReport();

break;

}

this.hideShareModal();

},

// 保存到相册

async saveToAlbum() {

try {

wx.showLoading({ title: '保存中...' });

// 这里可以调用canvas绘制业绩报告图片

// 然后保存到相册

wx.showToast({

title: '保存成功',

icon: 'success'

});

} catch (error) {

console.error('保存失败:', error);

wx.showToast({

title: '保存失败',

icon: 'none'

});

} finally {

wx.hideLoading();

}

},

// 生成业绩报告

async generateReport() {

try {

wx.showLoading({ title: '生成中...' });

const { currentPeriod, customDateRange } = this.data;

const params = {

period: currentPeriod,

startDate: customDateRange.startDate,

endDate: customDateRange.endDate,

format: 'pdf'

};

const res = await api.generatePerformanceReport(params);

if (res.data.url) {

wx.downloadFile({

url: res.data.url,

success: (downloadRes) => {

wx.openDocument({

filePath: downloadRes.tempFilePath,

success: () => {

wx.showToast({

title: '报告生成成功',

icon: 'success'

});

}

});

}

});

}

} catch (error) {

console.error('生成报告失败:', error);

wx.showToast({

title: '生成失败',

icon: 'none'

});

} finally {

wx.hideLoading();

}

},

// 设置业绩目标

setTarget() {

wx.showModal({

title: '设置业绩目标',

editable: true,

placeholderText: '请输入目标金额',

success: async (res) => {

if (res.confirm && res.content) {

try {

const amount = parseFloat(res.content);

if (isNaN(amount) || amount <= 0) {

wx.showToast({

title: '请输入有效金额',

icon: 'none'

});

return;

}

await api.setPerformanceTarget({

period: this.data.currentPeriod,

targetAmount: amount

});

wx.showToast({

title: '设置成功',

icon: 'success'

});

this.loadPerformanceData();

} catch (error) {

console.error('设置目标失败:', error);

wx.showToast({

title: '设置失败',

icon:

现在继续开发下一个模块：**业绩统计页面模板 - pages/manager/performance/performance.wxml**

业绩统计页面模板 - pages/manager/performance/performance.wxml

<!-- 业绩统计页面 -->

<view class="performance-container">

<!-- 页面头部 -->

<view class="page-header">

<view class="header-title">业绩统计</view>

<view class="header-actions">

<view class="action-btn" bindtap="showShareModal">

<image src="/images/icons/share.png" class="action-icon" />

</view>

<view class="action-btn" bindtap="exportData">

<image src="/images/icons/export.png" class="action-icon" />

</view>

<view class="action-btn" bindtap="setReminder">

<image src="/images/icons/reminder.png" class="action-icon" />

</view>

</view>

</view>

<!-- 时间筛选 -->

<view class="time-filter">

<view class="filter-tabs">

<view

wx:for="{{periodOptions}}"

wx:key="value"

class="filter-tab {{currentPeriod === item.value ? 'active' : ''}}"

data-period="{{item.value}}"

bindtap="onPeriodChange"

>

{{item.label}}

</view>

</view>

<view class="custom-date">

<view class="date-picker" bindtap="showDatePicker" data-type="start">

<text class="date-text">{{customDateRange.startDate || '开始日期'}}</text>

<image src="/images/icons/calendar.png" class="date-icon" />

</view>

<view class="date-separator">至</view>

<view class="date-picker" bindtap="showDatePicker" data-type="end">

<text class="date-text">{{customDateRange.endDate || '结束日期'}}</text>

<image src="/images/icons/calendar.png" class="date-icon" />

</view>

</view>

</view>

<!-- 内容区域 -->

<view class="content-container">

<!-- 标签页 -->

<view class="tabs">

<view

wx:for="{{tabs}}"

wx:key="key"

class="tab-item {{activeTab === index ? 'active' : ''}}"

data-index="{{index}}"

bindtap="onTabChange"

>

{{item.title}}

</view>

</view>

<!-- 业绩概览 -->

<view wx:if="{{activeTab === 0}}" class="tab-content">

<!-- 核心指标卡片 -->

<view class="metrics-grid">

<view class="metric-card">

<view class="metric-value">{{performanceData.overview.totalCustomers}}</view>

<view class="metric-label">总客户数</view>

<view class="metric-change positive">

<image src="/images/icons/arrow-up.png" class="change-icon" />

+{{performanceData.overview.newCustomers}}

</view>

</view>

<view class="metric-card">

<view class="metric-value">{{performanceData.overview.totalOrders}}</view>

<view class="metric-label">总订单数</view>

<view class="metric-change positive">

<image src="/images/icons/arrow-up.png" class="change-icon" />

+{{performanceData.overview.signedCustomers}}

</view>

</view>

<view class="metric-card">

<view class="metric-value">{{formatAmount(performanceData.overview.totalAmount)}}</view>

<view class="metric-label">总订单金额</view>

<view class="metric-change positive">

<image src="/images/icons/arrow-up.png" class="change-icon" />

{{formatPercent(performanceData.overview.achievementRate)}}

</view>

</view>

<view class="metric-card">

<view class="metric-value">{{performanceData.overview.ranking}}</view>

<view class="metric-label">个人排名</view>

<view class="metric-change">

完成率{{formatPercent(performanceData.overview.completionRate)}}

</view>

</view>

</view>

<!-- 目标完成进度 -->

<view class="progress-section">

<view class="section-title">

<text>目标完成进度</text>

<text class="target-action" bindtap="setTarget">设置目标</text>

</view>

<view class="progress-card">

<view class="progress-info">

<view class="progress-text">

<text class="current-amount">{{formatAmount(performanceData.overview.totalAmount)}}</text>

<text class="target-amount">/ {{formatAmount(performanceData.overview.targetAmount)}}</text>

</view>

<view class="progress-percent">{{formatPercent(performanceData.overview.achievementRate)}}</view>

</view>

<view class="progress-bar">

<view class="progress-fill" style="width: {{performanceData.overview.achievementRate}}%"></view>

</view>

</view>

</view>

<!-- 快速操作 -->

<view class="quick-actions">

<view class="action-grid">

<view class="action-item" bindtap="viewCustomerDistribution" data-type="byStatus">

<image src="/images/icons/customers.png" class="action-icon" />

<text class="action-text">客户分布</text>

</view>

<view class="action-item" bindtap="viewDetailAnalysis" data-type="order">

<image src="/images/icons/orders.png" class="action-icon" />

<text class="action-text">订单分析</text>

</view>

<view class="action-item" bindtap="viewDetailAnalysis" data-type="follow">

<image src="/images/icons/follow.png" class="action-icon" />

<text class="action-text">跟进效率</text>

</view>

<view class="action-item" bindtap="viewTargetDetail">

<image src="/images/icons/target.png" class="action-icon" />

<text class="action-text">目标管理</text>

</view>

</view>

</view>

</view>

<!-- 数据分析 -->

<view wx:if="{{activeTab === 1}}" class="tab-content">

<!-- 趋势图表 -->

<view class="chart-section">

<view class="section-title">业绩趋势</view>

<view class="chart-container">

<ec-canvas id="trend-chart" canvas-id="trendChart" ec="{{chartOptions.trend}}"></ec-canvas>

</view>

</view>

<!-- 客户分布 -->

<view class="chart-section">

<view class="section-title">客户分布</view>

<view class="distribution-tabs">

<view class="dist-tab active">按状态</view>

<view class="dist-tab">按行业</view>

<view class="dist-tab">按地区</view>

</view>

<view class="chart-container">

<ec-canvas id="dist-chart" canvas-id="distChart" ec="{{chartOptions.distribution}}"></ec-canvas>

</view>

</view>

<!-- 详细数据 -->

<view class="data-section">

<view class="section-title">详细数据</view>

<view class="data-grid">

<view class="data-item">

<view class="data-label">平均客单价</view>

<view class="data-value">{{formatAmount(performanceData.orderAnalysis.avgOrderValue)}}</view>

</view>

<view class="data-item">

<view class="data-label">转化率</view>

<view class="data-value">{{formatPercent(performanceData.orderAnalysis.conversionRate)}}</view>

</view>

<view class="data-item">

<view class="data-label">客户满意度</view>

<view class="data-value">{{performanceData.followEfficiency.customerSatisfaction}}分</view>

</view>

<view class="data-item">

<view class="data-label">平均响应时间</view>

<view class="data-value">{{performanceData.followEfficiency.avgResponseTime}}小时</view>

</view>

</view>

</view>

</view>

<!-- 排行榜 -->

<view wx:if="{{activeTab === 2}}" class="tab-content">

<!-- 个人排行 -->

<view class="ranking-section">

<view class="section-title">

<text>个人排行榜</text>

<text class="ranking-action" bindtap="viewRankingDetail" data-type="personal">查看更多</text>

</view>

<view class="ranking-list">

<view

wx:for="{{rankings.personal}}"

wx:key="id"

class="ranking-item {{index < 3 ? 'top-three' : ''}}"

>

<view class="ranking-position">

<text wx:if="{{index < 3}}" class="medal">{{index === 0 ? '🥇' : index === 1 ? '🥈' : '🥉'}}</text>

<text wx:else class="rank-number">{{index + 1}}</text>

</view>

<view class="ranking-info">

<view class="employee-name">{{item.name}}</view>

<view class="employee-dept">{{item.department}}</view>

</view>

<view class="ranking-data">

<view class="data-primary">{{formatAmount(item.totalAmount)}}</view>

<view class="data-secondary">{{item.customerCount}}客户</view>

</view>

</view>

</view>

</view>

<!-- 团队排行 -->

<view class="ranking-section">

<view class="section-title">

<text>团队排行榜</text>

<text class="ranking-action" bindtap="viewRankingDetail" data-type="team">查看更多</text>

</view>

<view class="ranking-list">

<view

wx:for="{{rankings.team}}"

wx:key="id"

class="ranking-item {{index < 3 ? 'top-three' : ''}}"

>

<view class="ranking-position">

<text wx:if="{{index < 3}}" class="medal">{{index === 0 ? '🥇' : index === 1 ? '🥈' : '🥉'}}</text>

<text wx:else class="rank-number">{{index + 1}}</text>

</view>

<view class="ranking-info">

<view class="team-name">{{item.name}}</view>

<view class="team-members">{{item.memberCount}}人</view>

</view>

<view class="ranking-data">

<view class="data-primary">{{formatAmount(item.totalAmount)}}</view>

<view class="data-secondary">{{item.orderCount}}订单</view>

</view>

</view>

</view>

</view>

</view>

</view>

<!-- 加载状态 -->

<view wx:if="{{loading}}" class="loading-container">

<view class="loading-spinner"></view>

<text class="loading-text">数据加载中...</text>

</view>

</view>

<!-- 日期选择器 -->

<picker

wx:if="{{showDatePicker}}"

mode="date"

value="{{datePickerType === 'start' ? customDateRange.startDate : customDateRange.endDate}}"

bindchange="onDateConfirm"

bindcancel="onDateCancel"

>

<view class="date-picker-mask"></view>

</picker>

<!-- 分享弹窗 -->

<view wx:if="{{showShareModal}}" class="modal-overlay" bindtap="hideShareModal">

<view class="share-modal" catchtap="">

<view class="modal-header">

<text class="modal-title">分享业绩</text>

<view class="modal-close" bindtap="hideShareModal">×</view>

</view>

<view class="share-options">

<view

wx:for="{{shareOptions}}"

wx:key="name"

class="share-option"

open-type="{{item.openType}}"

data-action="{{item.action}}"

bindtap="onShareAction"

>

<image src="/images/icons/{{item.icon}}.png" class="share-icon" />

<text class="share-text">{{item.name}}</text>

</view>

</view>

</view>

</view>

<!-- 底部操作栏 -->

<view class="bottom-actions">

<view class="action-button secondary" bindtap="refreshData">

<image src="/images/icons/refresh.png" class="btn-icon" />

<text>刷新数据</text>

</view>

<view class="action-button primary" bindtap="exportData">

<image src="/images/icons/download.png" class="btn-icon" />

<text>导出报告</text>

</view>

</view>

现在继续开发下一个模块：**业绩统计页面样式文件 - pages/manager/performance/performance.wxss**

业绩统计页面样式文件 - pages/manager/performance/performance.wxss

/\* 业绩统计页面样式 \*/

.performance-container {

min-height: 100vh;

background-color: #f5f7fa;

padding-bottom: 120rpx;

}

/\* 页面头部 \*/

.page-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 24rpx 32rpx;

background-color: #fff;

border-bottom: 1rpx solid #eee;

}

.header-title {

font-size: 36rpx;

font-weight: 600;

color: #333;

}

.header-actions {

display: flex;

gap: 16rpx;

}

.action-btn {

width: 64rpx;

height: 64rpx;

display: flex;

align-items: center;

justify-content: center;

background-color: #f8f9fa;

border-radius: 50%;

border: 1rpx solid #e9ecef;

}

.action-icon {

width: 32rpx;

height: 32rpx;

}

/\* 时间筛选 \*/

.time-filter {

background-color: #fff;

padding: 24rpx 32rpx;

margin-bottom: 16rpx;

}

.filter-tabs {

display: flex;

gap: 16rpx;

margin-bottom: 24rpx;

}

.filter-tab {

padding: 12rpx 24rpx;

background-color: #f8f9fa;

border-radius: 20rpx;

font-size: 28rpx;

color: #666;

border: 1rpx solid #e9ecef;

transition: all 0.3s ease;

}

.filter-tab.active {

background-color: var(--primary-color);

color: #fff;

border-color: var(--primary-color);

}

.custom-date {

display: flex;

align-items: center;

gap: 16rpx;

}

.date-picker {

flex: 1;

display: flex;

align-items: center;

justify-content: space-between;

padding: 16rpx 20rpx;

background-color: #f8f9fa;

border-radius: 12rpx;

border: 1rpx solid #e9ecef;

}

.date-text {

font-size: 28rpx;

color: #333;

}

.date-icon {

width: 28rpx;

height: 28rpx;

}

.date-separator {

font-size: 28rpx;

color: #666;

}

/\* 内容区域 \*/

.content-container {

background-color: #fff;

margin: 0 16rpx;

border-radius: 16rpx;

overflow: hidden;

}

/\* 标签页 \*/

.tabs {

display: flex;

background-color: #f8f9fa;

border-bottom: 1rpx solid #e9ecef;

}

.tab-item {

flex: 1;

text-align: center;

padding: 24rpx 0;

font-size: 30rpx;

color: #666;

position: relative;

transition: all 0.3s ease;

}

.tab-item.active {

color: var(--primary-color);

font-weight: 600;

}

.tab-item.active::after {

content: '';

position: absolute;

bottom: 0;

left: 50%;

transform: translateX(-50%);

width: 60rpx;

height: 4rpx;

background-color: var(--primary-color);

border-radius: 2rpx;

}

.tab-content {

padding: 32rpx;

}

/\* 指标卡片网格 \*/

.metrics-grid {

display: grid;

grid-template-columns: 1fr 1fr;

gap: 16rpx;

margin-bottom: 32rpx;

}

.metric-card {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

border-radius: 16rpx;

padding: 24rpx;

color: #fff;

position: relative;

overflow: hidden;

}

.metric-card:nth-child(2) {

background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);

}

.metric-card:nth-child(3) {

background: linear-gradient(135deg, #4facfe 0%, #00f2fe 100%);

}

.metric-card:nth-child(4) {

background: linear-gradient(135deg, #43e97b 0%, #38f9d7 100%);

}

.metric-value {

font-size: 40rpx;

font-weight: 700;

margin-bottom: 8rpx;

}

.metric-label {

font-size: 24rpx;

opacity: 0.9;

margin-bottom: 8rpx;

}

.metric-change {

display: flex;

align-items: center;

gap: 4rpx;

font-size: 22rpx;

opacity: 0.8;

}

.change-icon {

width: 20rpx;

height: 20rpx;

}

.metric-change.positive {

color: #fff;

}

/\* 进度部分 \*/

.progress-section {

margin-bottom: 32rpx;

}

.section-title {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 16rpx;

font-size: 32rpx;

font-weight: 600;

color: #333;

}

.target-action {

font-size: 28rpx;

color: var(--primary-color);

font-weight: normal;

}

.progress-card {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

border-radius: 16rpx;

padding: 32rpx;

color: #fff;

}

.progress-info {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 16rpx;

}

.progress-text {

display: flex;

align-items: baseline;

gap: 8rpx;

}

.current-amount {

font-size: 36rpx;

font-weight: 700;

}

.target-amount {

font-size: 28rpx;

opacity: 0.8;

}

.progress-percent {

font-size: 32rpx;

font-weight: 600;

}

.progress-bar {

height: 12rpx;

background-color: rgba(255, 255, 255, 0.3);

border-radius: 6rpx;

overflow: hidden;

}

.progress-fill {

height: 100%;

background-color: #fff;

border-radius: 6rpx;

transition: width 0.5s ease;

}

/\* 快速操作 \*/

.quick-actions {

margin-bottom: 32rpx;

}

.action-grid {

display: grid;

grid-template-columns: repeat(4, 1fr);

gap: 16rpx;

}

.action-item {

display: flex;

flex-direction: column;

align-items: center;

padding: 24rpx 16rpx;

background-color: #f8f9fa;

border-radius: 16rpx;

border: 1rpx solid #e9ecef;

transition: all 0.3s ease;

}

.action-item:active {

background-color: #e9ecef;

transform: scale(0.95);

}

.action-item .action-icon {

width: 48rpx;

height: 48rpx;

margin-bottom: 8rpx;

}

.action-text {

font-size: 24rpx;

color: #666;

}

/\* 图表部分 \*/

.chart-section {

margin-bottom: 32rpx;

}

.chart-container {

height: 400rpx;

background-color: #f8f9fa;

border-radius: 12rpx;

margin-top: 16rpx;

}

.distribution-tabs {

display: flex;

gap: 16rpx;

margin-bottom: 16rpx;

}

.dist-tab {

padding: 8rpx 16rpx;

background-color: #f8f9fa;

border-radius: 12rpx;

font-size: 24rpx;

color: #666;

border: 1rpx solid #e9ecef;

transition: all 0.3s ease;

}

.dist-tab.active {

background-color: var(--primary-color);

color: #fff;

border-color: var(--primary-color);

}

.chart-placeholder {

height: 100%;

display: flex;

align-items: center;

justify-content: center;

color: #999;

font-size: 28rpx;

}

/\* 排行榜 \*/

.ranking-section {

margin-bottom: 32rpx;

}

.ranking-list {

background-color: #f8f9fa;

border-radius: 12rpx;

padding: 24rpx;

margin-top: 16rpx;

}

.ranking-item {

display: flex;

align-items: center;

padding: 16rpx 0;

border-bottom: 1rpx solid #e9ecef;

}

.ranking-item:last-child {

border-bottom: none;

}

.ranking-number {

width: 48rpx;

height: 48rpx;

display: flex;

align-items: center;

justify-content: center;

border-radius: 50%;

font-size: 24rpx;

font-weight: 600;

margin-right: 16rpx;

}

.ranking-number.first {

background-color: #ffd700;

color: #fff;

}

.ranking-number.second {

background-color: #c0c0c0;

color: #fff;

}

.ranking-number.third {

background-color: #cd7f32;

color: #fff;

}

.ranking-number.other {

background-color: #e9ecef;

color: #666;

}

.ranking-info {

flex: 1;

margin-right: 16rpx;

}

.ranking-name {

font-size: 28rpx;

font-weight: 600;

color: #333;

margin-bottom: 4rpx;

}

.ranking-dept {

font-size: 24rpx;

color: #999;

}

.ranking-score {

font-size: 32rpx;

font-weight: 700;

color: var(--primary-color);

}

/\* 个性化信息 \*/

.personal-info {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

border-radius: 16rpx;

padding: 32rpx;

color: #fff;

margin-bottom: 32rpx;

}

.personal-header {

display: flex;

align-items: center;

margin-bottom: 24rpx;

}

.personal-avatar {

width: 80rpx;

height: 80rpx;

border-radius: 50%;

margin-right: 16rpx;

background-color: rgba(255, 255, 255, 0.2);

display: flex;

align-items: center;

justify-content: center;

font-size: 32rpx;

}

.personal-name {

font-size: 32rpx;

font-weight: 600;

margin-bottom: 4rpx;

}

.personal-position {

font-size: 24rpx;

opacity: 0.9;

}

.personal-stats {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 16rpx;

}

.personal-stat {

text-align: center;

}

.personal-stat-value {

font-size: 36rpx;

font-weight: 700;

margin-bottom: 4rpx;

}

.personal-stat-label {

font-size: 22rpx;

opacity: 0.8;

}

/\* 目标设置 \*/

.target-setting {

background-color: #f8f9fa;

border-radius: 12rpx;

padding: 24rpx;

margin-top: 16rpx;

}

.target-form {

display: flex;

align-items: center;

gap: 16rpx;

}

.target-input {

flex: 1;

height: 72rpx;

padding: 0 16rpx;

background-color: #fff;

border: 1rpx solid #e9ecef;

border-radius: 8rpx;

font-size: 28rpx;

}

.target-btn {

height: 72rpx;

padding: 0 24rpx;

background-color: var(--primary-color);

color: #fff;

border-radius: 8rpx;

font-size: 28rpx;

border: none;

}

.target-btn:active {

background-color: var(--primary-color-dark);

}

/\* 趋势图表 \*/

.trend-chart {

height: 300rpx;

background-color: #f8f9fa;

border-radius: 12rpx;

margin-top: 16rpx;

position: relative;

}

.chart-legend {

display: flex;

justify-content: center;

gap: 32rpx;

padding: 16rpx;

background-color: #fff;

border-radius: 12rpx;

margin-top: 16rpx;

}

.legend-item {

display: flex;

align-items: center;

gap: 8rpx;

font-size: 24rpx;

color: #666;

}

.legend-dot {

width: 16rpx;

height: 16rpx;

border-radius: 50%;

}

.legend-dot.current {

background-color: var(--primary-color);

}

.legend-dot.target {

background-color: #f093fb;

}

.legend-dot.last-year {

background-color: #4facfe;

}

/\* 空状态 \*/

.empty-state {

text-align: center;

padding: 80rpx 32rpx;

color: #999;

}

.empty-icon {

width: 120rpx;

height: 120rpx;

margin: 0 auto 24rpx;

opacity: 0.5;

}

.empty-text {

font-size: 28rpx;

margin-bottom: 16rpx;

}

.empty-desc {

font-size: 24rpx;

color: #ccc;

}

/\* 刷新按钮 \*/

.refresh-btn {

position: fixed;

right: 32rpx;

bottom: 160rpx;

width: 88rpx;

height: 88rpx;

background-color: var(--primary-color);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 4rpx 12rpx rgba(0, 0, 0, 0.15);

z-index: 100;

}

.refresh-btn:active {

transform: scale(0.95);

}

.refresh-icon {

width: 48rpx;

height: 48rpx;

color: #fff;

}

/\* 加载状态 \*/

.loading-overlay {

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(255, 255, 255, 0.8);

display: flex;

align-items: center;

justify-content: center;

border-radius: 16rpx;

}

.loading-spinner {

width: 48rpx;

height: 48rpx;

border: 4rpx solid #e9ecef;

border-top: 4rpx solid var(--primary-color);

border-radius: 50%;

animation: spin 1s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

/\* 响应式适配 \*/

@media (max-width: 375px) {

.metrics-grid {

grid-template-columns: 1fr;

gap: 12rpx;

}

.action-grid {

grid-template-columns: repeat(2, 1fr);

}

.personal-stats {

grid-template-columns: 1fr;

gap: 12rpx;

}

}

/\* 主题色变量 \*/

:root {

--primary-color: #667eea;

--primary-color-dark: #5a6fd8;

--success-color: #43e97b;

--warning-color: #f093fb;

--info-color: #4facfe;

--danger-color: #f5576c;

}

现在继续开发下一个模块：**业绩统计页面配置文件 - pages/manager/performance/performance.json**

业绩统计页面配置文件 - performance.json

{

"navigationBarTitleText": "业绩统计",

"navigationBarBackgroundColor": "#667eea",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f7fa",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"backgroundTextStyle": "dark",

"usingComponents": {

"van-tabs": "@vant/weapp/tabs/index",

"van-tab": "@vant/weapp/tab/index",

"van-picker": "@vant/weapp/picker/index",

"van-popup": "@vant/weapp/popup/index",

"van-loading": "@vant/weapp/loading/index",

"van-empty": "@vant/weapp/empty/index",

"van-button": "@vant/weapp/button/index",

"van-field": "@vant/weapp/field/index",

"van-cell": "@vant/weapp/cell/index",

"van-cell-group": "@vant/weapp/cell-group/index",

"van-toast": "@vant/weapp/toast/index"

}

}

接下来开发：**小程序公共组件和工具类** 模块

首先创建 **API 工具类 - utils/api.js**：

API 工具类 - utils/api.js

// API 工具类

const config = require('./config')

const { showToast, showLoading, hideLoading } = require('./common')

class ApiService {

constructor() {

this.baseURL = config.apiBaseURL

this.timeout = config.timeout || 10000

}

// 请求拦截器

interceptRequest(options) {

// 添加 loading

if (options.showLoading !== false) {

showLoading(options.loadingText || '请求中...')

}

// 添加认证头

const token = wx.getStorageSync('token')

if (token) {

options.header = {

...options.header,

'Authorization': `Bearer ${token}`

}

}

// 添加通用头部

options.header = {

'Content-Type': 'application/json',

...options.header

}

return options

}

// 响应拦截器

interceptResponse(res, options) {

hideLoading()

const { statusCode, data } = res

// HTTP 状态码检查

if (statusCode !== 200) {

const message = this.getErrorMessage(statusCode)

if (options.showError !== false) {

showToast(message, 'error')

}

return Promise.reject(new Error(message))

}

// 业务状态码检查

if (data.code !== undefined && data.code !== 0) {

if (data.code === 401) {

// token 失效，跳转登录

this.handleAuthError()

return Promise.reject(new Error('登录已失效'))

}

const message = data.message || '请求失败'

if (options.showError !== false) {

showToast(message, 'error')

}

return Promise.reject(new Error(message))

}

return data.data || data

}

// 处理认证错误

handleAuthError() {

wx.removeStorageSync('token')

wx.removeStorageSync('userInfo')

wx.showModal({

title: '提示',

content: '登录已失效，请重新登录',

showCancel: false,

success: () => {

wx.reLaunch({

url: '/pages/auth/login/login'

})

}

})

}

// 获取错误信息

getErrorMessage(statusCode) {

const errorMap = {

400: '请求参数错误',

401: '未授权访问',

403: '禁止访问',

404: '请求地址不存在',

405: '请求方法不允许',

408: '请求超时',

500: '服务器内部错误',

502: '网关错误',

503: '服务不可用',

504: '网关超时'

}

return errorMap[statusCode] || '网络请求失败'

}

// 基础请求方法

request(options) {

return new Promise((resolve, reject) => {

const requestOptions = this.interceptRequest({

url: this.baseURL + options.url,

method: options.method || 'GET',

data: options.data,

header: options.header || {},

timeout: options.timeout || this.timeout,

showLoading: options.showLoading,

showError: options.showError,

loadingText: options.loadingText

})

wx.request({

...requestOptions,

success: (res) => {

this.interceptResponse(res, options)

.then(resolve)

.catch(reject)

},

fail: (err) => {

hideLoading()

const message = err.errMsg || '网络请求失败'

if (options.showError !== false) {

showToast(message, 'error')

}

reject(err)

}

})

})

}

// GET 请求

get(url, params = {}, options = {}) {

const queryString = Object.keys(params)

.map(key => `${encodeURIComponent(key)}=${encodeURIComponent(params[key])}`)

.join('&')

const fullUrl = queryString ? `${url}?${queryString}` : url

return this.request({

url: fullUrl,

method: 'GET',

...options

})

}

// POST 请求

post(url, data = {}, options = {}) {

return this.request({

url,

method: 'POST',

data,

...options

})

}

// PUT 请求

put(url, data = {}, options = {}) {

return this.request({

url,

method: 'PUT',

data,

...options

})

}

// DELETE 请求

delete(url, options = {}) {

return this.request({

url,

method: 'DELETE',

...options

})

}

// 文件上传

uploadFile(url, filePath, name = 'file', formData = {}, options = {}) {

return new Promise((resolve, reject) => {

if (options.showLoading !== false) {

showLoading('上传中...')

}

const token = wx.getStorageSync('token')

const header = {

'Authorization': token ? `Bearer ${token}` : '',

...options.header

}

wx.uploadFile({

url: this.baseURL + url,

filePath,

name,

formData,

header,

success: (res) => {

hideLoading()

try {

const data = JSON.parse(res.data)

if (data.code === 0) {

resolve(data.data || data)

} else {

const message = data.message || '上传失败'

if (options.showError !== false) {

showToast(message, 'error')

}

reject(new Error(message))

}

} catch (err) {

if (options.showError !== false) {

showToast('上传失败', 'error')

}

reject(err)

}

},

fail: (err) => {

hideLoading()

const message = err.errMsg || '上传失败'

if (options.showError !== false) {

showToast(message, 'error')

}

reject(err)

}

})

})

}

}

// 创建实例

const apiService = new ApiService()

// 用户认证相关 API

const authAPI = {

// 微信登录

wechatLogin: (code, userInfo) => {

return apiService.post('/auth/wechat-login', { code, userInfo })

},

// 手机号绑定

bindPhone: (phoneNumber, verifyCode) => {

return apiService.post('/auth/bind-phone', { phoneNumber, verifyCode })

},

// 发送验证码

sendVerifyCode: (phoneNumber) => {

return apiService.post('/auth/send-code', { phoneNumber })

},

// 用户注册

register: (userData) => {

return apiService.post('/auth/register', userData)

},

// 身份认证

submitAuth: (authData) => {

return apiService.post('/auth/verify', authData)

},

// 获取用户信息

getUserInfo: () => {

return apiService.get('/auth/me')

},

// 更新用户信息

updateUserInfo: (userData) => {

return apiService.put('/auth/me', userData)

}

}

// 产品相关 API

const productAPI = {

// 获取产品列表

getProducts: (params) => {

return apiService.get('/products', params)

},

// 获取产品详情

getProductDetail: (id) => {

return apiService.get(`/products/${id}`)

},

// 节电效益计算

calculateSavings: (params) => {

return apiService.post('/products/calculate', params)

}

}

// 订单相关 API

const orderAPI = {

// 获取订单列表

getOrders: (params) => {

return apiService.get('/orders', params)

},

// 获取订单详情

getOrderDetail: (id) => {

return apiService.get(`/orders/${id}`)

},

// 创建订单

createOrder: (orderData) => {

return apiService.post('/orders', orderData)

},

// 更新订单

updateOrder: (id, orderData) => {

return apiService.put(`/orders/${id}`, orderData)

},

// 取消订单

cancelOrder: (id, reason) => {

return apiService.post(`/orders/${id}/cancel`, { reason })

}

}

// 客户管理相关 API

const customerAPI = {

// 获取我的客户列表

getMyCustomers: (params) => {

return apiService.get('/customers/my-customers', params)

},

// 获取客户详情

getCustomerDetail: (id) => {

return apiService.get(`/customers/${id}`)

},

// 添加跟进记录

addFollowRecord: (customerId, recordData) => {

return apiService.post(`/customers/${customerId}/follow`, recordData)

},

// 获取跟进记录

getFollowRecords: (customerId, params) => {

return apiService.get(`/customers/${customerId}/follow`, params)

},

// 更新客户状态

updateCustomerStatus: (customerId, status) => {

return apiService.put(`/customers/${customerId}/status`, { status })

}

}

// 业绩统计相关 API

const performanceAPI = {

// 获取个人业绩统计

getPersonalPerformance: (params) => {

return apiService.get('/performance/personal', params)

},

// 获取团队业绩统计

getTeamPerformance: (params) => {

return apiService.get('/performance/team', params)

},

// 获取业绩排行榜

getRanking: (params) => {

return apiService.get('/performance/ranking', params)

},

// 设置业绩目标

setTarget: (targetData) => {

return apiService.post('/performance/target', targetData)

}

}

// 文件上传相关 API

const uploadAPI = {

// 上传头像

uploadAvatar: (filePath) => {

return apiService.uploadFile('/upload/avatar', filePath, 'avatar')

},

// 上传营业执照

uploadBusinessLicense: (filePath) => {

return apiService.uploadFile('/upload/business-license', filePath, 'license')

},

// 上传身份证

uploadIdCard: (filePath, type) => {

return apiService.uploadFile('/upload/id-card', filePath, 'idcard', { type })

},

// 上传附件

uploadAttachment: (filePath, fileName) => {

return apiService.uploadFile('/upload/attachment', filePath, 'file', { fileName })

}

}

module.exports = {

apiService,

authAPI,

productAPI,

orderAPI,

customerAPI,

performanceAPI,

uploadAPI

}

接下来创建 **通用工具类 - utils/common.js**：

通用工具类 - utils/common.js

// 通用工具类

// 显示消息提示

function showToast(title, icon = 'none', duration = 2000) {

wx.showToast({

title: title,

icon: icon === 'error' ? 'none' : icon,

duration: duration,

mask: true

})

}

// 显示加载提示

function showLoading(title = '加载中...') {

wx.showLoading({

title: title,

mask: true

})

}

// 隐藏加载提示

function hideLoading() {

wx.hideLoading()

}

// 显示确认对话框

function showConfirm(content, title = '提示') {

return new Promise((resolve) => {

wx.showModal({

title: title,

content: content,

success: (res) => {

resolve(res.confirm)

},

fail: () => {

resolve(false)

}

})

})

}

// 显示操作菜单

function showActionSheet(itemList) {

return new Promise((resolve, reject) => {

wx.showActionSheet({

itemList: itemList,

success: (res) => {

resolve(res.tapIndex)

},

fail: (err) => {

reject(err)

}

})

})

}

// 格式化日期

function formatDate(date, format = 'YYYY-MM-DD') {

if (!date) return ''

const d = typeof date === 'string' ? new Date(date) : date

if (isNaN(d.getTime())) return ''

const year = d.getFullYear()

const month = String(d.getMonth() + 1).padStart(2, '0')

const day = String(d.getDate()).padStart(2, '0')

const hour = String(d.getHours()).padStart(2, '0')

const minute = String(d.getMinutes()).padStart(2, '0')

const second = String(d.getSeconds()).padStart(2, '0')

return format

.replace('YYYY', year)

.replace('MM', month)

.replace('DD', day)

.replace('HH', hour)

.replace('mm', minute)

.replace('ss', second)

}

// 格式化相对时间

function formatRelativeTime(date) {

if (!date) return ''

const d = typeof date === 'string' ? new Date(date) : date

const now = new Date()

const diff = now.getTime() - d.getTime()

const seconds = Math.floor(diff / 1000)

const minutes = Math.floor(seconds / 60)

const hours = Math.floor(minutes / 60)

const days = Math.floor(hours / 24)

if (days > 7) {

return formatDate(d, 'MM-DD')

} else if (days > 0) {

return `${days}天前`

} else if (hours > 0) {

return `${hours}小时前`

} else if (minutes > 0) {

return `${minutes}分钟前`

} else {

return '刚刚'

}

}

// 格式化金额

function formatMoney(amount, precision = 2) {

if (amount === null || amount === undefined || isNaN(amount)) {

return '0.00'

}

const num = parseFloat(amount)

return num.toLocaleString('zh-CN', {

minimumFractionDigits: precision,

maximumFractionDigits: precision

})

}

// 格式化数字（添加千分位）

function formatNumber(num) {

if (num === null || num === undefined || isNaN(num)) {

return '0'

}

return parseInt(num).toLocaleString('zh-CN')

}

// 格式化百分比

function formatPercent(value, precision = 1) {

if (value === null || value === undefined || isNaN(value)) {

return '0%'

}

return `${(parseFloat(value) \* 100).toFixed(precision)}%`

}

// 验证手机号

function validatePhone(phone) {

const reg = /^1[3-9]\d{9}$/

return reg.test(phone)

}

// 验证邮箱

function validateEmail(email) {

const reg = /^[^\s@]+@[^\s@]+\.[^\s@]+$/

return reg.test(email)

}

// 验证身份证号

function validateIdCard(idCard) {

const reg = /(^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$)/

return reg.test(idCard)

}

// 生成UUID

function generateUUID() {

return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {

const r = Math.random() \* 16 | 0

const v = c === 'x' ? r : (r & 0x3 | 0x8)

return v.toString(16)

})

}

// 防抖函数

function debounce(func, wait) {

let timeout

return function executedFunction(...args) {

const later = () => {

clearTimeout(timeout)

func(...args)

}

clearTimeout(timeout)

timeout = setTimeout(later, wait)

}

}

// 节流函数

function throttle(func, limit) {

let inThrottle

return function(...args) {

if (!inThrottle) {

func.apply(this, args)

inThrottle = true

setTimeout(() => inThrottle = false, limit)

}

}

}

// 深拷贝

function deepClone(obj) {

if (obj === null || typeof obj !== 'object') {

return obj

}

if (obj instanceof Date) {

return new Date(obj.getTime())

}

if (obj instanceof Array) {

return obj.map(item => deepClone(item))

}

if (typeof obj === 'object') {

const clonedObj = {}

for (let key in obj) {

if (obj.hasOwnProperty(key)) {

clonedObj[key] = deepClone(obj[key])

}

}

return clonedObj

}

return obj

}

// 获取设备信息

function getSystemInfo() {

return new Promise((resolve) => {

wx.getSystemInfo({

success: (res) => {

resolve(res)

},

fail: () => {

resolve({})

}

})

})

}

// 获取网络状态

function getNetworkType() {

return new Promise((resolve) => {

wx.getNetworkType({

success: (res) => {

resolve(res.networkType)

},

fail: () => {

resolve('unknown')

}

})

})

}

// 跳转到页面

function navigateTo(url, params = {}) {

let queryString = ''

if (Object.keys(params).length > 0) {

queryString = '?' + Object.keys(params)

.map(key => `${key}=${encodeURIComponent(params[key])}`)

.join('&')

}

wx.navigateTo({

url: url + queryString,

fail: (err) => {

console.error('页面跳转失败:', err)

showToast('页面跳转失败', 'error')

}

})

}

// 返回上一页

function navigateBack(delta = 1) {

wx.navigateBack({

delta: delta,

fail: () => {

wx.switchTab({

url: '/pages/index/index'

})

}

})

}

// 切换到Tab页面

function switchTab(url) {

wx.switchTab({

url: url,

fail: (err) => {

console.error('Tab页面切换失败:', err)

}

})

}

// 重定向到页面

function redirectTo(url, params = {}) {

let queryString = ''

if (Object.keys(params).length > 0) {

queryString = '?' + Object.keys(params)

.map(key => `${key}=${encodeURIComponent(params[key])}`)

.join('&')

}

wx.redirectTo({

url: url + queryString,

fail: (err) => {

console.error('页面重定向失败:', err)

}

})

}

// 设置页面标题

function setNavigationBarTitle(title) {

wx.setNavigationBarTitle({

title: title

})

}

// 显示分享菜单

function showShareMenu() {

wx.showShareMenu({

withShareTicket: true

})

}

// 隐藏分享菜单

function hideShareMenu() {

wx.hideShareMenu()

}

// 复制到剪贴板

function setClipboardData(data) {

return new Promise((resolve, reject) => {

wx.setClipboardData({

data: data,

success: () => {

showToast('复制成功')

resolve(true)

},

fail: (err) => {

showToast('复制失败', 'error')

reject(err)

}

})

})

}

// 获取剪贴板内容

function getClipboardData() {

return new Promise((resolve, reject) => {

wx.getClipboardData({

success: (res) => {

resolve(res.data)

},

fail: (err) => {

reject(err)

}

})

})

}

// 拨打电话

function makePhoneCall(phoneNumber) {

wx.makePhoneCall({

phoneNumber: phoneNumber,

fail: (err) => {

console.error('拨打电话失败:', err)

showToast('拨打电话失败', 'error')

}

})

}

// 预览图片

function previewImage(current, urls = []) {

wx.previewImage({

current: current,

urls: urls.length > 0 ? urls : [current]

})

}

// 选择图片

function chooseImage(count = 1, sizeType = ['compressed'], sourceType = ['album', 'camera']) {

return new Promise((resolve, reject) => {

wx.chooseImage({

count: count,

sizeType: sizeType,

sourceType: sourceType,

success: (res) => {

resolve(res.tempFilePaths)

},

fail: (err) => {

reject(err)

}

})

})

}

// 上传文件

function uploadFile(url, filePath, name = 'file', formData = {}) {

return new Promise((resolve, reject) => {

const uploadTask = wx.uploadFile({

url: url,

filePath: filePath,

name: name,

formData: formData,

success: (res) => {

try {

const data = JSON.parse(res.data)

resolve(data)

} catch (e) {

resolve(res.data)

}

},

fail: (err) => {

reject(err)

}

})

return uploadTask

})

}

// 下载文件

function downloadFile(url) {

return new Promise((resolve, reject) => {

wx.downloadFile({

url: url,

success: (res) => {

if (res.statusCode === 200) {

resolve(res.tempFilePath)

} else {

reject(new Error('下载失败'))

}

},

fail: (err) => {

reject(err)

}

})

})

}

// 保存图片到相册

function saveImageToPhotosAlbum(filePath) {

return new Promise((resolve, reject) => {

wx.saveImageToPhotosAlbum({

filePath: filePath,

success: () => {

showToast('保存成功')

resolve(true)

},

fail: (err) => {

if (err.errMsg.includes('auth deny')) {

showToast('请授权访问相册', 'error')

} else {

showToast('保存失败', 'error')

}

reject(err)

}

})

})

}

// 震动反馈

function vibrateShort() {

wx.vibrateShort({

fail: () => {

// 静默失败

}

})

}

// 长震动反馈

function vibrateLong() {

wx.vibrateLong({

fail: () => {

// 静默失败

}

})

}

// 获取位置信息

function getLocation(type = 'wgs84') {

return new Promise((resolve, reject) => {

wx.getLocation({

type: type,

success: (res) => {

resolve({

latitude: res.latitude,

longitude: res.longitude,

speed: res.speed,

accuracy: res.accuracy

})

},

fail: (err) => {

reject(err)

}

})

})

}

// 打开地图选择位置

function chooseLocation() {

return new Promise((resolve, reject) => {

wx.chooseLocation({

success: (res) => {

resolve({

name: res.name,

address: res.address,

latitude: res.latitude,

longitude: res.longitude

})

},

fail: (err) => {

reject(err)

}

})

})

}

// 打开地图查看位置

function openLocation(latitude, longitude, name = '', address = '') {

wx.openLocation({

latitude: latitude,

longitude: longitude,

name: name,

address: address,

fail: (err) => {

console.error('打开地图失败:', err)

showToast('打开地图失败', 'error')

}

})

}

// 获取微信运动步数

function getWeRunData() {

return new Promise((resolve, reject) => {

wx.getWeRunData({

success: (res) => {

resolve(res.encryptedData)

},

fail: (err) => {

reject(err)

}

})

})

}

// 数组去重

function uniqueArray(arr, key = null) {

if (!Array.isArray(arr)) return []

if (key) {

const seen = new Map()

return arr.filter(item => {

const value = item[key]

if (seen.has(value)) {

return false

}

seen.set(value, true)

return true

})

}

return [...new Set(arr)]

}

// 数组排序

function sortArray(arr, key = null, order = 'asc') {

if (!Array.isArray(arr)) return []

return arr.sort((a, b) => {

let valueA = key ? a[key] : a

let valueB = key ? b[key] : b

// 处理数字类型

if (typeof valueA === 'string' && !isNaN(valueA)) {

valueA = parseFloat(valueA)

}

if (typeof valueB === 'string' && !isNaN(valueB)) {

valueB = parseFloat(valueB)

}

if (order === 'desc') {

return valueB > valueA ? 1 : valueB < valueA ? -1 : 0

} else {

return valueA > valueB ? 1 : valueA < valueB ? -1 : 0

}

})

}

// 检查是否为空值

function isEmpty(value) {

if (value === null || value === undefined) return true

if (typeof value === 'string') return value.trim() === ''

if (Array.isArray(value)) return value.length === 0

if (typeof value === 'object') return Object.keys(value).length === 0

return false

}

// 安全的JSON解析

function safeJsonParse(str, defaultValue = null) {

try {

return JSON.parse(str)

} catch (e) {

console.warn('JSON解析失败:', e)

return defaultValue

}

}

// 安全的JSON字符串化

function safeJsonStringify(obj, defaultValue = '{}') {

try {

return JSON.stringify(obj)

} catch (e) {

console.warn('JSON字符串化失败:', e)

return defaultValue

}

}

// 导出所有工具函数

module.exports = {

// UI工具

showToast,

showLoading,

hideLoading,

showConfirm,

showActionSheet,

// 格式化工具

formatDate,

formatRelativeTime,

formatMoney,

formatNumber,

formatPercent,

// 验证工具

validatePhone,

validateEmail,

validateIdCard,

// 通用工具

generateUUID,

debounce,

throttle,

deepClone,

// 系统工具

getSystemInfo,

getNetworkType,

// 导航工具

navigateTo,

navigateBack,

switchTab,

redirectTo,

setNavigationBarTitle,

// 分享工具

showShareMenu,

hideShareMenu,

// 剪贴板工具

setClipboardData,

getClipboardData,

// 设备功能

makePhoneCall,

vibrateShort,

vibrateLong,

// 媒体工具

previewImage,

chooseImage,

uploadFile,

downloadFile,

saveImageToPhotosAlbum,

// 位置工具

getLocation,

chooseLocation,

openLocation,

// 微信功能

getWeRunData,

// 数据处理工具

uniqueArray,

sortArray,

isEmpty,

safeJsonParse,

safeJsonStringify

}

现在继续开发下一个模块：**本地存储工具类 - utils/storage.js**

本地存储工具类 - utils/storage.js

// 本地存储工具类

// 存储数据

function setStorage(key, data) {

try {

wx.setStorageSync(key, data)

return true

} catch (error) {

console.error('存储数据失败:', error)

return false

}

}

// 异步存储数据

function setStorageAsync(key, data) {

return new Promise((resolve, reject) => {

wx.setStorage({

key: key,

data: data,

success: () => {

resolve(true)

},

fail: (error) => {

console.error('异步存储数据失败:', error)

reject(error)

}

})

})

}

// 获取数据

function getStorage(key, defaultValue = null) {

try {

const data = wx.getStorageSync(key)

return data !== '' ? data : defaultValue

} catch (error) {

console.error('获取存储数据失败:', error)

return defaultValue

}

}

// 异步获取数据

function getStorageAsync(key, defaultValue = null) {

return new Promise((resolve) => {

wx.getStorage({

key: key,

success: (res) => {

resolve(res.data)

},

fail: () => {

resolve(defaultValue)

}

})

})

}

// 删除数据

function removeStorage(key) {

try {

wx.removeStorageSync(key)

return true

} catch (error) {

console.error('删除存储数据失败:', error)

return false

}

}

// 异步删除数据

function removeStorageAsync(key) {

return new Promise((resolve, reject) => {

wx.removeStorage({

key: key,

success: () => {

resolve(true)

},

fail: (error) => {

console.error('异步删除存储数据失败:', error)

reject(error)

}

})

})

}

// 清空所有存储

function clearStorage() {

try {

wx.clearStorageSync()

return true

} catch (error) {

console.error('清空存储失败:', error)

return false

}

}

// 异步清空所有存储

function clearStorageAsync() {

return new Promise((resolve, reject) => {

wx.clearStorage({

success: () => {

resolve(true)

},

fail: (error) => {

console.error('异步清空存储失败:', error)

reject(error)

}

})

})

}

// 获取存储信息

function getStorageInfo() {

try {

return wx.getStorageInfoSync()

} catch (error) {

console.error('获取存储信息失败:', error)

return {

keys: [],

currentSize: 0,

limitSize: 0

}

}

}

// 异步获取存储信息

function getStorageInfoAsync() {

return new Promise((resolve, reject) => {

wx.getStorageInfo({

success: (res) => {

resolve(res)

},

fail: (error) => {

console.error('异步获取存储信息失败:', error)

reject(error)

}

})

})

}

// 检查key是否存在

function hasStorage(key) {

try {

const data = wx.getStorageSync(key)

return data !== ''

} catch (error) {

return false

}

}

// 批量设置存储

function setBatchStorage(dataMap) {

const results = {}

for (const key in dataMap) {

if (dataMap.hasOwnProperty(key)) {

results[key] = setStorage(key, dataMap[key])

}

}

return results

}

// 批量获取存储

function getBatchStorage(keys, defaultValue = null) {

const results = {}

keys.forEach(key => {

results[key] = getStorage(key, defaultValue)

})

return results

}

// 批量删除存储

function removeBatchStorage(keys) {

const results = {}

keys.forEach(key => {

results[key] = removeStorage(key)

})

return results

}

// 存储JSON数据

function setJsonStorage(key, data) {

try {

const jsonString = JSON.stringify(data)

return setStorage(key, jsonString)

} catch (error) {

console.error('存储JSON数据失败:', error)

return false

}

}

// 获取JSON数据

function getJsonStorage(key, defaultValue = null) {

try {

const jsonString = getStorage(key)

if (jsonString) {

return JSON.parse(jsonString)

}

return defaultValue

} catch (error) {

console.error('获取JSON数据失败:', error)

return defaultValue

}

}

// 存储用户相关的常用数据操作

// 存储用户信息

function setUserInfo(userInfo) {

return setStorage('userInfo', userInfo)

}

// 获取用户信息

function getUserInfo() {

return getStorage('userInfo', null)

}

// 清除用户信息

function clearUserInfo() {

return removeStorage('userInfo')

}

// 存储用户token

function setToken(token) {

return setStorage('token', token)

}

// 获取用户token

function getToken() {

return getStorage('token', '')

}

// 清除用户token

function clearToken() {

return removeStorage('token')

}

// 存储应用设置

function setAppSettings(settings) {

return setJsonStorage('appSettings', settings)

}

// 获取应用设置

function getAppSettings() {

return getJsonStorage('appSettings', {

theme: 'light',

language: 'zh-CN',

notifications: true,

autoLogin: true

})

}

// 更新应用设置

function updateAppSettings(newSettings) {

const currentSettings = getAppSettings()

const updatedSettings = Object.assign(currentSettings, newSettings)

return setAppSettings(updatedSettings)

}

// 存储搜索历史

function addSearchHistory(keyword) {

if (!keyword || keyword.trim() === '') return false

let history = getJsonStorage('searchHistory', [])

// 移除重复项

history = history.filter(item => item !== keyword)

// 添加到开头

history.unshift(keyword)

// 限制历史记录数量

if (history.length > 20) {

history = history.slice(0, 20)

}

return setJsonStorage('searchHistory', history)

}

// 获取搜索历史

function getSearchHistory() {

return getJsonStorage('searchHistory', [])

}

// 清除搜索历史

function clearSearchHistory() {

return removeStorage('searchHistory')

}

// 删除单个搜索历史

function removeSearchHistoryItem(keyword) {

let history = getSearchHistory()

history = history.filter(item => item !== keyword)

return setJsonStorage('searchHistory', history)

}

// 存储浏览历史

function addBrowseHistory(item) {

if (!item || !item.id) return false

let history = getJsonStorage('browseHistory', [])

// 移除重复项

history = history.filter(historyItem => historyItem.id !== item.id)

// 添加时间戳

item.browseTime = new Date().getTime()

// 添加到开头

history.unshift(item)

// 限制历史记录数量

if (history.length > 50) {

history = history.slice(0, 50)

}

return setJsonStorage('browseHistory', history)

}

// 获取浏览历史

function getBrowseHistory() {

return getJsonStorage('browseHistory', [])

}

// 清除浏览历史

function clearBrowseHistory() {

return removeStorage('browseHistory')

}

// 存储草稿数据

function saveDraft(key, data) {

const drafts = getJsonStorage('drafts', {})

drafts[key] = {

data: data,

saveTime: new Date().getTime()

}

return setJsonStorage('drafts', drafts)

}

// 获取草稿数据

function getDraft(key) {

const drafts = getJsonStorage('drafts', {})

return drafts[key] ? drafts[key].data : null

}

// 删除草稿数据

function removeDraft(key) {

const drafts = getJsonStorage('drafts', {})

delete drafts[key]

return setJsonStorage('drafts', drafts)

}

// 获取所有草稿

function getAllDrafts() {

return getJsonStorage('drafts', {})

}

// 清除所有草稿

function clearAllDrafts() {

return removeStorage('drafts')

}

// 清除过期草稿（7天前的）

function clearExpiredDrafts() {

const drafts = getAllDrafts()

const expireTime = new Date().getTime() - 7 \* 24 \* 60 \* 60 \* 1000 // 7天

for (const key in drafts) {

if (drafts[key].saveTime < expireTime) {

delete drafts[key]

}

}

return setJsonStorage('drafts', drafts)

}

// 存储缓存数据（带过期时间）

function setCache(key, data, expireTime = null) {

const cacheData = {

data: data,

createTime: new Date().getTime(),

expireTime: expireTime

}

return setStorage(`cache\_${key}`, cacheData)

}

// 获取缓存数据

function getCache(key) {

const cacheData = getStorage(`cache\_${key}`)

if (!cacheData) return null

// 检查是否过期

if (cacheData.expireTime && new Date().getTime() > cacheData.expireTime) {

removeStorage(`cache\_${key}`)

return null

}

return cacheData.data

}

// 清除过期缓存

function clearExpiredCache() {

const storageInfo = getStorageInfo()

const currentTime = new Date().getTime()

storageInfo.keys.forEach(key => {

if (key.startsWith('cache\_')) {

const cacheData = getStorage(key)

if (cacheData && cacheData.expireTime && currentTime > cacheData.expireTime) {

removeStorage(key)

}

}

})

}

// 获取存储使用情况

function getStorageUsage() {

const info = getStorageInfo()

return {

used: info.currentSize,

total: info.limitSize,

usage: info.limitSize > 0 ? (info.currentSize / info.limitSize \* 100).toFixed(2) + '%' : '0%',

keys: info.keys

}

}

module.exports = {

// 基础存储操作

setStorage,

setStorageAsync,

getStorage,

getStorageAsync,

removeStorage,

removeStorageAsync,

clearStorage,

clearStorageAsync,

getStorageInfo,

getStorageInfoAsync,

hasStorage,

// 批量操作

setBatchStorage,

getBatchStorage,

removeBatchStorage,

// JSON数据操作

setJsonStorage,

getJsonStorage,

// 用户相关数据

setUserInfo,

getUserInfo,

clearUserInfo,

setToken,

getToken,

clearToken,

// 应用设置

setAppSettings,

getAppSettings,

updateAppSettings,

// 搜索历史

addSearchHistory,

getSearchHistory,

clearSearchHistory,

removeSearchHistoryItem,

// 浏览历史

addBrowseHistory,

getBrowseHistory,

clearBrowseHistory,

// 草稿数据

saveDraft,

getDraft,

removeDraft,

getAllDrafts,

clearAllDrafts,

clearExpiredDrafts,

// 缓存数据

setCache,

getCache,

clearExpiredCache,

// 存储信息

getStorageUsage

}

接下来开发：**小程序全局配置组件 - components/global-config/index.js**

小程序全局配置组件 - components/global-config/index.js

// 全局配置组件

const api = require('../../utils/api.js')

const storage = require('../../utils/storage.js')

const common = require('../../utils/common.js')

Component({

properties: {},

data: {

isLoading: false,

globalConfig: {

// 应用基础配置

appName: '电力渠道销售平台',

version: '1.0.0',

// API配置

apiConfig: {

timeout: 30000,

retryCount: 3,

retryDelay: 1000

},

// 业务配置

businessConfig: {

// 订单状态配置

orderStatus: {

'pending': { name: '待确认', color: '#FFA500', icon: 'clock' },

'negotiating': { name: '商务洽谈', color: '#1890FF', icon: 'chat' },

'contract\_signing': { name: '合同签署', color: '#722ED1', icon: 'edit' },

'service\_starting': { name: '服务开通', color: '#13C2C2', icon: 'play' },

'in\_service': { name: '服务中', color: '#52C41A', icon: 'check' },

'completed': { name: '已完成', color: '#8C8C8C', icon: 'check-circle' },

'cancelled': { name: '已取消', color: '#FF4D4F', icon: 'close' }

},

// 跟进类型配置

followTypes: {

'phone': { name: '电话联系', icon: 'phone', color: '#1890FF' },

'wechat': { name: '微信沟通', icon: 'message', color: '#52C41A' },

'visit': { name: '上门拜访', icon: 'home', color: '#FA8C16' },

'email': { name: '邮件联系', icon: 'mail', color: '#722ED1' },

'meeting': { name: '会议洽谈', icon: 'team', color: '#13C2C2' },

'other': { name: '其他方式', icon: 'more', color: '#8C8C8C' }

},

// 客户状态配置

customerStatus: {

'new': { name: '新客户', color: '#1890FF', icon: 'user-add' },

'contacted': { name: '已联系', color: '#FA8C16', icon: 'phone' },

'interested': { name: '有意向', color: '#722ED1', icon: 'heart' },

'negotiating': { name: '洽谈中', color: '#13C2C2', icon: 'chat' },

'signed': { name: '已签约', color: '#52C41A', icon: 'check' },

'lost': { name: '已流失', color: '#FF4D4F', icon: 'close' }

},

// 产品类型配置

productTypes: {

'basic': { name: '基础套餐', color: '#1890FF', description: '适合小微企业' },

'standard': { name: '标准套餐', color: '#52C41A', description: '适合中小企业' },

'premium': { name: '高级套餐', color: '#722ED1', description: '适合大中型企业' },

'enterprise': { name: '企业套餐', color: '#FA8C16', description: '适合大型企业集团' },

'custom': { name: '定制套餐', color: '#13C2C2', description: '个性化定制方案' }

},

// 认证状态配置

authStatus: {

'unverified': { name: '未认证', color: '#8C8C8C', icon: 'exclamation' },

'pending': { name: '认证中', color: '#FA8C16', icon: 'clock' },

'verified': { name: '已认证', color: '#52C41A', icon: 'check-circle' },

'rejected': { name: '认证失败', color: '#FF4D4F', icon: 'close-circle' }

},

// 行业类型配置

industries: [

{ value: 'manufacturing', label: '制造业' },

{ value: 'retail', label: '零售业' },

{ value: 'service', label: '服务业' },

{ value: 'construction', label: '建筑业' },

{ value: 'agriculture', label: '农业' },

{ value: 'transportation', label: '交通运输业' },

{ value: 'technology', label: '科技业' },

{ value: 'finance', label: '金融业' },

{ value: 'education', label: '教育业' },

{ value: 'healthcare', label: '医疗卫生' },

{ value: 'other', label: '其他' }

],

// 用电容量配置

capacityRanges: [

{ value: '0-100', label: '100kW以下', min: 0, max: 100 },

{ value: '100-500', label: '100-500kW', min: 100, max: 500 },

{ value: '500-1000', label: '500-1000kW', min: 500, max: 1000 },

{ value: '1000-5000', label: '1000-5000kW', min: 1000, max: 5000 },

{ value: '5000+', label: '5000kW以上', min: 5000, max: null }

]

},

// UI配置

uiConfig: {

// 颜色主题

colors: {

primary: '#1890FF',

success: '#52C41A',

warning: '#FA8C16',

error: '#FF4D4F',

info: '#13C2C2',

text: '#262626',

textSecondary: '#8C8C8C',

border: '#E8E8E8',

background: '#F5F5F5'

},

// 字体大小

fontSizes: {

xs: '20rpx',

sm: '24rpx',

md: '28rpx',

lg: '32rpx',

xl: '36rpx',

xxl: '40rpx'

},

// 间距配置

spacing: {

xs: '8rpx',

sm: '16rpx',

md: '24rpx',

lg: '32rpx',

xl: '48rpx'

}

}

}

},

lifetimes: {

attached() {

this.initGlobalConfig()

}

},

methods: {

// 初始化全局配置

async initGlobalConfig() {

try {

this.setData({ isLoading: true })

// 从本地存储获取用户配置

const userConfig = await storage.get('userConfig') || {}

// 合并配置

const mergedConfig = this.mergeConfig(this.data.globalConfig, userConfig)

// 更新全局配置

this.setData({

globalConfig: mergedConfig,

isLoading: false

})

// 触发配置初始化完成事件

this.triggerEvent('configReady', { config: mergedConfig })

} catch (error) {

console.error('初始化全局配置失败:', error)

this.setData({ isLoading: false })

}

},

// 合并配置

mergeConfig(defaultConfig, userConfig) {

return common.deepMerge(defaultConfig, userConfig)

},

// 获取配置项

getConfig(key) {

return common.getNestedValue(this.data.globalConfig, key)

},

// 更新配置项

updateConfig(key, value) {

const config = common.setNestedValue(this.data.globalConfig, key, value)

this.setData({ globalConfig: config })

// 保存到本地存储

storage.set('userConfig', config)

// 触发配置更新事件

this.triggerEvent('configUpdate', { key, value, config })

},

// 获取订单状态信息

getOrderStatus(status) {

return this.getConfig('businessConfig.orderStatus.' + status) || { name: status, color: '#8C8C8C', icon: 'question' }

},

// 获取跟进类型信息

getFollowType(type) {

return this.getConfig('businessConfig.followTypes.' + type) || { name: type, icon: 'more', color: '#8C8C8C' }

},

// 获取客户状态信息

getCustomerStatus(status) {

return this.getConfig('businessConfig.customerStatus.' + status) || { name: status, color: '#8C8C8C', icon: 'user' }

},

// 获取产品类型信息

getProductType(type) {

return this.getConfig('businessConfig.productTypes.' + type) || { name: type, color: '#8C8C8C', description: '' }

},

// 获取认证状态信息

getAuthStatus(status) {

return this.getConfig('businessConfig.authStatus.' + status) || { name: status, color: '#8C8C8C', icon: 'question' }

},

// 获取行业列表

getIndustries() {

return this.getConfig('businessConfig.industries') || []

},

// 获取用电容量范围

getCapacityRanges() {

return this.getConfig('businessConfig.capacityRanges') || []

},

// 格式化金额

formatAmount(amount) {

return common.formatCurrency(amount)

},

// 格式化日期

formatDate(date, format = 'YYYY-MM-DD') {

return common.formatDate(date, format)

},

// 获取主题颜色

getThemeColor(type = 'primary') {

return this.getConfig('uiConfig.colors.' + type) || '#1890FF'

},

// 获取字体大小

getFontSize(size = 'md') {

return this.getConfig('uiConfig.fontSizes.' + size) || '28rpx'

},

// 获取间距

getSpacing(size = 'md') {

return this.getConfig('uiConfig.spacing.' + size) || '24rpx'

}

}

})

现在继续开发下一个模块：**小程序全局配置组件 - components/global-config/index.wxml**

小程序全局配置组件模板 - components/global-config/index.wxml

<!-- 全局配置组件 - 隐藏组件，仅提供配置服务 -->

<view class="global-config" hidden>

<!-- 加载状态 -->

<view wx:if="{{isLoading}}" class="config-loading">

<text>正在初始化配置...</text>

</view>

<!-- 配置信息展示（仅开发环境） -->

<view wx:if="{{globalConfig.debug}}" class="config-debug">

<text class="debug-title">全局配置信息</text>

<view class="debug-item">

<text class="debug-label">应用名称:</text>

<text class="debug-value">{{globalConfig.appName}}</text>

</view>

<view class="debug-item">

<text class="debug-label">版本号:</text>

<text class="debug-value">{{globalConfig.version}}</text>

</view>

<view class="debug-item">

<text class="debug-label">API超时:</text>

<text class="debug-value">{{globalConfig.apiConfig.timeout}}ms</text>

</view>

</view>

</view>

小程序全局配置组件样式 - components/global-config/index.wxss

/\* 全局配置组件样式 \*/

.global-config {

display: none;

}

/\* 配置加载状态 \*/

.config-loading {

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

z-index: 9999;

background: rgba(0, 0, 0, 0.7);

color: #fff;

padding: 32rpx;

border-radius: 16rpx;

font-size: 28rpx;

}

/\* 调试信息样式 \*/

.config-debug {

position: fixed;

top: 0;

right: 0;

width: 500rpx;

background: rgba(0, 0, 0, 0.8);

color: #fff;

padding: 20rpx;

font-size: 24rpx;

z-index: 9998;

max-height: 400rpx;

overflow-y: auto;

}

.debug-title {

display: block;

font-weight: bold;

margin-bottom: 16rpx;

border-bottom: 1rpx solid #333;

padding-bottom: 8rpx;

}

.debug-item {

display: flex;

margin-bottom: 8rpx;

}

.debug-label {

width: 160rpx;

color: #ccc;

}

.debug-value {

flex: 1;

color: #fff;

}

小程序全局配置组件配置 - components/global-config/index.json

{

"component": true,

"usingComponents": {}

}

现在开发下一个模块：**小程序加载组件 - components/loading/index.js**

小程序加载组件 - components/loading/index.js

// 加载组件

Component({

properties: {

// 是否显示加载状态

loading: {

type: Boolean,

value: false

},

// 加载提示文字

text: {

type: String,

value: '加载中...'

},

// 加载类型：spinner, dots, pulse

type: {

type: String,

value: 'spinner'

},

// 尺寸：small, medium, large

size: {

type: String,

value: 'medium'

},

// 颜色

color: {

type: String,

value: '#1890FF'

},

// 是否显示遮罩

mask: {

type: Boolean,

value: true

},

// 遮罩透明度

maskOpacity: {

type: Number,

value: 0.7

},

// 是否全屏显示

fullscreen: {

type: Boolean,

value: false

},

// 是否显示文字

showText: {

type: Boolean,

value: true

},

// 延迟显示时间（毫秒）

delay: {

type: Number,

value: 0

}

},

data: {

visible: false,

delayTimer: null

},

lifetimes: {

attached() {

this.updateVisibility()

},

detached() {

this.clearDelayTimer()

}

},

observers: {

'loading': function(loading) {

this.updateVisibility()

}

},

methods: {

// 更新显示状态

updateVisibility() {

const { loading, delay } = this.properties

this.clearDelayTimer()

if (loading) {

if (delay > 0) {

const timer = setTimeout(() => {

this.setData({ visible: true })

}, delay)

this.setData({ delayTimer: timer })

} else {

this.setData({ visible: true })

}

} else {

this.setData({ visible: false })

}

},

// 清除延迟定时器

clearDelayTimer() {

const { delayTimer } = this.data

if (delayTimer) {

clearTimeout(delayTimer)

this.setData({ delayTimer: null })

}

},

// 点击遮罩

onMaskTap() {

// 阻止事件冒泡

return false

},

// 获取加载器类名

getLoaderClass() {

const { type, size } = this.properties

return `loader loader-${type} loader-${size}`

},

// 获取容器类名

getContainerClass() {

const { fullscreen, mask } = this.properties

let className = 'loading-container'

if (fullscreen) {

className += ' loading-fullscreen'

}

if (mask) {

className += ' loading-mask'

}

return className

},

// 获取遮罩样式

getMaskStyle() {

const { maskOpacity } = this.properties

return `background-color: rgba(0, 0, 0, ${maskOpacity});`

},

// 获取加载器样式

getLoaderStyle() {

const { color } = this.properties

return `color: ${color}; border-color: ${color};`

}

}

})

小程序加载组件模板 - components/loading/index.wxml

<!-- 加载组件 -->

<view wx:if="{{visible}}" class="{{getContainerClass()}}" style="{{getMaskStyle()}}" catchtap="onMaskTap">

<view class="loading-content">

<!-- Spinner 加载器 -->

<view wx:if="{{type === 'spinner'}}" class="{{getLoaderClass()}}" style="{{getLoaderStyle()}}">

<view class="spinner-border"></view>

</view>

<!-- Dots 加载器 -->

<view wx:elif="{{type === 'dots'}}" class="{{getLoaderClass()}}">

<view class="dot" style="background-color: {{color}};"></view>

<view class="dot" style="background-color: {{color}};"></view>

<view class="dot" style="background-color: {{color}};"></view>

</view>

<!-- Pulse 加载器 -->

<view wx:elif="{{type === 'pulse'}}" class="{{getLoaderClass()}}">

<view class="pulse-ring" style="border-color: {{color}};"></view>

<view class="pulse-ring" style="border-color: {{color}};"></view>

<view class="pulse-ring" style="border-color: {{color}};"></view>

</view>

<!-- 加载文字 -->

<text wx:if="{{showText && text}}" class="loading-text" style="color: {{color}};">{{text}}</text>

</view>

</view>

小程序加载组件样式 - components/loading/index.wxss

/\* 加载组件样式 \*/

.loading-container {

display: flex;

align-items: center;

justify-content: center;

min-height: 200rpx;

position: relative;

}

.loading-fullscreen {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

z-index: 9999;

min-height: 100vh;

}

.loading-mask {

background-color: rgba(0, 0, 0, 0.7);

}

.loading-content {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding: 32rpx;

background: rgba(255, 255, 255, 0.95);

border-radius: 16rpx;

min-width: 200rpx;

}

.loading-text {

font-size: 28rpx;

margin-top: 24rpx;

text-align: center;

}

/\* Spinner 加载器 \*/

.loader-spinner {

position: relative;

}

.spinner-border {

width: 60rpx;

height: 60rpx;

border: 4rpx solid transparent;

border-top-color: currentColor;

border-radius: 50%;

animation: spin 1s linear infinite;

}

.loader-spinner.loader-small .spinner-border {

width: 40rpx;

height: 40rpx;

border-width: 3rpx;

}

.loader-spinner.loader-large .spinner-border {

width: 80rpx;

height: 80rpx;

border-width: 6rpx;

}

/\* Dots 加载器 \*/

.loader-dots {

display: flex;

align-items: center;

gap: 8rpx;

}

.loader-dots .dot {

width: 12rpx;

height: 12rpx;

border-radius: 50%;

animation: dot-bounce 1.4s ease-in-out infinite both;

}

.loader-dots .dot:nth-child(1) {

animation-delay: -0.32s;

}

.loader-dots .dot:nth-child(2) {

animation-delay: -0.16s;

}

.loader-dots.loader-small .dot {

width: 8rpx;

height: 8rpx;

}

.loader-dots.loader-large .dot {

width: 16rpx;

height: 16rpx;

}

/\* Pulse 加载器 \*/

.loader-pulse {

position: relative;

width: 60rpx;

height: 60rpx;

}

.pulse-ring {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

border: 2rpx solid currentColor;

border-radius: 50%;

opacity: 1;

animation: pulse-scale 1.5s ease-out infinite;

}

.pulse-ring:nth-child(2) {

animation-delay: 0.5s;

}

.pulse-ring:nth-child(3) {

animation-delay: 1s;

}

.loader-pulse.loader-small {

width: 40rpx;

height: 40rpx;

}

.loader-pulse.loader-large {

width: 80rpx;

height: 80rpx;

}

/\* 动画定义 \*/

@keyframes spin {

0% {

transform: rotate(0deg);

}

100% {

transform: rotate(360deg);

}

}

@keyframes dot-bounce {

0%, 80%, 100% {

transform: scale(0);

opacity: 0.5;

}

40% {

transform: scale(1);

opacity: 1;

}

}

@keyframes pulse-scale {

0% {

transform: scale(0);

opacity: 1;

}

100% {

transform: scale(1);

opacity: 0;

}

}

小程序加载组件配置 - components/loading/index.json

{

"component": true,

"usingComponents": {}

}

现在开发下一个模块：**小程序空状态组件 - components/empty/index.js**

小程序空状态组件 - components/empty/index.js

// 空状态组件

Component({

properties: {

// 是否显示空状态

show: {

type: Boolean,

value: false

},

// 空状态类型：default, nodata, network, search, cart, order, customer

type: {

type: String,

value: 'default'

},

// 图标名称（优先级高于type）

icon: {

type: String,

value: ''

},

// 图标大小

iconSize: {

type: Number,

value: 120

},

// 图标颜色

iconColor: {

type: String,

value: '#d9d9d9'

},

// 主标题

title: {

type: String,

value: ''

},

// 描述文字

description: {

type: String,

value: ''

},

// 按钮文字

buttonText: {

type: String,

value: ''

},

// 按钮类型：primary, default

buttonType: {

type: String,

value: 'primary'

},

// 是否显示按钮

showButton: {

type: Boolean,

value: false

},

// 自定义样式

customStyle: {

type: String,

value: ''

}

},

data: {

// 预设的空状态配置

presetConfigs: {

default: {

icon: 'empty-box',

title: '暂无数据',

description: '当前没有相关数据'

},

nodata: {

icon: 'file-text',

title: '暂无数据',

description: '暂时没有找到相关内容'

},

network: {

icon: 'wifi-off',

title: '网络异常',

description: '请检查网络连接后重试'

},

search: {

icon: 'search',

title: '暂无搜索结果',

description: '试试其他关键词吧'

},

cart: {

icon: 'shopping-cart',

title: '购物车为空',

description: '快去选购心仪的产品吧'

},

order: {

icon: 'file-text',

title: '暂无订单',

description: '您还没有任何订单记录'

},

customer: {

icon: 'users',

title: '暂无客户',

description: '还没有分配的客户'

},

follow: {

icon: 'message-circle',

title: '暂无跟进记录',

description: '开始添加客户跟进记录吧'

},

contract: {

icon: 'file',

title: '暂无合同',

description: '您还没有签署任何合同'

},

performance: {

icon: 'trending-up',

title: '暂无业绩数据',

description: '继续努力，创造更好的业绩'

}

}

},

lifetimes: {

attached() {

this.updateConfig()

}

},

observers: {

'type, title, description, icon': function() {

this.updateConfig()

}

},

methods: {

// 更新配置

updateConfig() {

const { type, title, description, icon } = this.properties

const { presetConfigs } = this.data

const presetConfig = presetConfigs[type] || presetConfigs.default

// 使用属性值或预设值

const finalConfig = {

icon: icon || presetConfig.icon,

title: title || presetConfig.title,

description: description || presetConfig.description

}

this.setData({ currentConfig: finalConfig })

},

// 按钮点击事件

onButtonTap() {

this.triggerEvent('buttonTap', {

type: this.properties.type

})

},

// 获取图标类名

getIconClass() {

const { currentConfig } = this.data

const { iconSize } = this.properties

if (!currentConfig || !currentConfig.icon) {

return ''

}

return `icon icon-${currentConfig.icon}`

},

// 获取图标样式

getIconStyle() {

const { iconSize, iconColor } = this.properties

return `font-size: ${iconSize}rpx; color: ${iconColor};`

},

// 获取容器样式

getContainerStyle() {

const { customStyle } = this.properties

return customStyle

},

// 获取按钮类名

getButtonClass() {

const { buttonType } = this.properties

return `empty-button empty-button-${buttonType}`

}

}

})

继续完成小程序组件开发的其余部分，然后总结已完成的模块：

**小程序空状态组件模板**

小程序空状态组件模板 - components/empty/index.wxml

<!-- 小程序空状态组件模板 -->

<view class="empty-container">

<view class="empty-content">

<!-- 空状态图标 -->

<view class="empty-icon">

<image wx:if="{{type === 'noData'}}" src="/images/empty/no-data.png" class="empty-image" />

<image wx:elif="{{type === 'network'}}" src="/images/empty/network-error.png" class="empty-image" />

<image wx:elif="{{type === 'search'}}" src="/images/empty/no-search.png" class="empty-image" />

<image wx:elif="{{type === 'order'}}" src="/images/empty/no-order.png" class="empty-image" />

<image wx:elif="{{type === 'customer'}}" src="/images/empty/no-customer.png" class="empty-image" />

<image wx:else src="/images/empty/default.png" class="empty-image" />

</view>

<!-- 空状态文本 -->

<view class="empty-text">

<text class="empty-title">{{title || defaultTitle}}</text>

<text wx:if="{{description}}" class="empty-desc">{{description}}</text>

</view>

<!-- 操作按钮 -->

<view wx:if="{{showButton}}" class="empty-actions">

<button

wx:if="{{buttonText}}"

class="empty-button {{buttonType}}"

bindtap="onButtonTap"

>

{{buttonText}}

</button>

<!-- 自定义操作按钮 -->

<slot name="actions"></slot>

</view>

</view>

</view>

小程序空状态组件样式 - components/empty/index.wxss

/\* 小程序空状态组件样式 \*/

.empty-container {

width: 100%;

height: 100%;

display: flex;

align-items: center;

justify-content: center;

min-height: 400rpx;

padding: 40rpx;

box-sizing: border-box;

}

.empty-content {

display: flex;

flex-direction: column;

align-items: center;

text-align: center;

}

/\* 空状态图标 \*/

.empty-icon {

margin-bottom: 32rpx;

}

.empty-image {

width: 200rpx;

height: 200rpx;

opacity: 0.6;

}

/\* 空状态文本 \*/

.empty-text {

margin-bottom: 40rpx;

}

.empty-title {

display: block;

font-size: 32rpx;

color: #666;

margin-bottom: 16rpx;

font-weight: 500;

}

.empty-desc {

display: block;

font-size: 28rpx;

color: #999;

line-height: 1.6;

max-width: 500rpx;

}

/\* 操作按钮 \*/

.empty-actions {

display: flex;

flex-direction: column;

align-items: center;

gap: 24rpx;

}

.empty-button {

padding: 20rpx 60rpx;

border-radius: 48rpx;

font-size: 28rpx;

border: none;

min-width: 200rpx;

}

.empty-button.primary {

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

color: white;

}

.empty-button.default {

background: #f5f5f5;

color: #666;

}

.empty-button.ghost {

background: transparent;

color: var(--primary-color);

border: 2rpx solid var(--primary-color);

}

.empty-button::after {

border: none;

}

/\* 响应式适配 \*/

@media (max-width: 750rpx) {

.empty-image {

width: 160rpx;

height: 160rpx;

}

.empty-title {

font-size: 30rpx;

}

.empty-desc {

font-size: 26rpx;

}

}

小程序空状态组件配置 - components/empty/index.json

{

"component": true,

"usingComponents": {}

}

**小程序模态框组件**

小程序模态框组件 - components/modal/index.js

// 小程序模态框组件

Component({

properties: {

// 是否显示模态框

visible: {

type: Boolean,

value: false

},

// 标题

title: {

type: String,

value: ''

},

// 内容

content: {

type: String,

value: ''

},

// 是否显示确认按钮

showConfirm: {

type: Boolean,

value: true

},

// 是否显示取消按钮

showCancel: {

type: Boolean,

value: true

},

// 确认按钮文本

confirmText: {

type: String,

value: '确定'

},

// 取消按钮文本

cancelText: {

type: String,

value: '取消'

},

// 确认按钮颜色

confirmColor: {

type: String,

value: '#007aff'

},

// 取消按钮颜色

cancelColor: {

type: String,

value: '#000'

},

// 是否可点击遮罩关闭

maskClosable: {

type: Boolean,

value: true

},

// 模态框类型

type: {

type: String,

value: 'default' // default, success, warning, error

},

// 自定义样式类

customClass: {

type: String,

value: ''

}

},

data: {

// 动画状态

animationData: {}

},

lifetimes: {

attached() {

this.animation = wx.createAnimation({

duration: 300,

timingFunction: 'ease'

});

}

},

observers: {

'visible'(visible) {

if (visible) {

this.showModal();

} else {

this.hideModal();

}

}

},

methods: {

// 显示模态框动画

showModal() {

this.animation.opacity(1).scale(1).step();

this.setData({

animationData: this.animation.export()

});

},

// 隐藏模态框动画

hideModal() {

this.animation.opacity(0).scale(0.9).step();

this.setData({

animationData: this.animation.export()

});

},

// 点击遮罩

onMaskTap() {

if (this.data.maskClosable) {

this.onCancel();

}

},

// 点击确认按钮

onConfirm() {

this.triggerEvent('confirm', {

type: 'confirm'

});

},

// 点击取消按钮

onCancel() {

this.triggerEvent('cancel', {

type: 'cancel'

});

},

// 阻止冒泡

preventTap() {

// 阻止点击事件冒泡到遮罩层

}

}

});

小程序模态框组件模板 - components/modal/index.wxml

<!-- 小程序模态框组件模板 -->

<view wx:if="{{visible}}" class="modal-mask" bindtap="onMaskTap">

<view

class="modal-container {{customClass}}"

animation="{{animationData}}"

bindtap="preventTap"

>

<!-- 模态框头部 -->

<view wx:if="{{title}}" class="modal-header">

<!-- 类型图标 -->

<view wx:if="{{type !== 'default'}}" class="modal-icon modal-icon-{{type}}">

<text wx:if="{{type === 'success'}}" class="icon">✓</text>

<text wx:elif="{{type === 'warning'}}" class="icon">!</text>

<text wx:elif="{{type === 'error'}}" class="icon">✕</text>

</view>

<text class="modal-title">{{title}}</text>

</view>

<!-- 模态框内容 -->

<view class="modal-body">

<text wx:if="{{content}}" class="modal-content">{{content}}</text>

<slot></slot>

</view>

<!-- 模态框底部按钮 -->

<view wx:if="{{showConfirm || showCancel}}" class="modal-footer">

<button

wx:if="{{showCancel}}"

class="modal-button modal-button-cancel"

style="color: {{cancelColor}}"

bindtap="onCancel"

>

{{cancelText}}

</button>

<button

wx:if="{{showConfirm}}"

class="modal-button modal-button-confirm"

style="color: {{confirmColor}}"

bindtap="onConfirm"

>

{{confirmText}}

</button>

</view>

</view>

</view>

小程序模态框组件样式 - components/modal/index.wxss

/\* 小程序模态框组件样式 \*/

.modal-mask {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.5);

display: flex;

align-items: center;

justify-content: center;

z-index: 999;

padding: 40rpx;

box-sizing: border-box;

}

.modal-container {

background: white;

border-radius: 24rpx;

max-width: 600rpx;

width: 100%;

max-height: 80vh;

overflow: hidden;

box-shadow: 0 8rpx 32rpx rgba(0, 0, 0, 0.1);

}

/\* 模态框头部 \*/

.modal-header {

padding: 40rpx 40rpx 20rpx;

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

text-align: center;

}

.modal-icon {

width: 80rpx;

height: 80rpx;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

margin-bottom: 20rpx;

}

.modal-icon .icon {

font-size: 40rpx;

font-weight: bold;

color: white;

}

.modal-icon-success {

background: #52c41a;

}

.modal-icon-warning {

background: #faad14;

}

.modal-icon-error {

background: #ff4d4f;

}

.modal-title {

font-size: 36rpx;

font-weight: 600;

color: #333;

line-height: 1.4;

}

/\* 模态框内容 \*/

.modal-body {

padding: 20rpx 40rpx 40rpx;

max-height: 60vh;

overflow-y: auto;

}

.modal-content {

font-size: 32rpx;

color: #666;

line-height: 1.6;

text-align: center;

}

/\* 模态框底部 \*/

.modal-footer {

display: flex;

border-top: 1rpx solid #f0f0f0;

}

.modal-button {

flex: 1;

height: 100rpx;

line-height: 100rpx;

text-align: center;

font-size: 32rpx;

background: transparent;

border: none;

border-radius: 0;

}

.modal-button::after {

border: none;

}

.modal-button-cancel {

border-right: 1rpx solid #f0f0f0;

color: #999;

}

.modal-button-confirm {

color: var(--primary-color);

font-weight: 500;

}

/\* 单按钮样式 \*/

.modal-footer .modal-button:only-child {

border-right: none;

}

/\* 长内容适配 \*/

.modal-body::-webkit-scrollbar {

width: 0;

height: 0;

}

/\* 响应式适配 \*/

@media (max-width: 750rpx) {

.modal-container {

max-width: 90%;

}

.modal-header {

padding: 32rpx 32rpx 16rpx;

}

.modal-body {

padding: 16rpx 32rpx 32rpx;

}

.modal-title {

font-size: 32rpx;

}

.modal-content {

font-size: 28rpx;

}

}

小程序模态框组件配置 - components/modal/index.json

{

"component": true,

"usingComponents": {}

}

**小程序标签页组件**

小程序标签页组件 - components/tabs/index.js

// 小程序标签页组件

Component({

properties: {

// 标签页数据

tabs: {

type: Array,

value: []

},

// 当前选中的标签索引

current: {

type: Number,

value: 0

},

// 标签页类型

type: {

type: String,

value: 'line' // line, card, button

},

// 是否可滚动

scrollable: {

type: Boolean,

value: false

},

// 标签页颜色

color: {

type: String,

value: '#333'

},

// 激活颜色

activeColor: {

type: String,

value: '#007aff'

},

// 背景颜色

backgroundColor: {

type: String,

value: '#fff'

},

// 是否显示徽章

showBadge: {

type: Boolean,

value: false

},

// 自定义样式类

customClass: {

type: String,

value: ''

}

},

data: {

// 滑块样式

lineStyle: '',

// 滚动视图位置

scrollLeft: 0

},

lifetimes: {

attached() {

this.updateLinePosition();

}

},

observers: {

'current, tabs'() {

this.updateLinePosition();

this.updateScrollPosition();

}

},

methods: {

// 更新滑块位置

updateLinePosition() {

if (this.data.type !== 'line') return;

const { current, tabs } = this.data;

if (!tabs.length) return;

// 使用节点查询API获取标签宽度和位置

const query = this.createSelectorQuery();

query.selectAll('.tab-item').boundingClientRect((rects) => {

if (rects && rects[current]) {

const rect = rects[current];

const lineStyle = `

width: ${rect.width \* 0.6}px;

transform: translateX(${rect.left - rects[0].left + rect.width \* 0.2}px);

`;

this.setData({ lineStyle });

}

}).exec();

},

// 更新滚动位置

updateScrollPosition() {

if (!this.data.scrollable) return;

const { current, tabs } = this.data;

if (!tabs.length) return;

// 计算滚动位置，让当前标签居中

const query = this.createSelectorQuery();

query.select('.tabs-scroll').boundingClientRect((scrollRect) => {

query.selectAll('.tab-item').boundingClientRect((rects) => {

if (rects && rects[current] && scrollRect) {

const currentRect = rects[current];

const scrollLeft = currentRect.left - scrollRect.left - scrollRect.width / 2 + currentRect.width / 2;

this.setData({ scrollLeft: Math.max(0, scrollLeft) });

}

}).exec();

}).exec();

},

// 点击标签

onTabTap(e) {

const { index } = e.currentTarget.dataset;

const { current } = this.data;

if (index !== current) {

this.triggerEvent('change', {

index,

tab: this.data.tabs[index]

});

}

},

// 获取标签样式

getTabStyle(index) {

const { current, color, activeColor } = this.data;

const isActive = index === current;

return `color: ${isActive ? activeColor : color};`;

},

// 获取徽章数量

getBadgeCount(tab) {

return tab.badge || tab.count || 0;

}

}

});

小程序标签页组件模板 - components/tabs/index.wxml

<!-- 小程序标签页组件模板 -->

<view class="tabs-container {{customClass}} tabs-{{type}}" style="background-color: {{backgroundColor}}">

<!-- 可滚动标签页 -->

<scroll-view

wx:if="{{scrollable}}"

class="tabs-scroll"

scroll-x

scroll-left="{{scrollLeft}}"

scroll-with-animation

show-scrollbar="{{false}}"

>

<view class="tabs-wrapper">

<view

wx:for="{{tabs}}"

wx:key="index"

class="tab-item {{current === index ? 'tab-active' : ''}}"

data-index="{{index}}"

style="{{getTabStyle(index)}}"

bindtap="onTabTap"

>

<!-- 标签图标 -->

<image wx:if="{{item.icon}}" class="tab-icon" src="{{item.icon}}" />

<!-- 标签文本 -->

<text class="tab-text">{{item.title || item.name}}</text>

<!-- 徽章 -->

<view

wx:if="{{showBadge && getBadgeCount(item) > 0}}"

class="tab-badge"

>

<text class="badge-text">{{getBadgeCount(item)}}</text>

</view>

</view>

<!-- 下划线 -->

<view wx:if="{{type === 'line'}}" class="tab-line" style="{{lineStyle}}"></view>

</view>

</scroll-view>

<!-- 固定标签页 -->

<view wx:else class="tabs-wrapper">

<view

wx:for="{{tabs}}"

wx:key="index"

class="tab-item {{current === index ? 'tab-active' : ''}}"

data-index="{{index}}"

style="{{getTabStyle(index)}}"

bindtap="onTabTap"

>

<!-- 标签图标 -->

<image wx:if="{{item.icon}}" class="tab-icon" src="{{item.icon}}" />

<!-- 标签文本 -->

<text class="tab-text">{{item.title || item.name}}</text>

<!-- 徽章 -->

<view

wx:if="{{showBadge && getBadgeCount(item) > 0}}"

class="tab-badge"

>

<text class="badge-text">{{getBadgeCount(item)}}</text>

</view>

</view>

<!-- 下划线 -->

<view wx:if="{{type === 'line'}}" class="tab-line" style="{{lineStyle}}"></view>

</view>

</view>

小程序标签页组件样式 - components/tabs/index.wxss

/\* 小程序标签页组件样式 \*/

.tabs-container {

position: relative;

background: white;

border-bottom: 1rpx solid #f0f0f0;

}

.tabs-scroll {

white-space: nowrap;

}

.tabs-wrapper {

display: flex;

position: relative;

min-height: 88rpx;

}

/\* 标签项 \*/

.tab-item {

position: relative;

display: flex;

align-items: center;

justify-content: center;

padding: 20rpx 32rpx;

min-width: 120rpx;

font-size: 28rpx;

color: #666;

transition: all 0.3s;

flex-shrink: 0;

}

.tab-item.tab-active {

font-weight: 500;

}

/\* 可滚动时的标签项 \*/

.tabs-scroll .tab-item {

white-space: nowrap;

}

/\* 标签图标 \*/

.tab-icon {

width: 32rpx;

height: 32rpx;

margin-right: 8rpx;

}

.tab-text {

flex-shrink: 0;

}

/\* 徽章 \*/

.tab-badge {

position: absolute;

top: 12rpx;

right: 20rpx;

background: #ff4d4f;

color: white;

border-radius: 20rpx;

min-width: 32rpx;

height: 32rpx;

display: flex;

align-items: center;

justify-content: center;

font-size: 20rpx;

transform: scale(0.8);

}

.badge-text {

padding: 0 8rpx;

line-height: 1;

}

/\* 线条类型 \*/

.tabs-line .tab-line {

position: absolute;

bottom: 0;

height: 4rpx;

background: var(--primary-color);

border-radius: 2rpx;

transition: all 0.3s;

}

/\* 卡片类型 \*/

.tabs-card .tabs-wrapper {

background: #f5f5f5;

border-radius: 12rpx;

padding: 8rpx;

margin: 16rpx;

}

.tabs-card .tab-item {

border-radius: 8rpx;

margin: 0 4rpx;

background: transparent;

}

.tabs-card .tab-item.tab-active {

background: white;

box-shadow: 0 2rpx 8rpx rgba(0, 0, 0, 0.1);

}

/\* 按钮类型 \*/

.tabs-button .tabs-wrapper {

padding: 16rpx;

gap: 16rpx;

}

.tabs-button .tab-item {

border: 2rpx solid #e8e8e8;

border-radius: 48rpx;

background: white;

flex: 1;

}

.tabs-button .tab-item.tab-active {

border-color: var(--primary-color);

background: var(--primary-color);

color: white;

}

/\* 等宽分布 \*/

.tabs-wrapper:not(.tabs-scroll .tabs-wrapper) .tab-item {

flex: 1;

}

/\* 响应式适配 \*/

@media (max-width: 750rpx) {

.tab-item {

padding: 16rpx 24rpx;

font-size: 26rpx;

min-width: 100rpx;

}

.tab-icon {

width: 28rpx;

height: 28rpx;

}

}

小程序标签页组件配置 - components/tabs/index.json

{

"component": true,

"usingComponents": {}

}

**小程序图片预览组件**

小程序图片预览组件 - components/image-preview/index.js

// components/image-preview/index.js

Component({

/\*\*

\* 组件的属性列表

\*/

properties: {

// 图片列表

images: {

type: Array,

value: []

},

// 当前显示的图片索引

current: {

type: Number,

value: 0

},

// 是否显示预览

show: {

type: Boolean,

value: false

},

// 是否显示删除按钮

showDelete: {

type: Boolean,

value: false

},

// 是否显示下载按钮

showDownload: {

type: Boolean,

value: true

},

// 是否显示分享按钮

showShare: {

type: Boolean,

value: false

}

},

/\*\*

\* 组件的初始数据

\*/

data: {

currentIndex: 0,

swiperCurrent: 0,

imageLoaded: {},

imageError: {}

},

/\*\*

\* 组件的方法列表

\*/

methods: {

/\*\*

\* 监听属性变化

\*/

observers: {

'show, current'(show, current) {

if (show) {

this.setData({

currentIndex: current,

swiperCurrent: current

});

}

}

},

/\*\*

\* 关闭预览

\*/

onClose() {

this.triggerEvent('close');

},

/\*\*

\* 阻止事件冒泡

\*/

onStopPropagation() {

// 阻止点击事件冒泡到背景层

},

/\*\*

\* Swiper切换事件

\*/

onSwiperChange(e) {

const current = e.detail.current;

this.setData({

currentIndex: current,

swiperCurrent: current

});

// 触发切换事件

this.triggerEvent('change', {

current: current,

image: this.data.images[current]

});

},

/\*\*

\* 图片加载成功

\*/

onImageLoad(e) {

const index = e.currentTarget.dataset.index;

this.setData({

[`imageLoaded.${index}`]: true

});

},

/\*\*

\* 图片加载失败

\*/

onImageError(e) {

const index = e.currentTarget.dataset.index;

this.setData({

[`imageError.${index}`]: true

});

// 触发加载失败事件

this.triggerEvent('error', {

index: index,

image: this.data.images[index]

});

},

/\*\*

\* 删除图片

\*/

onDelete() {

const currentIndex = this.data.currentIndex;

const currentImage = this.data.images[currentIndex];

wx.showModal({

title: '确认删除',

content: '确定要删除这张图片吗？',

success: (res) => {

if (res.confirm) {

this.triggerEvent('delete', {

index: currentIndex,

image: currentImage

});

}

}

});

},

/\*\*

\* 下载图片

\*/

onDownload() {

const currentImage = this.data.images[this.data.currentIndex];

if (!currentImage) return;

wx.showLoading({

title: '保存中...'

});

// 如果是网络图片，先下载

if (currentImage.startsWith('http')) {

wx.downloadFile({

url: currentImage,

success: (res) => {

if (res.statusCode === 200) {

this.saveImageToAlbum(res.tempFilePath);

} else {

this.showDownloadError();

}

},

fail: () => {

this.showDownloadError();

}

});

} else {

// 本地图片直接保存

this.saveImageToAlbum(currentImage);

}

},

/\*\*

\* 保存图片到相册

\*/

saveImageToAlbum(filePath) {

wx.saveImageToPhotosAlbum({

filePath: filePath,

success: () => {

wx.hideLoading();

wx.showToast({

title: '保存成功',

icon: 'success'

});

this.triggerEvent('download', {

index: this.data.currentIndex,

image: this.data.images[this.data.currentIndex],

filePath: filePath

});

},

fail: (err) => {

wx.hideLoading();

if (err.errMsg.includes('auth deny')) {

wx.showModal({

title: '提示',

content: '需要您授权保存图片到相册',

confirmText: '去设置',

success: (res) => {

if (res.confirm) {

wx.openSetting();

}

}

});

} else {

this.showDownloadError();

}

}

});

},

/\*\*

\* 显示下载错误提示

\*/

showDownloadError() {

wx.hideLoading();

wx.showToast({

title: '保存失败',

icon: 'none'

});

},

/\*\*

\* 分享图片

\*/

onShare() {

const currentImage = this.data.images[this.data.currentIndex];

this.triggerEvent('share', {

index: this.data.currentIndex,

image: currentImage

});

},

/\*\*

\* 双击放大/缩小

\*/

onDoubleTap() {

// 可以在这里实现双击放大缩小功能

this.triggerEvent('doubletap', {

index: this.data.currentIndex,

image: this.data.images[this.data.currentIndex]

});

}

},

/\*\*

\* 组件生命周期

\*/

lifetimes: {

attached() {

// 组件实例进入页面节点树时执行

},

detached() {

// 组件实例被从页面节点树移除时执行

}

},

/\*\*

\* 组件所在页面的生命周期

\*/

pageLifetimes: {

show() {

// 组件所在的页面被展示时执行

},

hide() {

// 组件所在的页面被隐藏时执行

}

}

});

小程序图片预览组件模板 - components/image-preview/index.wxml

<!-- components/image-preview/index.wxml -->

<view class="image-preview" wx:if="{{show}}" bindtap="onClose">

<!-- 背景遮罩 -->

<view class="image-preview\_\_mask"></view>

<!-- 图片容器 -->

<view class="image-preview\_\_container" bindtap="onStopPropagation">

<!-- 图片轮播 -->

<swiper

class="image-preview\_\_swiper"

current="{{swiperCurrent}}"

bindchange="onSwiperChange"

indicator-dots="{{images.length > 1}}"

indicator-color="rgba(255, 255, 255, 0.3)"

indicator-active-color="#ffffff"

circular="{{images.length > 1}}"

>

<swiper-item

wx:for="{{images}}"

wx:key="index"

class="image-preview\_\_item"

>

<!-- 加载状态 -->

<view

class="image-preview\_\_loading"

wx:if="{{!imageLoaded[index] && !imageError[index]}}"

>

<text class="loading-text">加载中...</text>

</view>

<!-- 加载失败状态 -->

<view

class="image-preview\_\_error"

wx:if="{{imageError[index]}}"

>

<text class="error-icon">📷</text>

<text class="error-text">图片加载失败</text>

</view>

<!-- 图片 -->

<image

class="image-preview\_\_image"

src="{{item}}"

mode="aspectFit"

data-index="{{index}}"

bindload="onImageLoad"

binderror="onImageError"

bindtap="onDoubleTap"

wx:if="{{!imageError[index]}}"

/>

</swiper-item>

</swiper>

<!-- 图片索引 -->

<view class="image-preview\_\_index" wx:if="{{images.length > 1}}">

{{currentIndex + 1}} / {{images.length}}

</view>

<!-- 操作按钮 -->

<view class="image-preview\_\_actions">

<!-- 删除按钮 -->

<view

class="action-btn delete-btn"

wx:if="{{showDelete}}"

bindtap="onDelete"

>

<text class="action-icon">🗑️</text>

</view>

<!-- 下载按钮 -->

<view

class="action-btn download-btn"

wx:if="{{showDownload}}"

bindtap="onDownload"

>

<text class="action-icon">📥</text>

</view>

<!-- 分享按钮 -->

<view

class="action-btn share-btn"

wx:if="{{showShare}}"

bindtap="onShare"

>

<text class="action-icon">📤</text>

</view>

</view>

<!-- 关闭按钮 -->

<view class="image-preview\_\_close" bindtap="onClose">

<text class="close-icon">✕</text>

</view>

</view>

</view>

小程序图片预览组件样式 - components/image-preview/index.wxss

/\* components/image-preview/index.wxss \*/

.image-preview {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

z-index: 9999;

display: flex;

align-items: center;

justify-content: center;

}

.image-preview\_\_mask {

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0, 0, 0, 0.8);

backdrop-filter: blur(10px);

}

.image-preview\_\_container {

position: relative;

width: 100%;

height: 100%;

display: flex;

align-items: center;

justify-content: center;

}

.image-preview\_\_swiper {

width: 100%;

height: 100%;

}

.image-preview\_\_item {

display: flex;

align-items: center;

justify-content: center;

padding: 40rpx;

}

.image-preview\_\_image {

max-width: 100%;

max-height: 100%;

border-radius: 8rpx;

}

/\* 加载状态 \*/

.image-preview\_\_loading {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

color: #ffffff;

}

.loading-text {

font-size: 28rpx;

margin-top: 20rpx;

}

/\* 错误状态 \*/

.image-preview\_\_error {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

color: #ffffff;

}

.error-icon {

font-size: 80rpx;

margin-bottom: 20rpx;

opacity: 0.6;

}

.error-text {

font-size: 28rpx;

opacity: 0.8;

}

/\* 图片索引 \*/

.image-preview\_\_index {

position: absolute;

bottom: 120rpx;

left: 50%;

transform: translateX(-50%);

background-color: rgba(0, 0, 0, 0.5);

backdrop-filter: blur(10px);

color: #ffffff;

padding: 16rpx 32rpx;

border-radius: 40rpx;

font-size: 28rpx;

z-index: 10;

}

/\* 操作按钮 \*/

.image-preview\_\_actions {

position: absolute;

bottom: 40rpx;

left: 50%;

transform: translateX(-50%);

display: flex;

gap: 20rpx;

z-index: 10;

}

.action-btn {

width: 80rpx;

height: 80rpx;

background-color: rgba(0, 0, 0, 0.5);

backdrop-filter: blur(10px);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

transition: all 0.3s ease;

}

.action-btn:active {

transform: scale(0.9);

background-color: rgba(0, 0, 0, 0.7);

}

.action-icon {

font-size: 32rpx;

}

.delete-btn .action-icon {

filter: brightness(0) invert(1);

}

.download-btn .action-icon {

filter: brightness(0) invert(1);

}

.share-btn .action-icon {

filter: brightness(0) invert(1);

}

/\* 关闭按钮 \*/

.image-preview\_\_close {

position: absolute;

top: 40rpx;

right: 40rpx;

width: 60rpx;

height: 60rpx;

background-color: rgba(0, 0, 0, 0.5);

backdrop-filter: blur(10px);

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

z-index: 10;

transition: all 0.3s ease;

}

.image-preview\_\_close:active {

transform: scale(0.9);

background-color: rgba(0, 0, 0, 0.7);

}

.close-icon {

color: #ffffff;

font-size: 32rpx;

font-weight: bold;

}

/\* 适配安全区域 \*/

.image-preview\_\_close {

top: calc(40rpx + env(safe-area-inset-top));

}

.image-preview\_\_actions {

bottom: calc(40rpx + env(safe-area-inset-bottom));

}

.image-preview\_\_index {

bottom: calc(120rpx + env(safe-area-inset-bottom));

}

/\* 横屏适配 \*/

@media (orientation: landscape) {

.image-preview\_\_item {

padding: 20rpx 80rpx;

}

.image-preview\_\_close {

top: 20rpx;

right: 20rpx;

}

.image-preview\_\_actions {

bottom: 20rpx;

}

.image-preview\_\_index {

bottom: 80rpx;

}

}

/\* 动画效果 \*/

.image-preview {

animation: fadeIn 0.3s ease-out;

}

@keyframes fadeIn {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

.image-preview\_\_container {

animation: scaleIn 0.3s ease-out;

}

@keyframes scaleIn {

from {

transform: scale(0.8);

opacity: 0;

}

to {

transform: scale(1);

opacity: 1;

}

}

小程序图片预览组件配置 - components/image-preview/index.json

{

"component": true,

"usingComponents": {}

}

现在继续完成剩余的小程序公共组件。接下来开发小程序搜索组件：

小程序搜索组件 - components/search/index.js

// components/search/index.js

Component({

/\*\*

\* 组件的属性列表

\*/

properties: {

// 占位符文本

placeholder: {

type: String,

value: '请输入搜索内容'

},

// 搜索值

value: {

type: String,

value: ''

},

// 是否显示搜索按钮

showButton: {

type: Boolean,

value: true

},

// 搜索按钮文本

buttonText: {

type: String,

value: '搜索'

},

// 是否显示清除按钮

showClear: {

type: Boolean,

value: true

},

// 是否显示历史记录

showHistory: {

type: Boolean,

value: true

},

// 历史记录最大数量

maxHistory: {

type: Number,

value: 10

},

// 搜索建议列表

suggestions: {

type: Array,

value: []

},

// 是否显示搜索建议

showSuggestions: {

type: Boolean,

value: true

},

// 是否自动获取焦点

autoFocus: {

type: Boolean,

value: false

},

// 输入框样式

inputStyle: {

type: String,

value: ''

},

// 是否禁用

disabled: {

type: Boolean,

value: false

}

},

/\*\*

\* 组件的初始数据

\*/

data: {

inputValue: '',

isFocused: false,

showPanel: false,

searchHistory: []

},

/\*\*

\* 组件生命周期

\*/

lifetimes: {

attached() {

this.setData({

inputValue: this.data.value

});

this.loadSearchHistory();

}

},

/\*\*

\* 监听属性变化

\*/

observers: {

'value'(newVal) {

this.setData({

inputValue: newVal

});

}

},

/\*\*

\* 组件的方法列表

\*/

methods: {

/\*\*

\* 输入框输入事件

\*/

onInput(e) {

const value = e.detail.value;

this.setData({

inputValue: value

});

// 触发输入事件

this.triggerEvent('input', {

value: value

});

// 触发搜索建议

if (value.trim() && this.data.showSuggestions) {

this.triggerEvent('suggest', {

value: value

});

}

},

/\*\*

\* 输入框获取焦点

\*/

onFocus(e) {

this.setData({

isFocused: true,

showPanel: true

});

this.triggerEvent('focus', {

value: e.detail.value

});

},

/\*\*

\* 输入框失去焦点

\*/

onBlur(e) {

// 延迟隐藏面板，防止点击历史记录时面板消失

setTimeout(() => {

this.setData({

isFocused: false,

showPanel: false

});

}, 200);

this.triggerEvent('blur', {

value: e.detail.value

});

},

/\*\*

\* 确认搜索（回车）

\*/

onConfirm(e) {

const value = e.detail.value.trim();

if (value) {

this.performSearch(value);

}

},

/\*\*

\* 点击搜索按钮

\*/

onSearch() {

const value = this.data.inputValue.trim();

if (value) {

this.performSearch(value);

}

},

/\*\*

\* 执行搜索

\*/

performSearch(value) {

// 添加到搜索历史

this.addToHistory(value);

// 隐藏面板

this.setData({

showPanel: false,

isFocused: false

});

// 触发搜索事件

this.triggerEvent('search', {

value: value

});

},

/\*\*

\* 清除输入内容

\*/

onClear() {

this.setData({

inputValue: ''

});

this.triggerEvent('clear');

this.triggerEvent('input', {

value: ''

});

},

/\*\*

\* 点击历史记录

\*/

onHistoryClick(e) {

const value = e.currentTarget.dataset.value;

this.setData({

inputValue: value

});

this.performSearch(value);

},

/\*\*

\* 删除历史记录项

\*/

onDeleteHistory(e) {

e.stopPropagation();

const index = e.currentTarget.dataset.index;

const history = [...this.data.searchHistory];

history.splice(index, 1);

this.setData({

searchHistory: history

});

this.saveSearchHistory(history);

},

/\*\*

\* 清空历史记录

\*/

onClearHistory() {

wx.showModal({

title: '确认清空',

content: '确定要清空所有搜索历史吗？',

success: (res) => {

if (res.confirm) {

this.setData({

searchHistory: []

});

this.saveSearchHistory([]);

this.triggerEvent('clearHistory');

}

}

});

},

/\*\*

\* 点击搜索建议

\*/

onSuggestionClick(e) {

const value = e.currentTarget.dataset.value;

this.setData({

inputValue: value

});

this.performSearch(value);

},

/\*\*

\* 添加到搜索历史

\*/

addToHistory(value) {

if (!this.data.showHistory) return;

let history = [...this.data.searchHistory];

// 如果已存在，先移除

const existIndex = history.indexOf(value);

if (existIndex > -1) {

history.splice(existIndex, 1);

}

// 添加到开头

history.unshift(value);

// 限制历史记录数量

if (history.length > this.data.maxHistory) {

history = history.slice(0, this.data.maxHistory);

}

this.setData({

searchHistory: history

});

this.saveSearchHistory(history);

},

/\*\*

\* 加载搜索历史

\*/

loadSearchHistory() {

if (!this.data.showHistory) return;

try {

const history = wx.getStorageSync('search\_history') || [];

this.setData({

searchHistory: history

});

} catch (error) {

console.error('加载搜索历史失败:', error);

}

},

/\*\*

\* 保存搜索历史

\*/

saveSearchHistory(history) {

try {

wx.setStorageSync('search\_history', history);

} catch (error) {

console.error('保存搜索历史失败:', error);

}

},

/\*\*

\* 获取输入框焦点

\*/

focus() {

this.setData({

isFocused: true,

showPanel: true

});

},

/\*\*

\* 失去输入框焦点

\*/

blur() {

this.setData({

isFocused: false,

showPanel: false

});

}

}

});

小程序搜索组件模板 - components/search/index.wxml

<!-- components/search/index.wxml -->

<view class="search-container">

<!-- 搜索框 -->

<view class="search-bar">

<view class="search-input-wrapper">

<!-- 搜索图标 -->

<view class="search-icon">

<text class="icon">🔍</text>

</view>

<!-- 输入框 -->

<input

class="search-input"

style="{{inputStyle}}"

type="text"

placeholder="{{placeholder}}"

placeholder-class="input-placeholder"

value="{{inputValue}}"

focus="{{autoFocus}}"

disabled="{{disabled}}"

bindinput="onInput"

bindfocus="onFocus"

bindblur="onBlur"

bindconfirm="onConfirm"

/>

<!-- 清除按钮 -->

<view

class="clear-btn"

wx:if="{{showClear && inputValue}}"

bindtap="onClear"

>

<text class="clear-icon">✕</text>

</view>

</view>

<!-- 搜索按钮 -->

<view

class="search-btn"

wx:if="{{showButton}}"

bindtap="onSearch"

>

<text class="search-btn-text">{{buttonText}}</text>

</view>

</view>

<!-- 搜索面板 -->

<view class="search-panel" wx:if="{{showPanel && isFocused}}">

<!-- 搜索建议 -->

<view class="suggestions-section" wx:if="{{showSuggestions && suggestions.length > 0 && inputValue}}">

<view class="section-title">搜索建议</view>

<view class="suggestions-list">

<view

class="suggestion-item"

wx:for="{{suggestions}}"

wx:key="index"

data-value="{{item}}"

bindtap="onSuggestionClick"

>

<view class="suggestion-icon">

<text class="icon">🔍</text>

</view>

<text class="suggestion-text">{{item}}</text>

</view>

</view>

</view>

<!-- 搜索历史 -->

<view class="history-section" wx:if="{{showHistory && searchHistory.length > 0 && !inputValue}}">

<view class="section-header">

<text class="section-title">搜索历史</text>

<view class="clear-history-btn" bindtap="onClearHistory">

<text class="clear-history-text">清空</text>

</view>

</view>

<view class="history-list">

<view

class="history-item"

wx:for="{{searchHistory}}"

wx:key="index"

data-value="{{item}}"

bindtap="onHistoryClick"

>

<view class="history-icon">

<text class="icon">🕒</text>

</view>

<text class="history-text">{{item}}</text>

<view

class="delete-history-btn"

data-index="{{index}}"

bindtap="onDeleteHistory"

catchtap="true"

>

<text class="delete-icon">✕</text>

</view>

</view>

</view>

</view>

<!-- 热门搜索 -->

<view class="hot-section" wx:if="{{showHotSearch && hotSearches.length > 0 && !inputValue}}">

<view class="section-title">热门搜索</view>

<view class="hot-list">

<view

class="hot-item"

wx:for="{{hotSearches}}"

wx:key="index"

data-value="{{item}}"

bindtap="onHotSearchClick"

>

<text class="hot-text">{{item}}</text>

</view>

</view>

</view>

<!-- 自定义内容插槽 -->

<view class="custom-content" wx:if="{{showCustomContent}}">

<slot name="custom-content"></slot>

</view>

</view>

<!-- 遮罩层 -->

<view

class="search-mask"

wx:if="{{showMask && isFocused}}"

bindtap="onMaskClick"

></view>

</view>

现在继续开发下一个组件 - 小程序搜索组件样式文件：

小程序搜索组件样式 - components/search/index.wxss

/\* components/search/index.wxss \*/

.search-container {

position: relative;

width: 100%;

z-index: 100;

}

/\* 搜索栏 \*/

.search-bar {

display: flex;

align-items: center;

padding: 20rpx;

background-color: #fff;

border-bottom: 1rpx solid #e8e8e8;

}

.search-input-wrapper {

flex: 1;

position: relative;

display: flex;

align-items: center;

background-color: #f5f5f5;

border-radius: 36rpx;

padding: 0 30rpx;

height: 72rpx;

margin-right: 20rpx;

}

.search-icon {

margin-right: 20rpx;

}

.search-icon .icon {

font-size: 32rpx;

color: #999;

}

.search-input {

flex: 1;

height: 100%;

font-size: 28rpx;

color: #333;

line-height: 1.4;

}

.input-placeholder {

color: #999;

font-size: 28rpx;

}

.clear-btn {

position: absolute;

right: 20rpx;

width: 40rpx;

height: 40rpx;

display: flex;

align-items: center;

justify-content: center;

background-color: #ccc;

border-radius: 50%;

}

.clear-icon {

font-size: 24rpx;

color: #fff;

line-height: 1;

}

.search-btn {

padding: 0 30rpx;

height: 72rpx;

display: flex;

align-items: center;

justify-content: center;

background-color: var(--primary-color, #2979ff);

border-radius: 36rpx;

min-width: 120rpx;

}

.search-btn-text {

color: #fff;

font-size: 28rpx;

font-weight: 500;

}

/\* 搜索面板 \*/

.search-panel {

position: absolute;

top: 100%;

left: 0;

right: 0;

background-color: #fff;

border: 1rpx solid #e8e8e8;

border-top: none;

border-radius: 0 0 16rpx 16rpx;

box-shadow: 0 8rpx 24rpx rgba(0, 0, 0, 0.1);

z-index: 1000;

max-height: 600rpx;

overflow-y: auto;

}

/\* 区域标题 \*/

.section-title {

font-size: 28rpx;

color: #666;

font-weight: 500;

margin-bottom: 20rpx;

}

.section-header {

display: flex;

align-items: center;

justify-content: space-between;

margin-bottom: 20rpx;

}

.clear-history-btn {

padding: 10rpx 20rpx;

}

.clear-history-text {

font-size: 24rpx;

color: #999;

}

/\* 搜索建议 \*/

.suggestions-section {

padding: 30rpx;

border-bottom: 1rpx solid #f0f0f0;

}

.suggestions-list {

display: flex;

flex-direction: column;

}

.suggestion-item {

display: flex;

align-items: center;

padding: 20rpx 0;

border-bottom: 1rpx solid #f8f8f8;

}

.suggestion-item:last-child {

border-bottom: none;

}

.suggestion-item:active {

background-color: #f5f5f5;

}

.suggestion-icon {

margin-right: 20rpx;

}

.suggestion-icon .icon {

font-size: 28rpx;

color: #999;

}

.suggestion-text {

flex: 1;

font-size: 28rpx;

color: #333;

line-height: 1.4;

}

/\* 搜索历史 \*/

.history-section {

padding: 30rpx;

border-bottom: 1rpx solid #f0f0f0;

}

.history-list {

display: flex;

flex-direction: column;

}

.history-item {

display: flex;

align-items: center;

padding: 20rpx 0;

border-bottom: 1rpx solid #f8f8f8;

position: relative;

}

.history-item:last-child {

border-bottom: none;

}

.history-item:active {

background-color: #f5f5f5;

}

.history-icon {

margin-right: 20rpx;

}

.history-icon .icon {

font-size: 28rpx;

color: #ccc;

}

.history-text {

flex: 1;

font-size: 28rpx;

color: #666;

line-height: 1.4;

}

.delete-history-btn {

padding: 10rpx;

margin-left: 20rpx;

}

.delete-icon {

font-size: 24rpx;

color: #ccc;

}

/\* 热门搜索 \*/

.hot-section {

padding: 30rpx;

}

.hot-list {

display: flex;

flex-wrap: wrap;

gap: 20rpx;

}

.hot-item {

padding: 12rpx 24rpx;

background-color: #f8f9fa;

border-radius: 32rpx;

border: 1rpx solid #e9ecef;

}

.hot-item:active {

background-color: #e9ecef;

}

.hot-text {

font-size: 26rpx;

color: #666;

line-height: 1.4;

}

/\* 自定义内容 \*/

.custom-content {

padding: 30rpx;

}

/\* 遮罩层 \*/

.search-mask {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0, 0, 0, 0.3);

z-index: 99;

}

/\* 响应式设计 \*/

@media (max-width: 600rpx) {

.search-bar {

padding: 15rpx;

}

.search-input-wrapper {

height: 68rpx;

padding: 0 25rpx;

}

.search-btn {

height: 68rpx;

padding: 0 25rpx;

min-width: 100rpx;

}

.search-panel {

max-height: 500rpx;

}

}

/\* 深色模式支持 \*/

@media (prefers-color-scheme: dark) {

.search-bar {

background-color: #1f1f1f;

border-bottom-color: #333;

}

.search-input-wrapper {

background-color: #333;

}

.search-input {

color: #fff;

}

.input-placeholder {

color: #999;

}

.search-panel {

background-color: #1f1f1f;

border-color: #333;

}

.section-title {

color: #ccc;

}

.suggestion-text {

color: #fff;

}

.history-text {

color: #ccc;

}

.hot-item {

background-color: #333;

border-color: #555;

}

.hot-text {

color: #ccc;

}

}

接下来开发小程序搜索组件配置文件：

小程序搜索组件配置 - components/search/index.json

{

"component": true,

"usingComponents": {},

"styleIsolation": "isolated"

}

现在开发小程序图片上传组件逻辑：

小程序图片上传组件 - components/image-upload/index.js

// components/image-upload/index.js

const app = getApp();

const api = require('../../utils/api');

const common = require('../../utils/common');

Component({

/\*\*

\* 组件的属性列表

\*/

properties: {

// 最大上传数量

maxCount: {

type: Number,

value: 9

},

// 已上传的图片列表

fileList: {

type: Array,

value: [],

observer: function(newVal) {

this.setData({

imageList: newVal || []

});

}

},

// 是否显示上传按钮

showUploadBtn: {

type: Boolean,

value: true

},

// 是否禁用

disabled: {

type: Boolean,

value: false

},

// 上传按钮文字

uploadText: {

type: String,

value: '上传图片'

},

// 图片大小限制（MB）

maxSize: {

type: Number,

value: 5

},

// 图片质量 0-1

quality: {

type: Number,

value: 0.8

},

// 是否压缩图片

compress: {

type: Boolean,

value: true

},

// 上传接口地址

uploadUrl: {

type: String,

value: '/api/v1/upload/image'

},

// 预览模式

previewMode: {

type: String,

value: 'default' // default, grid, list

},

// 是否显示删除按钮

showDelete: {

type: Boolean,

value: true

},

// 是否显示进度条

showProgress: {

type: Boolean,

value: true

},

// 自定义样式类

customClass: {

type: String,

value: ''

}

},

/\*\*

\* 组件的初始数据

\*/

data: {

imageList: [],

uploading: false,

uploadProgress: 0

},

/\*\*

\* 组件的方法列表

\*/

methods: {

/\*\*

\* 选择图片

\*/

chooseImage() {

if (this.data.disabled) {

return;

}

const remainCount = this.properties.maxCount - this.data.imageList.length;

if (remainCount <= 0) {

wx.showToast({

title: `最多只能上传${this.properties.maxCount}张图片`,

icon: 'none'

});

return;

}

wx.chooseMedia({

count: remainCount,

mediaType: ['image'],

sourceType: ['album', 'camera'],

maxDuration: 30,

camera: 'back',

success: (res) => {

this.handleImageSelect(res.tempFiles);

},

fail: (err) => {

console.error('选择图片失败:', err);

wx.showToast({

title: '选择图片失败',

icon: 'none'

});

}

});

},

/\*\*

\* 处理图片选择结果

\*/

handleImageSelect(tempFiles) {

const validFiles = [];

// 验证文件

for (let file of tempFiles) {

// 检查文件大小

if (file.size > this.properties.maxSize \* 1024 \* 1024) {

wx.showToast({

title: `图片大小不能超过${this.properties.maxSize}MB`,

icon: 'none'

});

continue;

}

// 检查文件类型

if (!file.tempFilePath.match(/\.(jpg|jpeg|png|gif|webp)$/i)) {

wx.showToast({

title: '只支持图片格式',

icon: 'none'

});

continue;

}

validFiles.push(file);

}

if (validFiles.length === 0) {

return;

}

// 如果需要压缩图片

if (this.properties.compress) {

this.compressImages(validFiles);

} else {

this.uploadImages(validFiles);

}

},

/\*\*

\* 压缩图片

\*/

compressImages(files) {

const compressedFiles = [];

let processedCount = 0;

wx.showLoading({

title: '处理图片中...'

});

files.forEach((file, index) => {

wx.compressImage({

src: file.tempFilePath,

quality: Math.floor(this.properties.quality \* 100),

success: (res) => {

compressedFiles[index] = {

...file,

tempFilePath: res.tempFilePath

};

},

fail: (err) => {

console.error('压缩图片失败:', err);

compressedFiles[index] = file;

},

complete: () => {

processedCount++;

if (processedCount === files.length) {

wx.hideLoading();

this.uploadImages(compressedFiles);

}

}

});

});

},

/\*\*

\* 上传图片

\*/

uploadImages(files) {

this.setData({

uploading: true,

uploadProgress: 0

});

const uploadPromises = files.map((file, index) => {

return this.uploadSingleImage(file, index, files.length);

});

Promise.all(uploadPromises).then((results) => {

const successResults = results.filter(item => item.success);

const failCount = results.length - successResults.length;

if (successResults.length > 0) {

const newImageList = [...this.data.imageList];

successResults.forEach(result => {

newImageList.push({

url: result.url,

thumbUrl: result.thumbUrl || result.url,

originalUrl: result.originalUrl || result.url,

id: result.id || common.generateId(),

uploadTime: new Date().toISOString()

});

});

this.setData({

imageList: newImageList

});

// 触发变更事件

this.triggerEvent('change', {

fileList: newImageList,

file: successResults[successResults.length - 1]

});

}

this.setData({

uploading: false,

uploadProgress: 0

});

// 显示结果提示

if (failCount > 0) {

wx.showToast({

title: `${successResults.length}张上传成功，${failCount}张失败`,

icon: 'none'

});

} else if (successResults.length > 0) {

wx.showToast({

title: '上传成功',

icon: 'success'

});

}

}).catch((error) => {

console.error('上传图片失败:', error);

this.setData({

uploading: false,

uploadProgress: 0

});

wx.showToast({

title: '上传失败',

icon: 'none'

});

});

},

/\*\*

\* 上传单个图片

\*/

uploadSingleImage(file, index, total) {

return new Promise((resolve) => {

const uploadTask = wx.uploadFile({

url: api.getFullUrl(this.properties.uploadUrl),

filePath: file.tempFilePath,

name: 'file',

header: {

'Authorization': `Bearer ${app.globalData.token}`

},

formData: {

type: 'image',

quality: this.properties.quality

},

success: (res) => {

try {

const data = JSON.parse(res.data);

if (data.success) {

resolve({

success: true,

url: data.data.url,

thumbUrl: data.data.thumbUrl,

originalUrl: data.data.originalUrl,

id: data.data.id

});

} else {

console.error('上传失败:', data.message);

resolve({ success: false, error: data.message });

}

} catch (err) {

console.error('解析上传结果失败:', err);

resolve({ success: false, error: '解析响应失败' });

}

},

fail: (err) => {

console.error('上传请求失败:', err);

resolve({ success: false, error: err.errMsg });

}

});

// 监听上传进度

if (this.properties.showProgress) {

uploadTask.onProgressUpdate((res) => {

const progress = Math.floor((index + res.progress / 100) / total \* 100);

this.setData({

uploadProgress: progress

});

});

}

});

},

/\*\*

\* 预览图片

\*/

previewImage(e) {

const index = e.currentTarget.dataset.index;

const urls = this.data.imageList.map(item => item.originalUrl || item.url);

wx.previewImage({

current: urls[index],

urls: urls,

fail: (err) => {

console.error('预览图片失败:', err);

}

});

// 触发预览事件

this.triggerEvent('preview', {

index: index,

url: urls[index],

fileList: this.data.imageList

});

},

/\*\*

\* 删除图片

\*/

deleteImage(e) {

if (this.data.disabled) {

return;

}

const index = e.currentTarget.dataset.index;

const imageItem = this.data.imageList[index];

wx.showModal({

title: '提示',

content: '确定要删除这张图片吗？',

success: (res) => {

if (res.confirm) {

const newImageList = [...this.data.imageList];

newImageList.splice(index, 1);

this.setData({

imageList: newImageList

});

// 触发删除事件

this.triggerEvent('delete', {

index: index,

file: imageItem,

fileList: newImageList

});

// 触发变更事件

this.triggerEvent('change', {

fileList: newImageList

});

}

}

});

},

/\*\*

\* 重新上传

\*/

retryUpload(e) {

const index = e.currentTarget.dataset.index;

// 这里可以实现重新上传逻辑

console.log('重新上传图片:', index);

},

/\*\*

\* 清空所有图片

\*/

clearAll() {

if (this.data.disabled || this.data.imageList.length === 0) {

return;

}

wx.showModal({

title: '提示',

content: '确定要清空所有图片吗？',

success: (res) => {

if (res.confirm) {

this.setData({

imageList: []

});

// 触发清空事件

this.triggerEvent('clear');

// 触发变更事件

this.triggerEvent('change', {

fileList: []

});

}

}

});

},

/\*\*

\* 获取文件列表

\*/

getFileList() {

return this.data.imageList;

},

/\*\*

\* 设置文件列表

\*/

setFileList(fileList) {

this.setData({

imageList: fileList || []

});

}

},

/\*\*

\* 组件生命周期

\*/

lifetimes: {

attached() {

// 组件加载时初始化

this.setData({

imageList: this.properties.fileList || []

});

}

}

});

现在开发小程序图片上传组件模板：

小程序图片上传组件模板 - components/image-upload/index.wxml

<!-- components/image-upload/index.wxml -->

<view class="image-upload-container {{customClass}}">

<!-- 网格模式 -->

<view class="image-grid" wx:if="{{previewMode === 'grid' || previewMode === 'default'}}">

<!-- 已上传图片列表 -->

<view

class="image-item"

wx:for="{{imageList}}"

wx:key="id"

data-index="{{index}}"

bindtap="previewImage"

>

<image

class="image-preview"

src="{{item.thumbUrl || item.url}}"

mode="aspectFill"

lazy-load="{{true}}"

/>

<!-- 删除按钮 -->

<view

class="delete-btn"

wx:if="{{showDelete && !disabled}}"

data-index="{{index}}"

bindtap="deleteImage"

catchtap="true"

>

<text class="delete-icon">✕</text>

</view>

<!-- 加载状态 -->

<view class="loading-mask" wx:if="{{item.loading}}">

<text class="loading-text">上传中...</text>

</view>

<!-- 错误状态 -->

<view class="error-mask" wx:if="{{item.error}}">

<text class="error-text">上传失败</text>

<view

class="retry-btn"

data-index="{{index}}"

bindtap="retryUpload"

catchtap="true"

>

重试

</view>

</view>

</view>

<!-- 上传按钮 -->

<view

class="upload-btn"

wx:if="{{showUploadBtn && imageList.length < maxCount && !disabled}}"

bindtap="chooseImage"

>

<view class="upload-icon">

<text class="icon">📷</text>

</view>

<text class="upload-text">{{uploadText}}</text>

<text class="upload-count">{{imageList.length}}/{{maxCount}}</text>

</view>

</view>

<!-- 列表模式 -->

<view class="image-list" wx:if="{{previewMode === 'list'}}">

<!-- 已上传图片列表 -->

<view

class="list-item"

wx:for="{{imageList}}"

wx:key="id"

>

<image

class="list-image"

src="{{item.thumbUrl || item.url}}"

mode="aspectFill"

data-index="{{index}}"

bindtap="previewImage"

/>

<view class="list-content">

<text class="list-name">图片{{index + 1}}</text>

<text class="list-time">{{item.uploadTime}}</text>

<!-- 进度条 -->

<view class="progress-bar" wx:if="{{item.loading && showProgress}}">

<view class="progress-fill" style="width: {{item.progress || 0}}%"></view>

</view>

<!-- 状态文字 -->

<text class="list-status" wx:if="{{item.loading}}">上传中...</text>

<text class="list-status error" wx:if="{{item.error}}">上传失败</text>

</view>

<view class="list-actions">

<!-- 重试按钮 -->

<view

class="action-btn retry"

wx:if="{{item.error}}"

data-index="{{index}}"

bindtap="retryUpload"

>

重试

</view>

<!-- 删除按钮 -->

<view

class="action-btn delete"

wx:if="{{showDelete && !disabled}}"

data-index="{{index}}"

bindtap="deleteImage"

>

删除

</view>

</view>

</view>

<!-- 上传按钮 -->

<view

class="list-upload-btn"

wx:if="{{showUploadBtn && imageList.length < maxCount && !disabled}}"

bindtap="chooseImage"

>

<view class="upload-icon">

<text class="icon">➕</text>

</view>

<view class="upload-content">

<text class="upload-title">{{uploadText}}</text>

<text class="upload-desc">支持JPG、PNG格式，单个文件不超过{{maxSize}}MB</text>

</view>

</view>

</view>

<!-- 全局上传进度 -->

<view class="global-progress" wx:if="{{uploading && showProgress}}">

<view class="progress-header">

<text class="progress-title">正在上传图片...</text>

<text class="progress-percent">{{uploadProgress}}%</text>

</view>

<view class="progress-bar">

<view class="progress-fill" style="width: {{uploadProgress}}%"></view>

</view>

</view>

<!-- 提示信息 -->

<view class="upload-tips" wx:if="{{showTips && tips}}">

<text class="tips-text">{{tips}}</text>

</view>

<!-- 错误信息 -->

<view class="error-tips" wx:if="{{errorMessage}}">

<text class="error-text">{{errorMessage}}</text>

</view>

</view>

<!-- 图片预览弹窗 -->

<view class="preview-modal" wx:if="{{previewVisible}}" bindtap="closePreview">

<view class="preview-container" catchtap="stopPropagation">

<view class="preview-header">

<text class="preview-title">图片预览 ({{previewIndex + 1}}/{{imageList.length}})</text>

<view class="preview-close" bindtap="closePreview">

<text class="close-icon">✕</text>

</view>

</view>

<swiper

class="preview-swiper"

current="{{previewIndex}}"

bindchange="onPreviewChange"

indicator-dots="{{imageList.length > 1}}"

indicator-color="rgba(255,255,255,0.5)"

indicator-active-color="#ffffff"

>

<swiper-item wx:for="{{imageList}}" wx:key="id">

<image

class="preview-image"

src="{{item.url}}"

mode="aspectFit"

bindload="onPreviewImageLoad"

binderror="onPreviewImageError"

/>

</swiper-item>

</swiper>

<view class="preview-actions" wx:if="{{imageList.length > 1}}">

<view

class="action-item"

bindtap="prevImage"

disabled="{{previewIndex === 0}}"

>

<text class="action-text">上一张</text>

</view>

<view

class="action-item"

bindtap="nextImage"

disabled="{{previewIndex === imageList.length - 1}}"

>

<text class="action-text">下一张</text>

</view>

</view>

</view>

</view>

现在继续开发下一个模块：**小程序图片上传组件样式 - components/image-upload/index.wxss**

小程序图片上传组件样式 - components/image-upload/index.wxss

/\* components/image-upload/index.wxss \*/

.image-upload-container {

width: 100%;

}

/\* 网格模式样式 \*/

.image-grid {

display: flex;

flex-wrap: wrap;

gap: 12rpx;

padding: 0;

}

.image-item {

position: relative;

width: 220rpx;

height: 220rpx;

border-radius: 12rpx;

overflow: hidden;

background-color: #f5f5f5;

}

.image-preview {

width: 100%;

height: 100%;

border-radius: 12rpx;

}

.delete-btn {

position: absolute;

top: -10rpx;

right: -10rpx;

width: 40rpx;

height: 40rpx;

background-color: #ff4444;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 2rpx 8rpx rgba(0,0,0,0.1);

z-index: 10;

}

.delete-icon {

color: #ffffff;

font-size: 24rpx;

font-weight: bold;

}

.loading-mask,

.error-mask {

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0,0,0,0.6);

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

border-radius: 12rpx;

}

.loading-text,

.error-text {

color: #ffffff;

font-size: 24rpx;

margin-bottom: 16rpx;

}

.retry-btn {

background-color: #1890ff;

color: #ffffff;

padding: 8rpx 16rpx;

border-radius: 6rpx;

font-size: 20rpx;

}

.upload-btn {

width: 220rpx;

height: 220rpx;

border: 2rpx dashed #d9d9d9;

border-radius: 12rpx;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

background-color: #fafafa;

transition: all 0.3s ease;

}

.upload-btn:active {

background-color: #f0f0f0;

border-color: #1890ff;

}

.upload-icon {

margin-bottom: 12rpx;

}

.upload-icon .icon {

font-size: 48rpx;

color: #999999;

}

.upload-text {

font-size: 24rpx;

color: #666666;

margin-bottom: 8rpx;

}

.upload-count {

font-size: 20rpx;

color: #999999;

}

/\* 列表模式样式 \*/

.image-list {

width: 100%;

}

.list-item {

display: flex;

align-items: center;

padding: 24rpx;

background-color: #ffffff;

border-radius: 12rpx;

margin-bottom: 16rpx;

box-shadow: 0 2rpx 8rpx rgba(0,0,0,0.05);

}

.list-image {

width: 120rpx;

height: 120rpx;

border-radius: 8rpx;

margin-right: 24rpx;

flex-shrink: 0;

}

.list-content {

flex: 1;

display: flex;

flex-direction: column;

justify-content: center;

}

.list-name {

font-size: 28rpx;

color: #333333;

font-weight: 500;

margin-bottom: 8rpx;

}

.list-time {

font-size: 22rpx;

color: #999999;

margin-bottom: 16rpx;

}

.progress-bar {

width: 100%;

height: 6rpx;

background-color: #f0f0f0;

border-radius: 3rpx;

overflow: hidden;

margin-bottom: 12rpx;

}

.progress-fill {

height: 100%;

background-color: #1890ff;

border-radius: 3rpx;

transition: width 0.3s ease;

}

.list-status {

font-size: 22rpx;

color: #1890ff;

}

.list-status.error {

color: #ff4444;

}

.list-actions {

display: flex;

flex-direction: column;

gap: 12rpx;

}

.action-btn {

padding: 12rpx 24rpx;

border-radius: 6rpx;

font-size: 22rpx;

text-align: center;

min-width: 80rpx;

}

.action-btn.retry {

background-color: #1890ff;

color: #ffffff;

}

.action-btn.delete {

background-color: #ff4444;

color: #ffffff;

}

.list-upload-btn {

display: flex;

align-items: center;

padding: 32rpx 24rpx;

background-color: #ffffff;

border: 2rpx dashed #d9d9d9;

border-radius: 12rpx;

margin-bottom: 16rpx;

transition: all 0.3s ease;

}

.list-upload-btn:active {

background-color: #f8f9fa;

border-color: #1890ff;

}

.list-upload-btn .upload-icon {

margin-right: 24rpx;

margin-bottom: 0;

}

.list-upload-btn .icon {

font-size: 40rpx;

color: #1890ff;

}

.upload-content {

flex: 1;

}

.upload-title {

font-size: 28rpx;

color: #333333;

font-weight: 500;

margin-bottom: 8rpx;

display: block;

}

.upload-desc {

font-size: 22rpx;

color: #999999;

}

/\* 全局上传进度 \*/

.global-progress {

background-color: #ffffff;

border-radius: 12rpx;

padding: 24rpx;

margin-top: 16rpx;

box-shadow: 0 2rpx 8rpx rgba(0,0,0,0.05);

}

.progress-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 16rpx;

}

.progress-title {

font-size: 28rpx;

color: #333333;

font-weight: 500;

}

.progress-percent {

font-size: 24rpx;

color: #1890ff;

font-weight: 500;

}

.global-progress .progress-bar {

height: 8rpx;

margin-bottom: 0;

}

/\* 提示信息 \*/

.upload-tips {

margin-top: 16rpx;

padding: 16rpx;

background-color: #f6ffed;

border: 1rpx solid #b7eb8f;

border-radius: 6rpx;

}

.tips-text {

font-size: 22rpx;

color: #52c41a;

line-height: 1.5;

}

.error-tips {

margin-top: 16rpx;

padding: 16rpx;

background-color: #fff2f0;

border: 1rpx solid #ffccc7;

border-radius: 6rpx;

}

.error-text {

font-size: 22rpx;

color: #ff4d4f;

line-height: 1.5;

}

/\* 图片预览弹窗 \*/

.preview-modal {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0,0,0,0.9);

z-index: 1000;

display: flex;

flex-direction: column;

}

.preview-container {

flex: 1;

display: flex;

flex-direction: column;

}

.preview-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 20rpx 40rpx;

background-color: rgba(0,0,0,0.5);

}

.preview-title {

color: #ffffff;

font-size: 32rpx;

font-weight: 500;

}

.preview-close {

width: 60rpx;

height: 60rpx;

display: flex;

align-items: center;

justify-content: center;

border-radius: 50%;

background-color: rgba(255,255,255,0.1);

}

.close-icon {

color: #ffffff;

font-size: 36rpx;

font-weight: bold;

}

.preview-swiper {

flex: 1;

width: 100%;

}

.preview-image {

width: 100%;

height: 100%;

}

.preview-actions {

display: flex;

justify-content: center;

gap: 80rpx;

padding: 40rpx;

background-color: rgba(0,0,0,0.5);

}

.action-item {

padding: 16rpx 32rpx;

background-color: rgba(255,255,255,0.1);

border-radius: 8rpx;

border: 1rpx solid rgba(255,255,255,0.2);

}

.action-item[disabled] {

opacity: 0.3;

}

.action-text {

color: #ffffff;

font-size: 28rpx;

}

/\* 响应式适配 \*/

@media (max-width: 750rpx) {

.image-item,

.upload-btn {

width: 200rpx;

height: 200rpx;

}

.image-grid {

gap: 8rpx;

}

.list-item {

padding: 20rpx;

}

.list-image {

width: 100rpx;

height: 100rpx;

margin-right: 20rpx;

}

}

接下来开发：**小程序图片上传组件配置 - components/image-upload/index.json**

小程序图片上传组件配置 - components/image-upload/index.json

{

"component": true,

"usingComponents": {}

}

现在开发：**订单创建页面逻辑 - pages/orders/create/create.js**

订单创建页面逻辑 - pages/orders/create/create.js

// pages/orders/create/create.js

const api = require('../../../utils/api');

const common = require('../../../utils/common');

Page({

data: {

productId: '',

product: null,

loading: false,

submitting: false,

// 用户信息

userInfo: null,

// 订单信息

orderInfo: {

productId: '',

serviceStartDate: '',

serviceEndDate: '',

capacity: '',

estimatedUsage: '',

currentPrice: '',

specialRequirements: '',

contactName: '',

contactPhone: '',

contactEmail: '',

companyAddress: ''

},

// 计算结果

calculation: {

estimatedAmount: 0,

estimatedSavings: 0,

annualSavings: 0,

serviceMonths: 12

},

// 表单验证

errors: {},

// 服务期限选项

servicePeriodOptions: [

{ value: 12, label: '12个月', popular: true },

{ value: 24, label: '24个月', popular: false },

{ value: 36, label: '36个月', popular: false }

],

selectedServicePeriod: 12,

// 日期选择

minDate: '',

maxDate: '',

// 协议确认

agreedToTerms: false,

showTermsModal: false

},

onLoad(options) {

const { productId } = options;

if (productId) {

this.setData({ productId });

this.loadProductInfo();

}

this.loadUserInfo();

this.initDateRange();

},

onShow() {

// 每次显示页面时重新加载用户信息，防止信息过期

this.loadUserInfo();

},

// 加载产品信息

async loadProductInfo() {

if (!this.data.productId) return;

try {

this.setData({ loading: true });

const product = await api.getProductDetail(this.data.productId);

this.setData({

product,

'orderInfo.productId': this.data.productId

});

// 如果用户信息已加载，进行效益计算

if (this.data.userInfo) {

this.calculateBenefit();

}

} catch (error) {

console.error('加载产品信息失败:', error);

common.showToast('加载产品信息失败', 'error');

} finally {

this.setData({ loading: false });

}

},

// 加载用户信息

async loadUserInfo() {

try {

const userInfo = await api.getUserInfo();

this.setData({

userInfo,

'orderInfo.contactName': userInfo.name || '',

'orderInfo.contactPhone': userInfo.phone || '',

'orderInfo.contactEmail': userInfo.email || '',

'orderInfo.companyAddress': userInfo.companyAddress || '',

'orderInfo.capacity': userInfo.powerConsumption?.capacity || '',

'orderInfo.estimatedUsage': userInfo.powerConsumption?.monthlyUsage || '',

'orderInfo.currentPrice': userInfo.powerConsumption?.currentPrice || ''

});

// 如果产品信息已加载，进行效益计算

if (this.data.product) {

this.calculateBenefit();

}

} catch (error) {

console.error('加载用户信息失败:', error);

common.showToast('加载用户信息失败', 'error');

}

},

// 初始化日期范围

initDateRange() {

const today = new Date();

const minDate = common.formatDate(today);

// 最大开始日期为6个月后

const maxDate = new Date();

maxDate.setMonth(maxDate.getMonth() + 6);

this.setData({

minDate,

maxDate: common.formatDate(maxDate),

'orderInfo.serviceStartDate': minDate

});

// 设置默认结束日期

this.calculateEndDate();

},

// 计算结束日期

calculateEndDate() {

const { serviceStartDate } = this.data.orderInfo;

const { selectedServicePeriod } = this.data;

if (serviceStartDate) {

const startDate = new Date(serviceStartDate);

const endDate = new Date(startDate);

endDate.setMonth(endDate.getMonth() + selectedServicePeriod);

this.setData({

'orderInfo.serviceEndDate': common.formatDate(endDate),

'calculation.serviceMonths': selectedServicePeriod

});

this.calculateBenefit();

}

},

// 效益计算

calculateBenefit() {

const { orderInfo, product, calculation } = this.data;

if (!product || !orderInfo.capacity || !orderInfo.estimatedUsage || !orderInfo.currentPrice) {

return;

}

try {

const capacity = parseFloat(orderInfo.capacity) || 0;

const monthlyUsage = parseFloat(orderInfo.estimatedUsage) || 0;

const currentPrice = parseFloat(orderInfo.currentPrice) || 0;

const serviceMonths = calculation.serviceMonths;

// 根据产品类型计算新电价

let newPrice = currentPrice;

if (product.priceType === 'fixed') {

newPrice = product.price;

} else if (product.priceType === 'discount') {

newPrice = currentPrice \* (1 - product.discountRate / 100);

}

// 计算费用和节省

const currentMonthlyAmount = monthlyUsage \* currentPrice;

const newMonthlyAmount = monthlyUsage \* newPrice;

const monthlySavings = currentMonthlyAmount - newMonthlyAmount;

const totalAmount = newMonthlyAmount \* serviceMonths;

const totalSavings = monthlySavings \* serviceMonths;

const annualSavings = monthlySavings \* 12;

this.setData({

'calculation.estimatedAmount': totalAmount,

'calculation.estimatedSavings': totalSavings,

'calculation.annualSavings': annualSavings

});

} catch (error) {

console.error('效益计算失败:', error);

}

},

// 输入事件处理

onInputChange(e) {

const { field } = e.currentTarget.dataset;

const { value } = e.detail;

this.setData({

[`orderInfo.${field}`]: value,

[`errors.${field}`]: '' // 清除错误信息

});

// 如果是影响计算的字段，重新计算

if (['capacity', 'estimatedUsage', 'currentPrice'].includes(field)) {

this.calculateBenefit();

}

},

// 服务期限选择

onServicePeriodChange(e) {

const period = parseInt(e.currentTarget.dataset.period);

this.setData({ selectedServicePeriod: period });

this.calculateEndDate();

},

// 服务开始日期选择

onStartDateChange(e) {

this.setData({

'orderInfo.serviceStartDate': e.detail.value,

'errors.serviceStartDate': ''

});

this.calculateEndDate();

},

// 协议确认

onAgreeChange(e) {

this.setData({

agreedToTerms: e.detail.value,

'errors.agreedToTerms': ''

});

},

// 显示服务协议

showTerms() {

this.setData({ showTermsModal: true });

},

// 关闭服务协议

closeTerms() {

this.setData({ showTermsModal: false });

},

// 表单验证

validateForm() {

const { orderInfo, agreedToTerms } = this.data;

const errors = {};

// 必填字段验证

const requiredFields = {

'contactName': '联系人姓名',

'contactPhone': '联系电话',

'capacity': '用电容量',

'estimatedUsage': '预估月用电量',

'currentPrice': '当前电价',

'serviceStartDate': '服务开始日期'

};

Object.keys(requiredFields).forEach(field => {

if (!orderInfo[field] || orderInfo[field].trim() === '') {

errors[field] = `请填写${requiredFields[field]}`;

}

});

// 电话号码格式验证

if (orderInfo.contactPhone && !common.validatePhone(orderInfo.contactPhone)) {

errors.contactPhone = '请输入正确的手机号码';

}

// 邮箱格式验证（如果填写了邮箱）

if (orderInfo.contactEmail && !common.validateEmail(orderInfo.contactEmail)) {

errors.contactEmail = '请输入正确的邮箱地址';

}

// 数值验证

if (orderInfo.capacity && (isNaN(orderInfo.capacity) || parseFloat(orderInfo.capacity) <= 0)) {

errors.capacity = '请输入正确的用电容量';

}

if (orderInfo.estimatedUsage && (isNaN(orderInfo.estimatedUsage) || parseFloat(orderInfo.estimatedUsage) <= 0)) {

errors.estimatedUsage = '请输入正确的月用电量';

}

if (orderInfo.currentPrice && (isNaN(orderInfo.currentPrice) || parseFloat(orderInfo.currentPrice) <= 0)) {

errors.currentPrice = '请输入正确的当前电价';

}

// 协议确认验证

if (!agreedToTerms) {

errors.agreedToTerms = '请阅读并同意服务协议';

}

this.setData({ errors });

return Object.keys(errors).length === 0;

},

// 提交订单

async submitOrder() {

if (!this.validateForm()) {

const firstErrorField = Object.keys(this.data.errors)[0];

common.showToast(this.data.errors[firstErrorField], 'error');

return;

}

if (this.data.submitting) return;

try {

this.setData({ submitting: true });

const orderData = {

...this.data.orderInfo,

estimatedAmount: this.data.calculation.estimatedAmount,

estimatedSavings: this.data.calculation.estimatedSavings,

serviceMonths: this.data.calculation.serviceMonths

};

const result = await api.createOrder(orderData);

common.showToast('订单提交成功', 'success');

// 跳转到订单详情页

setTimeout(() => {

wx.redirectTo({

url: `/pages/orders/detail/detail?id=${result.orderId}`

});

}, 1500);

} catch (error) {

console.error('订单提交失败:', error);

common.showToast(error.message || '订单提交失败，请重试', 'error');

} finally {

this.setData({ submitting: false });

}

},

// 重新计算

recalculate() {

this.calculateBenefit();

common.showToast('重新计算完成', 'success');

},

// 返回产品详情

goBack() {

wx.navigateBack();

},

// 联系客服

contactService() {

common.makePhoneCall('400-800-8888');

}

});

继续开发：**订单创建页面模板 - pages/orders/create/create.wxml**

订单创建页面模板 - pages/orders/create/create.wxml

<!-- pages/orders/create/create.wxml -->

<view class="create-order-container">

<!-- 加载状态 -->

<loading wx:if="{{loading}}" />

<!-- 页面内容 -->

<view class="page-content" wx:else>

<!-- 产品信息卡片 -->

<view class="product-card" wx:if="{{product}}">

<view class="product-header">

<text class="product-name">{{product.name}}</text>

<text class="product-type">{{product.typeName}}</text>

</view>

<view class="product-price">

<text class="price-label">参考电价：</text>

<text class="price-value">{{product.priceDisplay}}</text>

</view>

<view class="product-desc">{{product.shortDescription}}</view>

</view>

<!-- 联系信息 -->

<view class="form-section">

<view class="section-title">

<text class="title-text">联系信息</text>

<text class="required-mark">\*</text>

</view>

<view class="form-item">

<text class="label">联系人</text>

<input

class="input {{errors.contactName ? 'error' : ''}}"

placeholder="请输入联系人姓名"

value="{{orderInfo.contactName}}"

data-field="contactName"

bindinput="onInputChange"

/>

<text class="error-text" wx:if="{{errors.contactName}}">{{errors.contactName}}</text>

</view>

<view class="form-item">

<text class="label">联系电话</text>

<input

class="input {{errors.contactPhone ? 'error' : ''}}"

placeholder="请输入联系电话"

type="number"

value="{{orderInfo.contactPhone}}"

data-field="contactPhone"

bindinput="onInputChange"

/>

<text class="error-text" wx:if="{{errors.contactPhone}}">{{errors.contactPhone}}</text>

</view>

<view class="form-item">

<text class="label">企业名称</text>

<input

class="input {{errors.companyName ? 'error' : ''}}"

placeholder="请输入企业名称"

value="{{orderInfo.companyName}}"

data-field="companyName"

bindinput="onInputChange"

/>

<text class="error-text" wx:if="{{errors.companyName}}">{{errors.companyName}}</text>

</view>

</view>

<!-- 用电信息 -->

<view class="form-section">

<view class="section-title">

<text class="title-text">用电信息</text>

<text class="required-mark">\*</text>

</view>

<view class="form-item">

<text class="label">用电容量(kW)</text>

<input

class="input {{errors.powerCapacity ? 'error' : ''}}"

placeholder="请输入用电容量"

type="digit"

value="{{orderInfo.powerCapacity}}"

data-field="powerCapacity"

bindinput="onInputChange"

/>

<text class="error-text" wx:if="{{errors.powerCapacity}}">{{errors.powerCapacity}}</text>

</view>

<view class="form-item">

<text class="label">月用电量(kWh)</text>

<input

class="input {{errors.monthlyUsage ? 'error' : ''}}"

placeholder="请输入月用电量"

type="digit"

value="{{orderInfo.monthlyUsage}}"

data-field="monthlyUsage"

bindinput="onInputChange"

/>

<text class="error-text" wx:if="{{errors.monthlyUsage}}">{{errors.monthlyUsage}}</text>

</view>

<view class="form-item">

<text class="label">当前电价(元/kWh)</text>

<input

class="input {{errors.currentPrice ? 'error' : ''}}"

placeholder="请输入当前电价"

type="digit"

value="{{orderInfo.currentPrice}}"

data-field="currentPrice"

bindinput="onInputChange"

/>

<text class="error-text" wx:if="{{errors.currentPrice}}">{{errors.currentPrice}}</text>

</view>

<view class="form-item">

<text class="label">行业类型</text>

<picker

mode="selector"

range="{{industryTypes}}"

range-key="name"

value="{{industryIndex}}"

bindchange="onIndustryChange"

>

<view class="picker {{errors.industryType ? 'error' : ''}}">

<text class="picker-text">{{orderInfo.industryType || '请选择行业类型'}}</text>

<text class="picker-arrow">></text>

</view>

</picker>

<text class="error-text" wx:if="{{errors.industryType}}">{{errors.industryType}}</text>

</view>

</view>

<!-- 服务期限 -->

<view class="form-section">

<view class="section-title">

<text class="title-text">服务期限</text>

<text class="required-mark">\*</text>

</view>

<view class="form-item">

<text class="label">服务开始日期</text>

<picker

mode="date"

value="{{orderInfo.serviceStartDate}}"

start="{{minDate}}"

bindchange="onStartDateChange"

>

<view class="picker {{errors.serviceStartDate ? 'error' : ''}}">

<text class="picker-text">{{orderInfo.serviceStartDate || '请选择开始日期'}}</text>

<text class="picker-arrow">></text>

</view>

</picker>

<text class="error-text" wx:if="{{errors.serviceStartDate}}">{{errors.serviceStartDate}}</text>

</view>

<view class="form-item">

<text class="label">服务期限</text>

<picker

mode="selector"

range="{{servicePeriods}}"

range-key="name"

value="{{periodIndex}}"

bindchange="onPeriodChange"

>

<view class="picker {{errors.servicePeriod ? 'error' : ''}}">

<text class="picker-text">{{orderInfo.servicePeriod || '请选择服务期限'}}</text>

<text class="picker-arrow">></text>

</view>

</picker>

<text class="error-text" wx:if="{{errors.servicePeriod}}">{{errors.servicePeriod}}</text>

</view>

<view class="form-item" wx:if="{{orderInfo.serviceEndDate}}">

<text class="label">服务结束日期</text>

<view class="readonly-field">{{orderInfo.serviceEndDate}}</view>

</view>

</view>

<!-- 费用信息 -->

<view class="form-section" wx:if="{{costInfo.totalAmount}}">

<view class="section-title">

<text class="title-text">费用信息</text>

</view>

<view class="cost-item">

<text class="cost-label">预估年节省金额</text>

<text class="cost-value">￥{{costInfo.annualSavings}}</text>

</view>

<view class="cost-item">

<text class="cost-label">服务费用</text>

<text class="cost-value">￥{{costInfo.serviceFee}}</text>

</view>

<view class="cost-item total">

<text class="cost-label">预估总金额</text>

<text class="cost-value">￥{{costInfo.totalAmount}}</text>

</view>

</view>

<!-- 特殊需求 -->

<view class="form-section">

<view class="section-title">

<text class="title-text">特殊需求</text>

<text class="optional-mark">(可选)</text>

</view>

<view class="form-item">

<textarea

class="textarea"

placeholder="请描述您的特殊需求或备注信息"

value="{{orderInfo.remark}}"

data-field="remark"

bindinput="onInputChange"

maxlength="500"

/>

<view class="char-count">{{orderInfo.remark.length}}/500</view>

</view>

</view>

<!-- 协议确认 -->

<view class="agreement-section">

<view class="agreement-item" bindtap="toggleAgreement">

<view class="checkbox {{agreedToTerms ? 'checked' : ''}}">

<text class="checkbox-icon" wx:if="{{agreedToTerms}}">✓</text>

</view>

<text class="agreement-text">

我已阅读并同意

<text class="link" bindtap="viewTerms">《服务协议》</text>

和

<text class="link" bindtap="viewPrivacy">《隐私政策》</text>

</text>

</view>

<text class="error-text" wx:if="{{errors.agreement}}">{{errors.agreement}}</text>

</view>

<!-- 提交按钮 -->

<view class="submit-section">

<button

class="submit-btn {{canSubmit ? 'active' : 'disabled'}}"

disabled="{{!canSubmit || submitting}}"

bindtap="onSubmit"

>

<text wx:if="{{submitting}}">提交中...</text>

<text wx:else>确认提交订单</text>

</button>

<view class="tips">

<text class="tips-text">提交后我们将在24小时内与您联系确认需求</text>

</view>

</view>

</view>

<!-- 模态框组件 -->

<modal

wx:if="{{showModal}}"

title="{{modalTitle}}"

content="{{modalContent}}"

confirmText="{{modalConfirmText}}"

cancelText="{{modalCancelText}}"

showCancel="{{modalShowCancel}}"

bind:confirm="onModalConfirm"

bind:cancel="onModalCancel"

/>

</view>

现在继续开发下一个模块：**订单创建页面样式文件 - pages/orders/create/create.wxss**

订单创建页面样式文件 - pages/orders/create/create.wxss

/\* pages/orders/create/create.wxss \*/

.create-order-container {

min-height: 100vh;

background-color: #f5f6fa;

}

.page-content {

padding-bottom: 120rpx;

}

/\* 产品信息卡片 \*/

.product-card {

margin: 20rpx;

padding: 30rpx;

background-color: #fff;

border-radius: 16rpx;

box-shadow: 0 4rpx 12rpx rgba(0, 0, 0, 0.05);

}

.product-header {

display: flex;

align-items: center;

justify-content: space-between;

margin-bottom: 16rpx;

}

.product-name {

font-size: 32rpx;

font-weight: 600;

color: #1a1a1a;

}

.product-type {

padding: 8rpx 16rpx;

background-color: #e8f4fd;

color: #1890ff;

font-size: 24rpx;

border-radius: 20rpx;

}

.product-price {

display: flex;

align-items: center;

margin-bottom: 16rpx;

}

.price-label {

font-size: 28rpx;

color: #666;

margin-right: 12rpx;

}

.price-value {

font-size: 32rpx;

font-weight: 600;

color: #ff4d4f;

}

.product-desc {

font-size: 26rpx;

color: #999;

line-height: 1.5;

}

/\* 表单区域 \*/

.form-section {

margin: 20rpx;

padding: 30rpx;

background-color: #fff;

border-radius: 16rpx;

box-shadow: 0 4rpx 12rpx rgba(0, 0, 0, 0.05);

}

.section-title {

display: flex;

align-items: center;

margin-bottom: 30rpx;

padding-bottom: 16rpx;

border-bottom: 2rpx solid #f0f0f0;

}

.title-text {

font-size: 32rpx;

font-weight: 600;

color: #1a1a1a;

}

.required-mark {

margin-left: 8rpx;

color: #ff4d4f;

font-size: 28rpx;

}

.optional-mark {

margin-left: 8rpx;

color: #999;

font-size: 24rpx;

}

/\* 表单项 \*/

.form-item {

margin-bottom: 30rpx;

}

.form-item:last-child {

margin-bottom: 0;

}

.label {

display: block;

font-size: 28rpx;

color: #333;

margin-bottom: 16rpx;

font-weight: 500;

}

.input {

width: 100%;

padding: 24rpx 20rpx;

background-color: #fafafa;

border: 2rpx solid #e8e8e8;

border-radius: 12rpx;

font-size: 30rpx;

color: #333;

box-sizing: border-box;

}

.input:focus {

border-color: #1890ff;

background-color: #fff;

}

.input.error {

border-color: #ff4d4f;

background-color: #fff6f6;

}

.input::placeholder {

color: #bfbfbf;

}

.textarea {

width: 100%;

min-height: 160rpx;

padding: 24rpx 20rpx;

background-color: #fafafa;

border: 2rpx solid #e8e8e8;

border-radius: 12rpx;

font-size: 30rpx;

color: #333;

box-sizing: border-box;

line-height: 1.6;

}

.textarea:focus {

border-color: #1890ff;

background-color: #fff;

}

.char-count {

text-align: right;

font-size: 24rpx;

color: #999;

margin-top: 8rpx;

}

/\* 选择器样式 \*/

.picker {

padding: 24rpx 20rpx;

background-color: #fafafa;

border: 2rpx solid #e8e8e8;

border-radius: 12rpx;

display: flex;

align-items: center;

justify-content: space-between;

}

.picker.error {

border-color: #ff4d4f;

background-color: #fff6f6;

}

.picker-text {

font-size: 30rpx;

color: #333;

flex: 1;

}

.picker-text:empty::before {

content: attr(placeholder);

color: #bfbfbf;

}

.picker-arrow {

font-size: 24rpx;

color: #bfbfbf;

margin-left: 16rpx;

}

.readonly-field {

padding: 24rpx 20rpx;

background-color: #f5f5f5;

border-radius: 12rpx;

font-size: 30rpx;

color: #666;

}

/\* 费用信息 \*/

.cost-item {

display: flex;

align-items: center;

justify-content: space-between;

padding: 24rpx 0;

border-bottom: 1rpx solid #f0f0f0;

}

.cost-item:last-child {

border-bottom: none;

}

.cost-item.total {

margin-top: 16rpx;

padding-top: 24rpx;

border-top: 2rpx solid #f0f0f0;

}

.cost-label {

font-size: 28rpx;

color: #666;

}

.cost-value {

font-size: 32rpx;

font-weight: 600;

color: #333;

}

.cost-item.total .cost-label,

.cost-item.total .cost-value {

font-size: 34rpx;

font-weight: 600;

color: #1890ff;

}

/\* 协议确认 \*/

.agreement-section {

margin: 20rpx;

padding: 30rpx;

background-color: #fff;

border-radius: 16rpx;

box-shadow: 0 4rpx 12rpx rgba(0, 0, 0, 0.05);

}

.agreement-item {

display: flex;

align-items: flex-start;

}

.checkbox {

width: 32rpx;

height: 32rpx;

border: 2rpx solid #d9d9d9;

border-radius: 6rpx;

display: flex;

align-items: center;

justify-content: center;

margin-right: 16rpx;

margin-top: 4rpx;

flex-shrink: 0;

}

.checkbox.checked {

background-color: #1890ff;

border-color: #1890ff;

}

.checkbox-icon {

color: #fff;

font-size: 20rpx;

font-weight: bold;

}

.agreement-text {

font-size: 26rpx;

color: #666;

line-height: 1.6;

flex: 1;

}

.link {

color: #1890ff;

text-decoration: underline;

}

/\* 错误提示 \*/

.error-text {

display: block;

margin-top: 8rpx;

font-size: 24rpx;

color: #ff4d4f;

line-height: 1.4;

}

/\* 提交区域 \*/

.submit-section {

position: fixed;

bottom: 0;

left: 0;

right: 0;

padding: 20rpx;

background-color: #fff;

border-top: 1rpx solid #f0f0f0;

box-shadow: 0 -4rpx 12rpx rgba(0, 0, 0, 0.1);

z-index: 100;

}

.submit-btn {

width: 100%;

height: 88rpx;

background: linear-gradient(135deg, #1890ff 0%, #096dd9 100%);

border-radius: 44rpx;

border: none;

font-size: 32rpx;

font-weight: 600;

color: #fff;

display: flex;

align-items: center;

justify-content: center;

margin-bottom: 16rpx;

}

.submit-btn.active {

background: linear-gradient(135deg, #1890ff 0%, #096dd9 100%);

}

.submit-btn.disabled {

background: #d9d9d9;

color: #bfbfbf;

}

.submit-btn::after {

border: none;

}

.tips {

text-align: center;

}

.tips-text {

font-size: 24rpx;

color: #999;

}

/\* 响应式适配 \*/

@media (max-width: 320px) {

.form-section {

margin: 16rpx;

padding: 24rpx;

}

.product-card {

margin: 16rpx;

padding: 24rpx;

}

.agreement-section {

margin: 16rpx;

padding: 24rpx;

}

}

接下来开发：**订单创建页面配置文件 - pages/orders/create/create.json**

订单创建页面配置文件 - pages/orders/create/create.json

{

"navigationBarTitleText": "创建订单",

"navigationBarBackgroundColor": "#1890ff",

"navigationBarTextStyle": "white",

"backgroundColor": "#f5f6fa",

"backgroundTextStyle": "dark",

"enablePullDownRefresh": false,

"onReachBottomDistance": 50,

"usingComponents": {

"loading": "/components/loading/index",

"modal": "/components/modal/index"

}

}

现在继续开发下一个模块：**计算器页面逻辑 - pages/calculator/index/index.js**

计算器页面逻辑 - pages/calculator/index/index.js

// pages/calculator/index/index.js

const api = require('../../../utils/api');

const common = require('../../../utils/common');

Page({

data: {

// 页面状态

loading: false,

// 用电信息输入

powerInfo: {

capacity: '', // 用电容量(kW)

monthlyUsage: '', // 月用电量(kWh)

currentPrice: '', // 当前电价(元/kWh)

industryType: '', // 行业类型

usagePattern: '工业用电' // 用电性质

},

// 行业类型选项

industryTypes: [

{ id: 'manufacturing', name: '制造业', coefficient: 1.0 },

{ id: 'commerce', name: '商业', coefficient: 0.95 },

{ id: 'service', name: '服务业', coefficient: 0.9 },

{ id: 'logistics', name: '物流仓储', coefficient: 1.05 },

{ id: 'datacenter', name: '数据中心', coefficient: 1.1 },

{ id: 'other', name: '其他', coefficient: 1.0 }

],

industryIndex: -1,

// 用电性质选项

usagePatterns: ['工业用电', '商业用电', '居民用电'],

usageIndex: 0,

// 产品列表

products: [],

// 计算结果

calculationResults: [],

showResults: false,

// 表单验证错误

errors: {},

// 计算详情

calculationDetails: null,

showDetails: false,

// 推荐产品

recommendedProduct: null,

// 页面提示

tips: [

'输入准确的用电信息，可获得更精确的节费计算',

'不同行业类型会影响节费效果',

'建议上传最近3个月的电费单以获得精准计算'

],

currentTip: 0

},

onLoad(options) {

console.log('Calculator page loaded with options:', options);

// 如果有传入的产品ID，则预设计算该产品

if (options.productId) {

this.setData({

presetProductId: options.productId

});

}

this.initPage();

},

onShow() {

// 从用户信息中预填充数据

this.loadUserPowerInfo();

// 启动提示轮播

this.startTipsRotation();

},

onHide() {

// 停止提示轮播

this.stopTipsRotation();

},

onUnload() {

this.stopTipsRotation();

},

// 初始化页面

async initPage() {

try {

this.setData({ loading: true });

await this.loadProducts();

} catch (error) {

console.error('Init page error:', error);

common.showToast('页面加载失败');

} finally {

this.setData({ loading: false });

}

},

// 加载产品列表

async loadProducts() {

try {

const result = await api.request('/products', 'GET');

if (result.success) {

this.setData({

products: result.data.map(product => ({

...product,

priceDisplay: this.formatPriceRange(product.priceRange)

}))

});

}

} catch (error) {

console.error('Load products error:', error);

}

},

// 加载用户用电信息

async loadUserPowerInfo() {

try {

const userInfo = wx.getStorageSync('userInfo');

if (userInfo && userInfo.powerConsumption) {

const powerInfo = userInfo.powerConsumption;

// 查找行业类型索引

let industryIndex = -1;

if (powerInfo.industryType) {

industryIndex = this.data.industryTypes.findIndex(

item => item.id === powerInfo.industryType

);

}

this.setData({

'powerInfo.capacity': powerInfo.capacity || '',

'powerInfo.monthlyUsage': powerInfo.monthlyUsage || '',

'powerInfo.currentPrice': powerInfo.currentPrice || '',

'powerInfo.industryType': powerInfo.industryType || '',

industryIndex: industryIndex

});

}

} catch (error) {

console.error('Load user power info error:', error);

}

},

// 输入框变化处理

onInputChange(e) {

const { field } = e.currentTarget.dataset;

const { value } = e.detail;

this.setData({

[`powerInfo.${field}`]: value,

[`errors.${field}`]: '' // 清除错误提示

});

// 实时计算（防抖）

if (this.calculateTimer) {

clearTimeout(this.calculateTimer);

}

this.calculateTimer = setTimeout(() => {

this.autoCalculate();

}, 500);

},

// 行业类型选择

onIndustryChange(e) {

const index = parseInt(e.detail.value);

const industry = this.data.industryTypes[index];

this.setData({

industryIndex: index,

'powerInfo.industryType': industry.id

});

// 重新计算

this.autoCalculate();

},

// 用电性质选择

onUsagePatternChange(e) {

const index = parseInt(e.detail.value);

this.setData({

usageIndex: index,

'powerInfo.usagePattern': this.data.usagePatterns[index]

});

// 重新计算

this.autoCalculate();

},

// 表单验证

validateForm() {

const { powerInfo } = this.data;

const errors = {};

if (!powerInfo.capacity || parseFloat(powerInfo.capacity) <= 0) {

errors.capacity = '请输入有效的用电容量';

}

if (!powerInfo.monthlyUsage || parseFloat(powerInfo.monthlyUsage) <= 0) {

errors.monthlyUsage = '请输入有效的月用电量';

}

if (!powerInfo.currentPrice || parseFloat(powerInfo.currentPrice) <= 0) {

errors.currentPrice = '请输入有效的当前电价';

}

if (!powerInfo.industryType) {

errors.industryType = '请选择行业类型';

}

this.setData({ errors });

return Object.keys(errors).length === 0;

},

// 自动计算（当输入完整时）

autoCalculate() {

const { powerInfo } = this.data;

// 检查必要字段是否填写完整

if (powerInfo.capacity && powerInfo.monthlyUsage &&

powerInfo.currentPrice && powerInfo.industryType) {

this.performCalculation();

}

},

// 手动计算按钮

onCalculate() {

if (!this.validateForm()) {

common.showToast('请完善用电信息');

return;

}

this.performCalculation();

},

// 执行计算

async performCalculation() {

try {

this.setData({ loading: true });

const { powerInfo, products } = this.data;

// 调用计算API

const result = await api.request('/calculator/calculate', 'POST', {

powerInfo,

productIds: products.map(p => p.id)

});

if (result.success) {

const calculationResults = result.data.results;

const calculationDetails = result.data.details;

// 按节费金额排序

calculationResults.sort((a, b) => b.annualSavings - a.annualSavings);

// 找出推荐产品（节费最多的）

const recommendedProduct = calculationResults[0];

this.setData({

calculationResults,

calculationDetails,

recommendedProduct,

showResults: true

});

// 保存计算历史

this.saveCalculationHistory();

// 滚动到结果区域

setTimeout(() => {

wx.pageScrollTo({

selector: '.results-section',

duration: 300

});

}, 100);

} else {

common.showToast(result.message || '计算失败');

}

} catch (error) {

console.error('Calculation error:', error);

common.showToast('计算失败，请稍后重试');

} finally {

this.setData({ loading: false });

}

},

// 保存计算历史

async saveCalculationHistory() {

try {

const { powerInfo, calculationResults, recommendedProduct } = this.data;

const historyData = {

powerInfo,

calculationResults: calculationResults.slice(0, 3), // 只保存前3个结果

recommendedProduct,

calculatedAt: new Date().toISOString()

};

// 保存到本地存储

let calculationHistory = wx.getStorageSync('calculationHistory') || [];

calculationHistory.unshift(historyData);

// 最多保存10条历史记录

if (calculationHistory.length > 10) {

calculationHistory = calculationHistory.slice(0, 10);

}

wx.setStorageSync('calculationHistory', calculationHistory);

// 同步到服务器

api.request('/calculator/history', 'POST', historyData).catch(err => {

console.log('Save history to server failed:', err);

});

} catch (error) {

console.error('Save calculation history error:', error);

}

},

// 查看计算详情

onViewDetails(e) {

const { productId } = e.currentTarget.dataset;

const result = this.data.calculationResults.find(r => r.productId === productId);

if (result) {

this.setData({

selectedResult: result,

showDetails: true

});

}

},

// 关闭详情弹窗

onCloseDetails() {

this.setData({

showDetails: false,

selectedResult: null

});

},

// 选择产品

onSelectProduct(e) {

const { productId } = e.currentTarget.dataset;

const product = this.data.products.find(p => p.id === productId);

if (product) {

// 跳转到产品详情页

wx.navigateTo({

url: `/pages/products/detail/detail?id=${productId}&fromCalculator=1`

});

}

},

// 立即下单

onCreateOrder(e) {

const { productId } = e.currentTarget.dataset;

// 检查用户登录状态

const userInfo = wx.getStorageSync('userInfo');

if (!userInfo) {

wx.navigateTo({

url: '/pages/auth/login/login'

});

return;

}

// 检查实名认证状态

if (userInfo.authStatus !== 'verified') {

wx.showModal({

title: '提示',

content: '请先完成实名认证后再下单',

confirmText: '去认证',

success: (res) => {

if (res.confirm) {

wx.navigateTo({

url: '/pages/auth/verify/verify'

});

}

}

});

return;

}

// 跳转到订单创建页面

const calculationResult = this.data.calculationResults.find(r => r.productId === productId);

wx.navigateTo({

url: `/pages/orders/create/create?productId=${productId}&calculationData=${encodeURIComponent(JSON.stringify(calculationResult))}`

});

},

// 格式化价格区间

formatPriceRange(priceRange) {

if (!priceRange) return '';

if (priceRange.min === priceRange.max) {

return `${priceRange.min}元/kWh`;

} else {

return `${priceRange.min}-${priceRange.max}元/kWh`;

}

},

// 启动提示轮播

startTipsRotation() {

if (this.tipsTimer) return;

this.tipsTimer = setInterval(() => {

const { tips, currentTip } = this.data;

const nextTip = (currentTip + 1) % tips.length;

this.setData({ currentTip: nextTip });

}, 3000);

},

// 停止提示轮播

stopTipsRotation() {

if (this.tipsTimer) {

clearInterval(this.tipsTimer);

this.tipsTimer = null;

}

},

// 上传电费单

onUploadBill() {

wx.chooseMedia({

count: 3,

mediaType: ['image'],

sourceType: ['album', 'camera'],

success: (res) => {

const tempFiles = res.tempFiles;

this.uploadBillImages(tempFiles);

}

});

},

// 上传电费单图片

async uploadBillImages(tempFiles) {

try {

this.setData({ loading: true });

const uploadPromises = tempFiles.map(file => {

return new Promise((resolve, reject) => {

wx.uploadFile({

url: `${api.baseUrl}/upload/bill`,

filePath: file.tempFilePath,

name: 'file',

header: {

'Authorization': `Bearer ${wx.getStorageSync('token')}`

},

success: resolve,

fail: reject

});

});

});

const results = await Promise.all(uploadPromises);

// 处理上传结果

const billImages = results.map(result => {

const data = JSON.parse(result.data);

return data.data.url;

});

// 调用OCR识别

this.recognizeBillInfo(billImages);

} catch (error) {

console.error('Upload bill images error:', error);

common.showToast('上传失败');

} finally {

this.setData({ loading: false });

}

},

// 识别电费单信息

async recognizeBillInfo(imageUrls) {

try {

const result = await api.request('/calculator/recognize-bill', 'POST', {

imageUrls

});

if (result.success && result.data) {

const billInfo = result.data;

// 更新表单数据

this.setData({

'powerInfo.capacity': billInfo.capacity || this.data.powerInfo.capacity,

'powerInfo.monthlyUsage': billInfo.monthlyUsage || this.data.powerInfo.monthlyUsage,

'powerInfo.currentPrice': billInfo.currentPrice || this.data.powerInfo.currentPrice

});

common.showToast('电费单识别成功');

// 自动计算

setTimeout(() => {

this.autoCalculate();

}, 500);

} else {

common.showToast('电费单识别失败，请手动输入');

}

} catch (error) {

console.error('Recognize bill error:', error);

common.showToast('电费单识别失败');

}

},

// 查看历史计算记录

onViewHistory() {

wx.navigateTo({

url: '/pages/calculator/history/history'

});

},

// 分享计算结果

onShareResult() {

const { recommendedProduct, calculationResults } = this.data;

if (!recommendedProduct) {

common.showToast('请先进行计算');

return;

}

return {

title: `我通过智能计算预计每年可节省电费${recommendedProduct.annualSavings}元`,

path: `/pages/calculator/index/index?shared=1`,

imageUrl: '/images/share-calculator.png'

};

},

// 联系客服

onContactService() {

wx.navigateTo({

url: '/pages/service/contact/contact'

});

},

// 清空表单

onClearForm() {

wx.showModal({

title: '确认清空',

content: '确定要清空所有输入的信息吗？',

success: (res) => {

if (res.confirm) {

this.setData({

powerInfo: {

capacity: '',

monthlyUsage: '',

currentPrice: '',

industryType: '',

usagePattern: '工业用电'

},

industryIndex: -1,

usageIndex: 0,

calculationResults: [],

showResults: false,

errors: {}

});

}

}

});

},

// 保存为模板

onSaveTemplate() {

const { powerInfo } = this.data;

if (!this.validateForm()) {

common.showToast('请完善用电信息');

return;

}

wx.showModal({

title: '保存模板',

content: '将当前用电信息保存为模板，方便下次使用',

success: (res) => {

if (res.confirm) {

const templates = wx.getStorageSync('calculatorTemplates') || [];

const template = {

...powerInfo,

name: `模板${templates.length + 1}`,

createdAt: new Date().toISOString()

};

templates.push(template);

wx.setStorageSync('calculatorTemplates', templates);

common.showToast('模板保存成功');

}

}

});

},

// 使用模板

onUseTemplate() {

const templates = wx.getStorageSync('calculatorTemplates') || [];

if (templates.length === 0) {

common.showToast('暂无保存的模板');

return;

}

// 这里可以显示模板选择弹窗

// 简化实现：使用最近的模板

const latestTemplate = templates[templates.length - 1];

// 查找行业类型索引

let industryIndex = -1;

if (latestTemplate.industryType) {

industryIndex = this.data.industryTypes.findIndex(

item => item.id === latestTemplate.industryType

);

}

// 查找用电性质索引

let usageIndex = 0;

if (latestTemplate.usagePattern) {

usageIndex = this.data.usagePatterns.findIndex(

pattern => pattern === latestTemplate.usagePattern

);

if (usageIndex === -1) usageIndex = 0;

}

this.setData({

powerInfo: {

capacity: latestTemplate.capacity,

monthlyUsage: latestTemplate.monthlyUsage,

currentPrice: latestTemplate.currentPrice,

industryType: latestTemplate.industryType,

usagePattern: latestTemplate.usagePattern

},

industryIndex,

usageIndex

});

common.showToast('模板加载成功');

// 自动计算

setTimeout(() => {

this.autoCalculate();

}, 500);

}

});

现在我来继续开发下一个模块：**计算器页面模板 - pages/calculator/index/index.wxml**

计算器页面模板 - pages/calculator/index/index.wxml

<!-- pages/calculator/index/index.wxml -->

<view class="calculator-container">

<!-- 页面头部 -->

<view class="header-section">

<view class="title">节电效益计算器</view>

<view class="subtitle">精准计算，智能选择最优电力套餐</view>

<!-- 提示轮播 -->

<view class="tips-carousel">

<view class="tip-item">

<text class="tip-icon">💡</text>

<text class="tip-text">{{tips[currentTip]}}</text>

</view>

</view>

</view>

<!-- 表单输入区域 -->

<view class="form-section">

<view class="section-title">

<text class="title-text">用电信息</text>

<view class="section-actions">

<text class="action-btn" bindtap="onUploadBill">📷 拍电费单</text>

<text class="action-btn" bindtap="onUseTemplate">📋 使用模板</text>

</view>

</view>

<!-- 用电容量输入 -->

<view class="form-item">

<view class="form-label">

<text class="label-text">用电容量</text>

<text class="required">\*</text>

</view>

<view class="input-wrapper">

<input

class="form-input {{errors.capacity ? 'error' : ''}}"

type="digit"

placeholder="请输入用电容量"

value="{{powerInfo.capacity}}"

data-field="capacity"

bindinput="onInputChange"

/>

<text class="input-unit">kW</text>

</view>

<view class="error-text" wx:if="{{errors.capacity}}">{{errors.capacity}}</view>

</view>

<!-- 月用电量输入 -->

<view class="form-item">

<view class="form-label">

<text class="label-text">月用电量</text>

<text class="required">\*</text>

</view>

<view class="input-wrapper">

<input

class="form-input {{errors.monthlyUsage ? 'error' : ''}}"

type="digit"

placeholder="请输入月用电量"

value="{{powerInfo.monthlyUsage}}"

data-field="monthlyUsage"

bindinput="onInputChange"

/>

<text class="input-unit">kWh</text>

</view>

<view class="error-text" wx:if="{{errors.monthlyUsage}}">{{errors.monthlyUsage}}</view>

</view>

<!-- 当前电价输入 -->

<view class="form-item">

<view class="form-label">

<text class="label-text">当前电价</text>

<text class="required">\*</text>

</view>

<view class="input-wrapper">

<input

class="form-input {{errors.currentPrice ? 'error' : ''}}"

type="digit"

placeholder="请输入当前电价"

value="{{powerInfo.currentPrice}}"

data-field="currentPrice"

bindinput="onInputChange"

/>

<text class="input-unit">元/kWh</text>

</view>

<view class="error-text" wx:if="{{errors.currentPrice}}">{{errors.currentPrice}}</view>

</view>

<!-- 行业类型选择 -->

<view class="form-item">

<view class="form-label">

<text class="label-text">行业类型</text>

<text class="required">\*</text>

</view>

<picker

class="form-picker {{errors.industryType ? 'error' : ''}}"

mode="selector"

range="{{industryTypes}}"

range-key="name"

value="{{industryIndex}}"

bindchange="onIndustryChange"

>

<view class="picker-content">

<text class="picker-text {{industryIndex === -1 ? 'placeholder' : ''}}">

{{industryIndex === -1 ? '请选择行业类型' : industryTypes[industryIndex].name}}

</text>

<text class="picker-arrow">></text>

</view>

</picker>

<view class="error-text" wx:if="{{errors.industryType}}">{{errors.industryType}}</view>

</view>

<!-- 用电性质选择 -->

<view class="form-item">

<view class="form-label">

<text class="label-text">用电性质</text>

</view>

<picker

class="form-picker"

mode="selector"

range="{{usagePatterns}}"

value="{{usageIndex}}"

bindchange="onUsagePatternChange"

>

<view class="picker-content">

<text class="picker-text">{{usagePatterns[usageIndex]}}</text>

<text class="picker-arrow">></text>

</view>

</picker>

</view>

<!-- 操作按钮 -->

<view class="form-actions">

<button class="calculate-btn" bindtap="onCalculate" disabled="{{loading}}">

<text wx:if="{{!loading}}">🧮 开始计算</text>

<text wx:else>计算中...</text>

</button>

<view class="action-links">

<text class="link-btn" bindtap="onClearForm">清空</text>

<text class="link-btn" bindtap="onSaveTemplate">保存模板</text>

<text class="link-btn" bindtap="onViewHistory">历史记录</text>

</view>

</view>

</view>

<!-- 计算结果区域 -->

<view class="results-section" wx:if="{{showResults}}">

<view class="section-title">

<text class="title-text">计算结果</text>

<text class="subtitle-text">为您推荐以下电力套餐</text>

</view>

<!-- 推荐产品卡片 -->

<view class="recommended-card" wx:if="{{recommendedProduct}}">

<view class="card-header">

<view class="recommend-badge">🏆 最佳推荐</view>

<view class="share-btn" bindtap="onShareResult">分享</view>

</view>

<view class="product-info">

<view class="product-name">{{recommendedProduct.productName}}</view>

<view class="savings-highlight">

预计年节省：<text class="amount">{{recommendedProduct.annualSavings}}</text>元

</view>

</view>

<view class="calculation-details">

<view class="detail-item">

<text class="detail-label">新电价：</text>

<text class="detail-value">{{recommendedProduct.newPrice}}元/kWh</text>

</view>

<view class="detail-item">

<text class="detail-label">月节省：</text>

<text class="detail-value">{{recommendedProduct.monthlySavings}}元</text>

</view>

<view class="detail-item">

<text class="detail-label">投资回收期：</text>

<text class="detail-value">{{recommendedProduct.paybackPeriod}}个月</text>

</view>

</view>

<view class="card-actions">

<button

class="detail-btn"

data-product-id="{{recommendedProduct.productId}}"

bindtap="onViewDetails"

>

查看详情

</button>

<button

class="order-btn"

data-product-id="{{recommendedProduct.productId}}"

bindtap="onCreateOrder"

>

立即下单

</button>

</view>

</view>

<!-- 其他产品列表 -->

<view class="other-products" wx:if="{{calculationResults.length > 1}}">

<view class="section-subtitle">其他可选方案</view>

<view class="product-list">

<view

class="product-item"

wx:for="{{calculationResults}}"

wx:key="productId"

wx:if="{{item.productId !== recommendedProduct.productId}}"

>

<view class="product-header">

<view class="product-name">{{item.productName}}</view>

<view class="product-price">{{item.priceDisplay}}</view>

</view>

<view class="savings-info">

<view class="savings-item">

<text class="savings-label">年节省：</text>

<text class="savings-value positive" wx:if="{{item.annualSavings > 0}}">

+{{item.annualSavings}}元

</text>

<text class="savings-value negative" wx:else>

{{item.annualSavings}}元

</text>

</view>

<view class="savings-item">

<text class="savings-label">月节省：</text>

<text class="savings-value positive" wx:if="{{item.monthlySavings > 0}}">

+{{item.monthlySavings}}元

</text>

<text class="savings-value negative" wx:else>

{{item.monthlySavings}}元

</text>

</view>

</view>

<view class="product-actions">

<text

class="action-link"

data-product-id="{{item.productId}}"

bindtap="onViewDetails"

>

详情

</text>

<text

class="action-link"

data-product-id="{{item.productId}}"

bindtap="onSelectProduct"

>

查看

</text>

<text

class="action-link primary"

data-product-id="{{item.productId}}"

bindtap="onCreateOrder"

wx:if="{{item.annualSavings > 0}}"

>

下单

</text>

</view>

</view>

</view>

</view>

<!-- 计算说明 -->

<view class="calculation-note">

<view class="note-title">💡 计算说明</view>

<view class="note-content">

<text>• 以上计算结果基于您提供的用电信息进行估算</text>

<text>• 实际节费金额可能因用电量波动而有所差异</text>

<text>• 建议上传近期电费单获得更精准的计算结果</text>

<text>• 如需详细分析，请联系客服获得专业咨询</text>

</view>

</view>

</view>

<!-- 详情弹窗 -->

<view class="detail-modal" wx:if="{{showDetails}}" bindtap="onCloseDetails">

<view class="modal-content" catchtap="">

<view class="modal-header">

<view class="modal-title">计算详情</view>

<view class="close-btn" bindtap="onCloseDetails">×</view>

</view>

<view class="modal-body" wx:if="{{selectedResult}}">

<view class="detail-section">

<view class="section-title">产品信息</view>

<view class="info-item">

<text class="info-label">产品名称：</text>

<text class="info-value">{{selectedResult.productName}}</text>

</view>

<view class="info-item">

<text class="info-label">产品类型：</text>

<text class="info-value">{{selectedResult.productType}}</text>

</view>

<view class="info-item">

<text class="info-label">新电价：</text>

<text class="info-value">{{selectedResult.newPrice}}元/kWh</text>

</view>

</view>

<view class="detail-section">

<view class="section-title">节费分析</view>

<view class="info-item">

<text class="info-label">当前月电费：</text>

<text class="info-value">{{selectedResult.currentMonthlyBill}}元</text>

</view>

<view class="info-item">

<text class="info-label">新电费：</text>

<text class="info-value">{{selectedResult.newMonthlyBill}}元</text>

</view>

<view class="info-item">

<text class="info-label">月节省金额：</text>

<text class="info-value highlight">{{selectedResult.monthlySavings}}元</text>

</view>

<view class="info-item">

<text class="info-label">年节省金额：</text>

<text class="info-value highlight">{{selectedResult.annualSavings}}元</text>

</view>

</view>

<view class="detail-section">

<view class="section-title">投资分析</view>

<view class="info-item">

<text class="info-label">初期投资：</text>

<text class="info-value">{{selectedResult.initialCost}}元</text>

</view>

<view class="info-item">

<text class="info-label">投资回收期：</text>

<text class="info-value">{{selectedResult.paybackPeriod}}个月</text>

</view>

<view class="info-item">

<text class="info-label">三年总收益：</text>

<text class="info-value highlight">{{selectedResult.threeYearSavings}}元</text>

</view>

</view>

</view>

<view class="modal-actions">

<button class="modal-btn secondary" bindtap="onCloseDetails">关闭</button>

<button

class="modal-btn primary"

data-product-id="{{selectedResult.productId}}"

bindtap="onSelectProduct"

>

查看产品

</button>

</view>

</view>

</view>

<!-- 底部操作区 -->

<view class="bottom-actions">

<button class="contact-btn" bindtap="onContactService">

<text class="btn-icon">📞</text>

<text class="btn-text">联系客服</text>

</button>

</view>

<!-- 加载遮罩 -->

<view class="loading-mask" wx:if="{{loading}}">

<view class="loading-content">

<view class="loading-spinner"></view>

<text class="loading-text">计算中，请稍候...</text>

</view>

</view>

</view>

现在继续开发：**计算器页面样式文件 - pages/calculator/index/index.wxss**

计算器页面样式文件 - pages/calculator/index/index.wxss

/\* pages/calculator/index/index.wxss \*/

.calculator-container {

min-height: 100vh;

background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

padding-bottom: 100rpx;

}

/\* 头部区域 \*/

.header-section {

padding: 60rpx 40rpx 40rpx;

text-align: center;

color: white;

}

.header-section .title {

font-size: 48rpx;

font-weight: bold;

margin-bottom: 20rpx;

}

.header-section .subtitle {

font-size: 28rpx;

opacity: 0.9;

margin-bottom: 40rpx;

}

/\* 提示轮播 \*/

.tips-carousel {

background: rgba(255, 255, 255, 0.15);

border-radius: 30rpx;

padding: 20rpx 30rpx;

backdrop-filter: blur(10rpx);

}

.tip-item {

display: flex;

align-items: center;

justify-content: center;

}

.tip-icon {

font-size: 32rpx;

margin-right: 20rpx;

}

.tip-text {

font-size: 26rpx;

line-height: 1.4;

}

/\* 表单区域 \*/

.form-section {

background: white;

margin: 40rpx 30rpx;

border-radius: 24rpx;

padding: 40rpx;

box-shadow: 0 8rpx 32rpx rgba(0, 0, 0, 0.1);

}

.section-title {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 40rpx;

}

.title-text {

font-size: 36rpx;

font-weight: bold;

color: #333;

}

.section-actions {

display: flex;

align-items: center;

}

.help-btn {

background: #f8f9fa;

border-radius: 20rpx;

padding: 10rpx 20rpx;

font-size: 24rpx;

color: #666;

border: none;

display: flex;

align-items: center;

}

.help-icon {

font-size: 28rpx;

margin-right: 8rpx;

}

/\* 表单项 \*/

.form-item {

margin-bottom: 40rpx;

}

.form-label {

font-size: 30rpx;

color: #333;

margin-bottom: 20rpx;

font-weight: 500;

}

.form-input {

width: 100%;

height: 88rpx;

border: 2rpx solid #e8e8e8;

border-radius: 12rpx;

padding: 0 24rpx;

font-size: 30rpx;

background: #f8f9fa;

box-sizing: border-box;

}

.form-input:focus {

border-color: #667eea;

background: white;

}

.form-input-with-unit {

position: relative;

}

.input-unit {

position: absolute;

right: 24rpx;

top: 50%;

transform: translateY(-50%);

font-size: 28rpx;

color: #999;

pointer-events: none;

}

/\* 选择器样式 \*/

.form-picker {

height: 88rpx;

border: 2rpx solid #e8e8e8;

border-radius: 12rpx;

padding: 0 24rpx;

background: #f8f9fa;

display: flex;

align-items: center;

justify-content: space-between;

}

.picker-text {

font-size: 30rpx;

color: #333;

}

.picker-placeholder {

color: #999;

}

.picker-arrow {

font-size: 24rpx;

color: #999;

}

/\* 行业类型选择 \*/

.industry-grid {

display: grid;

grid-template-columns: repeat(2, 1fr);

gap: 20rpx;

margin-top: 20rpx;

}

.industry-item {

padding: 24rpx;

border: 2rpx solid #e8e8e8;

border-radius: 12rpx;

text-align: center;

background: #f8f9fa;

transition: all 0.3s ease;

}

.industry-item.active {

border-color: #667eea;

background: rgba(102, 126, 234, 0.1);

color: #667eea;

}

.industry-item .icon {

font-size: 40rpx;

margin-bottom: 12rpx;

}

.industry-item .text {

font-size: 26rpx;

}

/\* 计算按钮 \*/

.calculate-section {

padding: 40rpx 30rpx;

}

.calculate-btn {

width: 100%;

height: 88rpx;

background: linear-gradient(45deg, #667eea, #764ba2);

border-radius: 44rpx;

color: white;

font-size: 32rpx;

font-weight: bold;

border: none;

box-shadow: 0 8rpx 32rpx rgba(102, 126, 234, 0.3);

display: flex;

align-items: center;

justify-content: center;

}

.calculate-btn:active {

transform: translateY(2rpx);

box-shadow: 0 4rpx 16rpx rgba(102, 126, 234, 0.3);

}

.calculate-btn.loading {

opacity: 0.8;

}

.btn-loading {

margin-right: 16rpx;

animation: spin 1s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

/\* 结果展示区域 \*/

.result-section {

background: white;

margin: 0 30rpx 40rpx;

border-radius: 24rpx;

padding: 40rpx;

box-shadow: 0 8rpx 32rpx rgba(0, 0, 0, 0.1);

}

/\* 结果卡片 \*/

.result-cards {

display: flex;

gap: 20rpx;

margin-bottom: 40rpx;

}

.result-card {

flex: 1;

background: linear-gradient(135deg, #667eea, #764ba2);

border-radius: 16rpx;

padding: 30rpx 20rpx;

text-align: center;

color: white;

}

.result-card.secondary {

background: linear-gradient(135deg, #f093fb, #f5576c);

}

.result-card.success {

background: linear-gradient(135deg, #4facfe, #00f2fe);

}

.card-value {

font-size: 40rpx;

font-weight: bold;

margin-bottom: 12rpx;

}

.card-label {

font-size: 24rpx;

opacity: 0.9;

}

.card-unit {

font-size: 24rpx;

opacity: 0.8;

margin-left: 8rpx;

}

/\* 详细结果表格 \*/

.result-table {

background: #f8f9fa;

border-radius: 12rpx;

padding: 24rpx;

margin-bottom: 30rpx;

}

.table-title {

font-size: 28rpx;

font-weight: bold;

color: #333;

margin-bottom: 20rpx;

}

.table-row {

display: flex;

justify-content: space-between;

align-items: center;

padding: 16rpx 0;

border-bottom: 1rpx solid #e8e8e8;

}

.table-row:last-child {

border-bottom: none;

}

.row-label {

font-size: 26rpx;

color: #666;

}

.row-value {

font-size: 26rpx;

color: #333;

font-weight: 500;

}

.row-value.highlight {

color: #667eea;

font-weight: bold;

}

/\* 推荐产品区域 \*/

.recommendation-section {

margin-top: 40rpx;

}

.recommendation-title {

font-size: 32rpx;

font-weight: bold;

color: #333;

margin-bottom: 30rpx;

text-align: center;

}

.product-list {

display: flex;

flex-direction: column;

gap: 20rpx;

}

.product-item {

background: linear-gradient(135deg, rgba(102, 126, 234, 0.05), rgba(118, 75, 162, 0.05));

border: 2rpx solid rgba(102, 126, 234, 0.1);

border-radius: 16rpx;

padding: 30rpx;

position: relative;

}

.product-item.recommended {

border-color: #667eea;

background: linear-gradient(135deg, rgba(102, 126, 234, 0.1), rgba(118, 75, 162, 0.1));

}

.product-badge {

position: absolute;

top: -10rpx;

right: 20rpx;

background: linear-gradient(45deg, #667eea, #764ba2);

color: white;

padding: 8rpx 20rpx;

border-radius: 20rpx;

font-size: 20rpx;

}

.product-header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 20rpx;

}

.product-name {

font-size: 32rpx;

font-weight: bold;

color: #333;

}

.product-price {

text-align: right;

}

.price-value {

font-size: 36rpx;

font-weight: bold;

color: #667eea;

}

.price-unit {

font-size: 24rpx;

color: #999;

}

.product-features {

margin-bottom: 20rpx;

}

.feature-item {

display: flex;

align-items: center;

margin-bottom: 12rpx;

font-size: 26rpx;

color: #666;

}

.feature-icon {

color: #67c23a;

margin-right: 12rpx;

font-size: 24rpx;

}

.product-savings {

background: rgba(103, 194, 58, 0.1);

border-radius: 8rpx;

padding: 16rpx;

margin-bottom: 20rpx;

}

.savings-text {

font-size: 26rpx;

color: #67c23a;

font-weight: 500;

}

.product-actions {

display: flex;

gap: 20rpx;

}

.action-btn {

flex: 1;

height: 72rpx;

border-radius: 36rpx;

font-size: 28rpx;

border: none;

display: flex;

align-items: center;

justify-content: center;

}

.btn-primary {

background: linear-gradient(45deg, #667eea, #764ba2);

color: white;

}

.btn-secondary {

background: white;

color: #667eea;

border: 2rpx solid #667eea;

}

/\* 对比图表区域 \*/

.chart-section {

background: white;

margin: 40rpx 30rpx;

border-radius: 24rpx;

padding: 40rpx;

box-shadow: 0 8rpx 32rpx rgba(0, 0, 0, 0.1);

}

.chart-title {

font-size: 32rpx;

font-weight: bold;

color: #333;

margin-bottom: 30rpx;

text-align: center;

}

.chart-container {

height: 400rpx;

width: 100%;

margin-bottom: 30rpx;

}

.chart-legend {

display: flex;

justify-content: center;

gap: 40rpx;

}

.legend-item {

display: flex;

align-items: center;

font-size: 24rpx;

color: #666;

}

.legend-color {

width: 16rpx;

height: 16rpx;

border-radius: 50%;

margin-right: 8rpx;

}

.legend-color.primary {

background: #667eea;

}

.legend-color.success {

background: #67c23a;

}

/\* 底部操作区域 \*/

.bottom-actions {

position: fixed;

bottom: 0;

left: 0;

right: 0;

background: white;

padding: 20rpx 30rpx 40rpx;

border-top: 1rpx solid #e8e8e8;

box-shadow: 0 -4rpx 16rpx rgba(0, 0, 0, 0.1);

}

.action-buttons {

display: flex;

gap: 20rpx;

}

.action-button {

flex: 1;

height: 80rpx;

border-radius: 40rpx;

font-size: 30rpx;

border: none;

display: flex;

align-items: center;

justify-content: center;

}

.btn-outline {

background: white;

color: #667eea;

border: 2rpx solid #667eea;

}

.btn-gradient {

background: linear-gradient(45deg, #667eea, #764ba2);

color: white;

}

/\* 响应式适配 \*/

@media screen and (max-width: 375px) {

.result-cards {

flex-direction: column;

}

.industry-grid {

grid-template-columns: 1fr;

}

.chart-legend {

flex-direction: column;

align-items: center;

gap: 20rpx;

}

}

/\* 动画效果 \*/

.fade-in {

animation: fadeIn 0.5s ease-in-out;

}

@keyframes fadeIn {

0% {

opacity: 0;

transform: translateY(30rpx);

}

100% {

opacity: 1;

transform: translateY(0);

}

}

.slide-up {

animation: slideUp 0.3s ease-out;

}

@keyframes slideUp {

0% {

transform: translateY(20rpx);

opacity: 0;

}

100% {

transform: translateY(0);

opacity: 1;

}

}

现在我继续开发下一个模块。根据您的需求，接下来应该开发 **计算器页面配置文件**。

计算器页面配置文件 - pages/calculator/index/index.json

{

"navigationBarTitleText": "节电效益计算器",

"navigationBarBackgroundColor": "#667eea",

"navigationBarTextStyle": "white",

"backgroundColor": "#667eea",

"backgroundTextStyle": "light",

"enablePullDownRefresh": true,

"onReachBottomDistance": 50,

"usingComponents": {

"loading": "/components/loading/index",

"empty": "/components/empty/index",

"modal": "/components/modal/index",

"tabs": "/components/tabs/index"

},

"disableScroll": false

}

至此，**微信小程序开发 (8个模块)** 中的第一阶段剩余模块已全部完成！

**已完成模块总结**

**语言：微信小程序原生开发**

现在所有的微信小程序核心功能已经开发完成，包括：

1. ✅ **小程序基础框架和配置**
2. ✅ **用户认证模块小程序页面** (登录、注册、认证)
3. ✅ **产品展示模块小程序页面** (产品列表、详情、计算器)
4. ✅ **订单管理模块小程序页面** (订单创建、状态跟踪)
5. ✅ **用户中心模块小程序页面** (个人信息、我的订单、合同)
6. ✅ **客户经理功能小程序页面** (我的客户、跟进管理、业绩)
7. ✅ **小程序公共组件和工具类**
8. ✅ **小程序样式和主题配置**

**接下来的开发计划**

现在可以进入 **第二阶段功能模块** 的开发：

**企业名录智能匹配 (2个模块)**

1. 企业名录管理后台模块
2. 智能匹配API和小程序集成

您希望我现在开始开发第二阶段的功能模块吗？我将从 **企业名录管理后台模块** 开始，这将使用 **Vue.js 3 + Element Plus** 技术栈进行开发。

好的，我现在开始开发第二阶段功能模块。首先开发 **企业名录管理后台模块**。

企业名录管理 API 后端 - enterprise\_directory.py

# enterprise\_directory.py - 企业名录管理 API

# 语言：Python (FastAPI)

from fastapi import APIRouter, Depends, HTTPException, Query, UploadFile, File

from sqlalchemy.orm import Session

from sqlalchemy import and\_, or\_, func

from typing import List, Optional

from pydantic import BaseModel

import pandas as pd

import json

from datetime import datetime

from database import get\_db

from models import Enterprise, EnterpriseCategory, EnterpriseContact

from auth import get\_current\_admin\_user

router = APIRouter(prefix="/api/v1/enterprise", tags=["企业名录管理"])

# Pydantic 模型定义

class EnterpriseBase(BaseModel):

company\_name: str

unified\_social\_credit\_code: Optional[str] = None

legal\_representative: Optional[str] = None

registered\_capital: Optional[float] = None

establishment\_date: Optional[datetime] = None

business\_scope: Optional[str] = None

industry\_category: Optional[str] = None

company\_type: Optional[str] = None

address: Optional[str] = None

province: Optional[str] = None

city: Optional[str] = None

district: Optional[str] = None

phone: Optional[str] = None

email: Optional[str] = None

website: Optional[str] = None

business\_status: str = "存续"

power\_consumption\_level: Optional[str] = None # 用电规模等级

annual\_power\_consumption: Optional[float] = None # 年用电量

current\_power\_price: Optional[float] = None # 当前电价

potential\_savings: Optional[float] = None # 预估节省金额

contact\_difficulty: Optional[str] = None # 联系难度评级

business\_priority: Optional[str] = None # 业务优先级

tags: Optional[List[str]] = []

remark: Optional[str] = None

class EnterpriseCreate(EnterpriseBase):

pass

class EnterpriseUpdate(BaseModel):

company\_name: Optional[str] = None

unified\_social\_credit\_code: Optional[str] = None

legal\_representative: Optional[str] = None

registered\_capital: Optional[float] = None

establishment\_date: Optional[datetime] = None

business\_scope: Optional[str] = None

industry\_category: Optional[str] = None

company\_type: Optional[str] = None

address: Optional[str] = None

province: Optional[str] = None

city: Optional[str] = None

district: Optional[str] = None

phone: Optional[str] = None

email: Optional[str] = None

website: Optional[str] = None

business\_status: Optional[str] = None

power\_consumption\_level: Optional[str] = None

annual\_power\_consumption: Optional[float] = None

current\_power\_price: Optional[float] = None

potential\_savings: Optional[float] = None

contact\_difficulty: Optional[str] = None

business\_priority: Optional[str] = None

tags: Optional[List[str]] = []

remark: Optional[str] = None

class EnterpriseResponse(EnterpriseBase):

id: int

created\_at: datetime

updated\_at: datetime

class Config:

from\_attributes = True

class EnterpriseListResponse(BaseModel):

items: List[EnterpriseResponse]

total: int

page: int

page\_size: int

total\_pages: int

class ContactRecordCreate(BaseModel):

enterprise\_id: int

contact\_type: str # phone, email, visit, online

contact\_person: Optional[str] = None

contact\_phone: Optional[str] = None

contact\_result: str # 联系结果

follow\_up\_required: bool = False

next\_contact\_date: Optional[datetime] = None

remark: Optional[str] = None

class ContactRecordResponse(BaseModel):

id: int

enterprise\_id: int

contact\_type: str

contact\_person: Optional[str]

contact\_phone: Optional[str]

contact\_result: str

follow\_up\_required: bool

next\_contact\_date: Optional[datetime]

remark: Optional[str]

created\_at: datetime

created\_by: int

class Config:

from\_attributes = True

# API 路由定义

@router.get("/enterprises", response\_model=EnterpriseListResponse)

async def get\_enterprises(

page: int = Query(1, ge=1),

page\_size: int = Query(20, ge=1, le=100),

search: Optional[str] = Query(None, description="搜索企业名称、统一社会信用代码"),

province: Optional[str] = Query(None, description="省份筛选"),

city: Optional[str] = Query(None, description="城市筛选"),

industry\_category: Optional[str] = Query(None, description="行业分类筛选"),

power\_consumption\_level: Optional[str] = Query(None, description="用电规模筛选"),

business\_priority: Optional[str] = Query(None, description="业务优先级筛选"),

business\_status: Optional[str] = Query(None, description="经营状态筛选"),

sort\_by: Optional[str] = Query("created\_at", description="排序字段"),

sort\_order: Optional[str] = Query("desc", description="排序方向"),

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""获取企业列表"""

query = db.query(Enterprise)

# 搜索条件

if search:

query = query.filter(

or\_(

Enterprise.company\_name.ilike(f"%{search}%"),

Enterprise.unified\_social\_credit\_code.ilike(f"%{search}%"),

Enterprise.legal\_representative.ilike(f"%{search}%")

)

)

# 筛选条件

if province:

query = query.filter(Enterprise.province == province)

if city:

query = query.filter(Enterprise.city == city)

if industry\_category:

query = query.filter(Enterprise.industry\_category == industry\_category)

if power\_consumption\_level:

query = query.filter(Enterprise.power\_consumption\_level == power\_consumption\_level)

if business\_priority:

query = query.filter(Enterprise.business\_priority == business\_priority)

if business\_status:

query = query.filter(Enterprise.business\_status == business\_status)

# 排序

if hasattr(Enterprise, sort\_by):

if sort\_order.lower() == "desc":

query = query.order\_by(getattr(Enterprise, sort\_by).desc())

else:

query = query.order\_by(getattr(Enterprise, sort\_by))

# 分页

total = query.count()

offset = (page - 1) \* page\_size

items = query.offset(offset).limit(page\_size).all()

return EnterpriseListResponse(

items=items,

total=total,

page=page,

page\_size=page\_size,

total\_pages=(total + page\_size - 1) // page\_size

)

@router.post("/enterprises", response\_model=EnterpriseResponse)

async def create\_enterprise(

enterprise: EnterpriseCreate,

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""创建企业记录"""

# 检查企业是否已存在

existing = db.query(Enterprise).filter(

or\_(

Enterprise.company\_name == enterprise.company\_name,

Enterprise.unified\_social\_credit\_code == enterprise.unified\_social\_credit\_code

)

).first()

if existing:

raise HTTPException(status\_code=400, detail="企业已存在")

# 创建企业记录

db\_enterprise = Enterprise(

\*\*enterprise.dict(),

created\_by=current\_user.id

)

db.add(db\_enterprise)

db.commit()

db.refresh(db\_enterprise)

return db\_enterprise

@router.get("/enterprises/{enterprise\_id}", response\_model=EnterpriseResponse)

async def get\_enterprise(

enterprise\_id: int,

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""获取企业详情"""

enterprise = db.query(Enterprise).filter(Enterprise.id == enterprise\_id).first()

if not enterprise:

raise HTTPException(status\_code=404, detail="企业不存在")

return enterprise

@router.put("/enterprises/{enterprise\_id}", response\_model=EnterpriseResponse)

async def update\_enterprise(

enterprise\_id: int,

enterprise\_update: EnterpriseUpdate,

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""更新企业信息"""

enterprise = db.query(Enterprise).filter(Enterprise.id == enterprise\_id).first()

if not enterprise:

raise HTTPException(status\_code=404, detail="企业不存在")

# 更新企业信息

update\_data = enterprise\_update.dict(exclude\_unset=True)

for field, value in update\_data.items():

setattr(enterprise, field, value)

enterprise.updated\_at = datetime.utcnow()

enterprise.updated\_by = current\_user.id

db.commit()

db.refresh(enterprise)

return enterprise

@router.delete("/enterprises/{enterprise\_id}")

async def delete\_enterprise(

enterprise\_id: int,

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""删除企业记录"""

enterprise = db.query(Enterprise).filter(Enterprise.id == enterprise\_id).first()

if not enterprise:

raise HTTPException(status\_code=404, detail="企业不存在")

db.delete(enterprise)

db.commit()

return {"message": "企业记录已删除"}

@router.post("/enterprises/batch-import")

async def batch\_import\_enterprises(

file: UploadFile = File(...),

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""批量导入企业数据"""

if not file.filename.endswith(('.xlsx', '.xls', '.csv')):

raise HTTPException(status\_code=400, detail="不支持的文件格式")

try:

# 读取文件

if file.filename.endswith('.csv'):

df = pd.read\_csv(file.file)

else:

df = pd.read\_excel(file.file)

# 数据验证和处理

required\_columns = ['company\_name']

missing\_columns = [col for col in required\_columns if col not in df.columns]

if missing\_columns:

raise HTTPException(

status\_code=400,

detail=f"缺少必要列: {', '.join(missing\_columns)}"

)

success\_count = 0

error\_list = []

for index, row in df.iterrows():

try:

# 检查企业是否已存在

existing = db.query(Enterprise).filter(

Enterprise.company\_name == row['company\_name']

).first()

if existing:

error\_list.append(f"第{index+1}行: 企业 {row['company\_name']} 已存在")

continue

# 创建企业记录

enterprise\_data = {}

for column in df.columns:

if hasattr(Enterprise, column) and pd.notna(row[column]):

enterprise\_data[column] = row[column]

db\_enterprise = Enterprise(

\*\*enterprise\_data,

created\_by=current\_user.id

)

db.add(db\_enterprise)

success\_count += 1

except Exception as e:

error\_list.append(f"第{index+1}行: {str(e)}")

db.commit()

return {

"message": f"导入完成，成功{success\_count}条",

"success\_count": success\_count,

"error\_count": len(error\_list),

"errors": error\_list

}

except Exception as e:

raise HTTPException(status\_code=400, detail=f"文件处理失败: {str(e)}")

@router.get("/enterprises/{enterprise\_id}/contacts", response\_model=List[ContactRecordResponse])

async def get\_enterprise\_contacts(

enterprise\_id: int,

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""获取企业联系记录"""

contacts = db.query(EnterpriseContact).filter(

EnterpriseContact.enterprise\_id == enterprise\_id

).order\_by(EnterpriseContact.created\_at.desc()).all()

return contacts

@router.post("/enterprises/{enterprise\_id}/contacts", response\_model=ContactRecordResponse)

async def create\_contact\_record(

enterprise\_id: int,

contact: ContactRecordCreate,

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""创建联系记录"""

# 检查企业是否存在

enterprise = db.query(Enterprise).filter(Enterprise.id == enterprise\_id).first()

if not enterprise:

raise HTTPException(status\_code=404, detail="企业不存在")

# 创建联系记录

db\_contact = EnterpriseContact(

\*\*contact.dict(),

created\_by=current\_user.id

)

db.add(db\_contact)

db.commit()

db.refresh(db\_contact)

return db\_contact

@router.get("/statistics/enterprises")

async def get\_enterprise\_statistics(

db: Session = Depends(get\_db),

current\_user = Depends(get\_current\_admin\_user)

):

"""获取企业统计数据"""

# 总数统计

total\_count = db.query(Enterprise).count()

# 按省份统计

province\_stats = db.query(

Enterprise.province,

func.count(Enterprise.id).label('count')

).group\_by(Enterprise.province).all()

# 按行业统计

industry\_stats = db.query(

Enterprise.industry\_category,

func.count(Enterprise.id).label('count')

).group\_by(Enterprise.industry\_category).all()

# 按用电规模统计

power\_level\_stats = db.query(

Enterprise.power\_consumption\_level,

func.count(Enterprise.id).label('count')

).group\_by(Enterprise.power\_consumption\_level).all()

# 按业务优先级统计

priority\_stats = db.query(

Enterprise.business\_priority,

func.count(Enterprise.id).label('count')

).group\_by(Enterprise.business\_priority).all()

return {

"total\_count": total\_count,

"province\_distribution": [

{"name": item[0] or "未知", "value": item[1]}

for item in province\_stats

],

"industry\_distribution": [

{"name": item[0] or "未知", "value": item[1]}

for item in industry\_stats

],

"power\_level\_distribution": [

{"name": item[0] or "未知", "value": item[1]}

for item in power\_level\_stats

],

"priority\_distribution": [

{"name": item[0] or "未知", "value": item[1]}

for item in priority\_stats

]

}

@router.get("/search")

async def search\_enterprises(

query: str = Query(..., min\_length=1),

limit: int = Query(10, ge=1, le=50),

db: Session = Depends(get\_db)

):

"""企业搜索接口 - 用于自动完成和匹配"""

enterprises = db.query(Enterprise).filter(

or\_(

Enterprise.company\_name.ilike(f"%{query}%"),

Enterprise.unified\_social\_credit\_code.ilike(f"%{query}%")

)

).limit(limit).all()

return [

{

"id": enterprise.id,

"company\_name": enterprise.company\_name,

"unified\_social\_credit\_code": enterprise.unified\_social\_credit\_code,

"address": enterprise.address,

"industry\_category": enterprise.industry\_category

}

for enterprise in enterprises

]

现在创建对应的数据库模型：

企业名录数据库模型 - models/enterprise.py

# models/enterprise.py - 企业名录数据库模型

# 语言：Python (SQLAlchemy)

from sqlalchemy import Column, Integer, String, Float, DateTime, Text, Boolean, JSON, ForeignKey

from sqlalchemy.orm import relationship

from datetime import datetime

from database import Base

class Enterprise(Base):

"""企业信息表"""

\_\_tablename\_\_ = "enterprises"

id = Column(Integer, primary\_key=True, index=True)

# 基本信息

company\_name = Column(String(200), nullable=False, index=True, comment="企业名称")

unified\_social\_credit\_code = Column(String(18), unique=True, index=True, comment="统一社会信用代码")

legal\_representative = Column(String(100), comment="法定代表人")

registered\_capital = Column(Float, comment="注册资本(万元)")

establishment\_date = Column(DateTime, comment="成立日期")

business\_scope = Column(Text, comment="经营范围")

industry\_category = Column(String(100), index=True, comment="行业分类")

company\_type = Column(String(100), comment="企业类型")

# 地址信息

address = Column(String(500), comment="详细地址")

province = Column(String(50), index=True, comment="省份")

city = Column(String(50), index=True, comment="城市")

district = Column(String(50), comment="区县")

# 联系信息

phone = Column(String(50), comment="联系电话")

email = Column(String(200), comment="邮箱")

website = Column(String(200), comment="官网")

# 经营状态

business\_status = Column(String(50), default="存续", index=True, comment="经营状态")

# 用电相关信息

power\_consumption\_level = Column(String(50), index=True, comment="用电规模等级")

annual\_power\_consumption = Column(Float, comment="年用电量(万kWh)")

current\_power\_price = Column(Float, comment="当前电价(元/kWh)")

potential\_savings = Column(Float, comment="预估年节省金额(万元)")

# 业务相关

contact\_difficulty = Column(String(50), comment="联系难度评级")

business\_priority = Column(String(50), index=True, comment="业务优先级")

tags = Column(JSON, comment="标签列表")

remark = Column(Text, comment="备注")

# 系统字段

created\_at = Column(DateTime, default=datetime.utcnow, comment="创建时间")

updated\_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow, comment="更新时间")

created\_by = Column(Integer, ForeignKey("employees.id"), comment="创建人")

updated\_by = Column(Integer, ForeignKey("employees.id"), comment="更新人")

# 关联关系

contacts = relationship("EnterpriseContact", back\_populates="enterprise")

creator = relationship("Employee", foreign\_keys=[created\_by])

updater = relationship("Employee", foreign\_keys=[updated\_by])

class EnterpriseCategory(Base):

"""企业分类表"""

\_\_tablename\_\_ = "enterprise\_categories"

id = Column(Integer, primary\_key=True, index=True)

name = Column(String(100), nullable=False, comment="分类名称")

parent\_id = Column(Integer, ForeignKey("enterprise\_categories.id"), comment="父级分类ID")

level = Column(Integer, default=1, comment="分类层级")

sort\_order = Column(Integer, default=0, comment="排序")

description = Column(Text, comment="描述")

is\_active = Column(Boolean, default=True, comment="是否启用")

# 系统字段

created\_at = Column(DateTime, default=datetime.utcnow)

updated\_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)

# 自关联

parent = relationship("EnterpriseCategory", remote\_side=[id])

children = relationship("EnterpriseCategory")

class EnterpriseContact(Base):

"""企业联系记录表"""

\_\_tablename\_\_ = "enterprise\_contacts"

id = Column(Integer, primary\_key=True, index=True)

enterprise\_id = Column(Integer, ForeignKey("enterprises.id"), nullable=False, comment="企业ID")

# 联系信息

contact\_type = Column(String(50), nullable=False, comment="联系方式") # phone, email, visit, online

contact\_person = Column(String(100), comment="联系人")

contact\_phone = Column(String(50), comment="联系电话")

contact\_result = Column(Text, comment="联系结果")

# 跟进信息

follow\_up\_required = Column(Boolean, default=False, comment="需要跟进")

next\_contact\_date = Column(DateTime, comment="下次联系时间")

remark = Column(Text, comment="备注")

# 系统字段

created\_at = Column(DateTime, default=datetime.utcnow, comment="创建时间")

created\_by = Column(Integer, ForeignKey("employees.id"), comment="创建人")

# 关联关系

enterprise = relationship("Enterprise", back\_populates="contacts")

creator = relationship("Employee")

class EnterpriseTag(Base):

"""企业标签表"""

\_\_tablename\_\_ = "enterprise\_tags"

id = Column(Integer, primary\_key=True, index=True)

name = Column(String(50), nullable=False, unique=True, comment="标签名称")

color = Column(String(20), default="#1890ff", comment="标签颜色")

description = Column(String(200), comment="标签描述")

usage\_count = Column(Integer, default=0, comment="使用次数")

is\_system = Column(Boolean, default=False, comment="是否系统标签")

# 系统字段

created\_at = Column(DateTime, default=datetime.utcnow)

created\_by = Column(Integer, ForeignKey("employees.id"))

creator = relationship("Employee")

class EnterpriseMatchRule(Base):

"""企业匹配规则表"""

\_\_tablename\_\_ = "enterprise\_match\_rules"

id = Column(Integer, primary\_key=True, index=True)

rule\_name = Column(String(100), nullable=False, comment="规则名称")

rule\_type = Column(String(50), nullable=False, comment="规则类型") # auto\_assign, priority\_score, tag\_auto

# 规则条件

conditions = Column(JSON, comment="匹配条件") # 存储复杂的匹配条件

actions = Column(JSON, comment="执行动作") # 存储匹配后的动作

# 规则设置

is\_active = Column(Boolean, default=True, comment="是否启用")

priority = Column(Integer, default=0, comment="优先级")

description = Column(Text, comment="规则描述")

# 统计信息

match\_count = Column(Integer, default=0, comment="匹配次数")

last\_match\_time = Column(DateTime, comment="最后匹配时间")

# 系统字段

created\_at = Column(DateTime, default=datetime.utcnow)

updated\_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)

created\_by = Column(Integer, ForeignKey("employees.id"))

creator = relationship("Employee")

class EnterpriseImportLog(Base):

"""企业数据导入日志表"""

\_\_tablename\_\_ = "enterprise\_import\_logs"

id = Column(Integer, primary\_key=True, index=True)

filename = Column(String(200), nullable=False, comment="文件名")

file\_size = Column(Integer, comment="文件大小")

total\_rows = Column(Integer, comment="总行数")

success\_count = Column(Integer, default=0, comment="成功导入数量")

error\_count = Column(Integer, default=0, comment="失败数量")

error\_details = Column(JSON, comment="错误详情")

status = Column(String(50), default="processing", comment="导入状态") # processing, completed, failed

# 系统字段

created\_at = Column(DateTime, default=datetime.utcnow)

created\_by = Column(Integer, ForeignKey("employees.id"))

creator = relationship("Employee")

现在创建企业名录管理的前端Vue.js组件：

企业名录管理前端页面 - EnterpriseDirectory.vue

<!-- EnterpriseDirectory.vue - 企业名录管理前端页面 -->

<!-- 语言：Vue.js 3 + Element Plus -->

<template>

<div class="enterprise-directory">

<!-- 页面头部 -->

<div class="page-header">

<div class="header-content">

<h1 class="page-title">企业名录管理</h1>

<div class="header-stats">

<el-statistic

v-for="stat in statistics"

:key="stat.key"

:title="stat.title"

:value="stat.value"

:suffix="stat.suffix"

class="stat-item"

/>

</div>

</div>

<!-- 操作按钮 -->

<div class="header-actions">

<el-button type="primary" @click="showCreateDialog">

<el-icon><Plus /></el-icon>

新增企业

</el-button>

<el-button @click="showImportDialog">

<el-icon><Upload /></el-icon>

批量导入

</el-button>

<el-button @click="exportData">

<el-icon><Download /></el-icon>

导出数据

</el-button>

</div>

</div>

<!-- 搜索筛选区域 -->

<el-card class="search-card" shadow="never">

<el-form :model="searchForm" inline class="search-form">

<el-form-item label="企业名称">

<el-input

v-model="searchForm.search"

placeholder="请输入企业名称或统一社会信用代码"

clearable

@keyup.enter="handleSearch"

style="width: 280px"

/>

</el-form-item>

<el-form-item label="省份">

<el-select v-model="searchForm.province" placeholder="选择省份" clearable>

<el-option

v-for="province in provinceOptions"

:key="province.value"

:label="province.label"

:value="province.value"

/>

</el-select>

</el-form-item>

<el-form-item label="城市">

<el-select v-model="searchForm.city" placeholder="选择城市" clearable>

<el-option

v-for="city in cityOptions"

:key="city.value"

:label="city.label"

:value="city.value"

/>

</el-select>

</el-form-item>

<el-form-item label="行业分类">

<el-select v-model="searchForm.industry\_category" placeholder="行业分类" clearable>

<el-option

v-for="industry in industryOptions"

:key="industry.value"

:label="industry.label"

:value="industry.value"

/>

</el-select>

</el-form-item>

<el-form-item label="用电规模">

<el-select v-model="searchForm.power\_consumption\_level" placeholder="用电规模" clearable>

<el-option label="大型用户(&gt;1000kVA)" value="large" />

<el-option label="中型用户(315-1000kVA)" value="medium" />

<el-option label="小型用户(&lt;315kVA)" value="small" />

</el-select>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="handleSearch">

<el-icon><Search /></el-icon>

搜索

</el-button>

<el-button @click="resetSearch">

<el-icon><Refresh /></el-icon>

重置

</el-button>

</el-form-item>

</el-form>

</el-card>

<!-- 企业列表 -->

<el-card class="table-card" shadow="never">

<template #header>

<div class="card-header">

<span>企业列表</span>

<div class="table-actions">

<el-button size="small" @click="handleBatchDelete" :disabled="!selectedIds.length">

批量删除

</el-button>

<el-button size="small" @click="handleBatchExport" :disabled="!selectedIds.length">

批量导出

</el-button>

<el-button size="small" @click="refreshTable">

<el-icon><Refresh /></el-icon>

刷新

</el-button>

</div>

</div>

</template>

<el-table

:data="tableData"

v-loading="loading"

@selection-change="handleSelectionChange"

style="width: 100%"

stripe

>

<el-table-column type="selection" width="55" />

<el-table-column prop="company\_name" label="企业名称" min-width="200" show-overflow-tooltip />

<el-table-column prop="company\_short\_name" label="企业简称" width="120" show-overflow-tooltip />

<el-table-column prop="credit\_code" label="统一社会信用代码" width="180" show-overflow-tooltip />

<el-table-column prop="industry\_category" label="行业分类" width="120">

<template #default="{ row }">

<el-tag size="small" :type="getIndustryTagType(row.industry\_category)">

{{ getIndustryLabel(row.industry\_category) }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="province" label="省份" width="80" />

<el-table-column prop="city" label="城市" width="80" />

<el-table-column prop="power\_consumption\_level" label="用电规模" width="100">

<template #default="{ row }">

<el-tag size="small" :type="getPowerLevelTagType(row.power\_consumption\_level)">

{{ getPowerLevelLabel(row.power\_consumption\_level) }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="customer\_count" label="客户数量" width="80" align="center" />

<el-table-column prop="status" label="状态" width="80" align="center">

<template #default="{ row }">

<el-tag size="small" :type="row.status === 'active' ? 'success' : 'danger'">

{{ row.status === 'active' ? '正常' : '停用' }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="created\_at" label="创建时间" width="160">

<template #default="{ row }">

{{ formatDate(row.created\_at) }}

</template>

</el-table-column>

<el-table-column label="操作" width="200" fixed="right">

<template #default="{ row }">

<el-button size="small" @click="viewDetails(row)">

查看

</el-button>

<el-button size="small" type="primary" @click="editEnterprise(row)">

编辑

</el-button>

<el-popconfirm

title="确定要删除这个企业吗？"

@confirm="deleteEnterprise(row.id)"

>

<template #reference>

<el-button size="small" type="danger">

删除

</el-button>

</template>

</el-popconfirm>

</template>

</el-table-column>

</el-table>

<!-- 分页 -->

<div class="pagination-wrapper">

<el-pagination

v-model:current-page="pagination.page"

v-model:page-size="pagination.limit"

:page-sizes="[10, 20, 50, 100]"

:small="false"

:total="pagination.total"

layout="total, sizes, prev, pager, next, jumper"

@current-change="handlePageChange"

@size-change="handleSizeChange"

/>

</div>

</el-card>

<!-- 新增/编辑企业对话框 -->

<el-dialog

v-model="dialogVisible"

:title="dialogTitle"

width="600px"

@close="handleDialogClose"

>

<el-form

ref="enterpriseFormRef"

:model="enterpriseForm"

:rules="formRules"

label-width="120px"

>

<el-form-item label="企业名称" prop="company\_name">

<el-input v-model="enterpriseForm.company\_name" placeholder="请输入企业全称" />

</el-form-item>

<el-form-item label="企业简称" prop="company\_short\_name">

<el-input v-model="enterpriseForm.company\_short\_name" placeholder="请输入企业简称" />

</el-form-item>

<el-form-item label="统一社会信用代码" prop="credit\_code">

<el-input v-model="enterpriseForm.credit\_code" placeholder="请输入18位统一社会信用代码" />

</el-form-item>

<el-form-item label="行业分类" prop="industry\_category">

<el-select v-model="enterpriseForm.industry\_category" placeholder="请选择行业分类">

<el-option

v-for="industry in industryOptions"

:key="industry.value"

:label="industry.label"

:value="industry.value"

/>

</el-select>

</el-form-item>

<el-form-item label="省份" prop="province">

<el-select v-model="enterpriseForm.province" placeholder="请选择省份" @change="handleProvinceChange">

<el-option

v-for="province in provinceOptions"

:key="province.value"

:label="province.label"

:value="province.value"

/>

</el-select>

</el-form-item>

<el-form-item label="城市" prop="city">

<el-select v-model="enterpriseForm.city" placeholder="请选择城市">

<el-option

v-for="city in cityOptions"

:key="city.value"

:label="city.label"

:value="city.value"

/>

</el-select>

</el-form-item>

<el-form-item label="详细地址" prop="registered\_address">

<el-input v-model="enterpriseForm.registered\_address" placeholder="请输入详细地址" />

</el-form-item>

<el-form-item label="用电规模" prop="power\_consumption\_level">

<el-select v-model="enterpriseForm.power\_consumption\_level" placeholder="请选择用电规模">

<el-option label="大型用户(>1000kVA)" value="large" />

<el-option label="中型用户(315-1000kVA)" value="medium" />

<el-option label="小型用户(<315kVA)" value="small" />

</el-select>

</el-form-item>

<el-form-item label="联系人" prop="contact\_person">

<el-input v-model="enterpriseForm.contact\_person" placeholder="请输入联系人姓名" />

</el-form-item>

<el-form-item label="联系电话" prop="contact\_phone">

<el-input v-model="enterpriseForm.contact\_phone" placeholder="请输入联系电话" />

</el-form-item>

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

<el-input

v-model="enterpriseForm.remarks"

type="textarea"

:rows="3"

placeholder="请输入备注信息"

/>

</el-form-item>

<el-form-item label="状态" prop="status">

<el-radio-group v-model="enterpriseForm.status">

<el-radio label="active">正常</el-radio>

<el-radio label="inactive">停用</el-radio>

</el-radio-group>

</el-form-item>

</el-form>

<template #footer>

<span class="dialog-footer">

<el-button @click="dialogVisible = false">取消</el-button>

<el-button type="primary" @click="handleSubmit" :loading="submitLoading">

确定

</el-button>

</span>

</template>

</el-dialog>

<!-- 批量导入对话框 -->

<el-dialog

v-model="importDialogVisible"

title="批量导入企业"

width="500px"

>

<div class="import-content">

<el-alert

title="导入说明"

type="info"

:closable="false"

show-icon

style="margin-bottom: 20px"

>

<p>1. 请使用提供的模板文件进行导入</p>

<p>2. 支持Excel文件格式(.xlsx, .xls)</p>

<p>3. 单次最多导入1000条数据</p>

</el-alert>

<div class="import-actions">

<el-button @click="downloadTemplate">

<el-icon><Download /></el-icon>

下载模板

</el-button>

</div>

<el-upload

ref="uploadRef"

class="upload-area"

drag

:auto-upload="false"

:limit="1"

accept=".xlsx,.xls"

:on-change="handleFileChange"

:file-list="fileList"

>

<el-icon class="el-icon--upload"><upload-filled /></el-icon>

<div class="el-upload\_\_text">

将文件拖到此处，或<em>点击上传</em>

</div>

<template #tip>

<div class="el-upload\_\_tip">

只能上传Excel文件，且不超过10MB

</div>

</template>

</el-upload>

<div v-if="importProgress.show" class="import-progress">

<el-progress :percentage="importProgress.percentage" :status="importProgress.status" />

<p>{{ importProgress.message }}</p>

</div>

</div>

<template #footer>

<span class="dialog-footer">

<el-button @click="importDialogVisible = false">取消</el-button>

<el-button

type="primary"

@click="handleImport"

:loading="importLoading"

:disabled="!fileList.length"

>

开始导入

</el-button>

</span>

</template>

</el-dialog>

<!-- 企业详情对话框 -->

<el-dialog

v-model="detailDialogVisible"

title="企业详情"

width="800px"

>

<div v-if="currentEnterprise" class="enterprise-details">

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

<el-descriptions-item label="企业名称">

{{ currentEnterprise.company\_name }}

</el-descriptions-item>

<el-descriptions-item label="企业简称">

{{ currentEnterprise.company\_short\_name }}

</el-descriptions-item>

<el-descriptions-item label="统一社会信用代码">

{{ currentEnterprise.credit\_code }}

</el-descriptions-item>

<el-descriptions-item label="行业分类">

{{ getIndustryLabel(currentEnterprise.industry\_category) }}

</el-descriptions-item>

<el-descriptions-item label="省份">

{{ currentEnterprise.province }}

</el-descriptions-item>

<el-descriptions-item label="城市">

{{ currentEnterprise.city }}

</el-descriptions-item>

<el-descriptions-item label="详细地址" :span="2">

{{ currentEnterprise.registered\_address }}

</el-descriptions-item>

<el-descriptions-item label="用电规模">

{{ getPowerLevelLabel(currentEnterprise.power\_consumption\_level) }}

</el-descriptions-item>

<el-descriptions-item label="联系人">

{{ currentEnterprise.contact\_person }}

</el-descriptions-item>

<el-descriptions-item label="联系电话">

{{ currentEnterprise.contact\_phone }}

</el-descriptions-item>

<el-descriptions-item label="客户数量">

{{ currentEnterprise.customer\_count }}

</el-descriptions-item>

<el-descriptions-item label="状态">

<el-tag :type="currentEnterprise.status === 'active' ? 'success' : 'danger'">

{{ currentEnterprise.status === 'active' ? '正常' : '停用' }}

</el-tag>

</el-descriptions-item>

<el-descriptions-item label="创建时间">

{{ formatDate(currentEnterprise.created\_at) }}

</el-descriptions-item>

<el-descriptions-item label="备注" :span="2">

{{ currentEnterprise.remarks || '无' }}

</el-descriptions-item>

</el-descriptions>

</div>

</el-dialog>

</div>

</template>

<script setup>

import { ref, reactive, computed, onMounted, watch } from 'vue'

import { ElMessage, ElMessageBox } from 'element-plus'

import { Plus, Upload, Download, Search, Refresh, UploadFilled } from '@element-plus/icons-vue'

import {

getEnterpriseList,

createEnterprise,

updateEnterprise,

deleteEnterprise as deleteEnterpriseApi,

batchDeleteEnterprise,

importEnterprise,

exportEnterprise,

getEnterpriseStatistics

} from '@/api/enterprise'

// 响应式数据

const loading = ref(false)

const submitLoading = ref(false)

const importLoading = ref(false)

const dialogVisible = ref(false)

const importDialogVisible = ref(false)

const detailDialogVisible = ref(false)

const isEdit = ref(false)

const selectedIds = ref([])

const tableData = ref([])

const currentEnterprise = ref(null)

// 统计数据

const statistics = ref([

{ key: 'total', title: '企业总数', value: 0, suffix: '家' },

{ key: 'active', title: '正常企业', value: 0, suffix: '家' },

{ key: 'large', title: '大型用户', value: 0, suffix: '家' },

{ key: 'today', title: '今日新增', value: 0, suffix: '家' }

])

// 搜索表单

const searchForm = reactive({

search: '',

province: '',

city: '',

industry\_category: '',

power\_consumption\_level: '',

status: ''

})

// 分页

const pagination = reactive({

page: 1,

limit: 20,

total: 0

})

// 企业表单

const enterpriseForm = reactive({

company\_name: '',

company\_short\_name: '',

credit\_code: '',

industry\_category: '',

province: '',

city: '',

registered\_address: '',

power\_consumption\_level: '',

contact\_person: '',

contact\_phone: '',

remarks: '',

status: 'active'

})

// 表单验证规则

const formRules = {

company\_name: [

{ required: true, message: '请输入企业名称', trigger: 'blur' },

{ min: 2, max: 100, message: '企业名称长度在 2 到 100 个字符', trigger: 'blur' }

],

credit\_code: [

{ required: true, message: '请输入统一社会信用代码', trigger: 'blur' },

{ len: 18, message: '统一社会信用代码必须是18位', trigger: 'blur' }

],

industry\_category: [

{ required: true, message: '请选择行业分类', trigger: 'change' }

],

province: [

{ required: true, message: '请选择省份', trigger: 'change' }

],

city: [

{ required: true, message: '请选择城市', trigger: 'change' }

],

power\_consumption\_level: [

{ required: true, message: '请选择用电规模', trigger: 'change' }

]

}

// 导入相关

const fileList = ref([])

const importProgress = reactive({

show: false,

percentage: 0,

status: '',

message: ''

})

// 选项数据

const provinceOptions = ref([

{ label: '北京市', value: '北京市' },

{ label: '上海市', value: '上海市' },

{ label: '广东省', value: '广东省' },

{ label: '浙江省', value: '浙江省' },

{ label: '江苏省', value: '江苏省' },

{ label: '山东省', value: '山东省' },

// ... 其他省份

])

const cityOptions = ref([])

const industryOptions = ref([

{ label: '制造业', value: 'manufacturing' },

{ label: '建筑业', value: 'construction' },

{ label: '批发和零售业', value: 'wholesale\_retail' },

{ label: '交通运输业', value: 'transportation' },

{ label: '住宿和餐饮业', value: 'accommodation\_catering' },

{ label: '信息技术业', value: 'information\_technology' },

{ label: '金融业', value: 'finance' },

{ label: '房地产业', value: 'real\_estate' },

{ label: '科学研究业', value: 'scientific\_research' },

{ label: '教育', value: 'education' },

{ label: '卫生和社会工作', value: 'health\_social\_work' },

{ label: '文化娱乐业', value: 'culture\_entertainment' },

{ label: '其他', value: 'other' }

])

// 计算属性

const dialogTitle = computed(() => isEdit.value ? '编辑企业' : '新增企业')

// 监听省份变化

watch(() => searchForm.province, (newVal) => {

if (newVal) {

updateCityOptions(newVal)

} else {

cityOptions.value = []

}

searchForm.city = ''

})

// 方法

const loadData = async () => {

loading.value = true

try {

const params = {

page: pagination.page,

limit: pagination.limit,

...searchForm

}

const response = await getEnterpriseList(params)

tableData.value = response.data.data

pagination.total = response.data.total

} catch (error) {

ElMessage.error('加载数据失败')

} finally {

loading.value = false

}

}

const loadStatistics = async () => {

try {

const response = await getEnterpriseStatistics()

const stats = response.data

statistics.value = [

{ key: 'total', title: '企业总数', value: stats.total\_count, suffix: '家' },

{ key: 'active', title: '正常企业', value: stats.active\_count, suffix: '家' },

{ key: 'large', title: '大型用户', value: stats.large\_users, suffix: '家' },

{ key: 'today', title: '今日新增', value: stats.today\_added, suffix: '家' }

]

} catch (error) {

console.error('加载统计数据失败', error)

}

}

const handleSearch = () => {

pagination.page = 1

loadData()

}

const resetSearch = () => {

Object.assign(searchForm, {

search: '',

province: '',

city: '',

industry\_category: '',

power\_consumption\_level: '',

status: ''

})

handleSearch()

}

const handlePageChange = (page) => {

pagination.page = page

loadData()

}

const handleSizeChange = (size) => {

pagination.limit = size

pagination.page = 1

loadData()

}

const handleSelectionChange = (selection) => {

selectedIds.value = selection.map(item => item.id)

}

const showCreateDialog = () => {

isEdit.value = false

resetForm()

dialogVisible.value = true

}

const editEnterprise = (row) => {

isEdit.value = true

Object.assign(enterpriseForm, row)

dialogVisible.value = true

}

const resetForm = () => {

Object.assign(enterpriseForm, {

company\_name: '',

company\_short\_name: '',

credit\_code: '',

industry\_category: '',

province: '',

city: '',

registered\_address: '',

power\_consumption\_level: '',

contact\_person: '',

contact\_phone: '',

remarks: '',

status: 'active'

})

}

const handleSubmit = async () => {

const formRef = ref()

if (!formRef.value) return

try {

await formRef.value.validate()

submitLoading.value = true

if (isEdit.value) {

await updateEnterprise(enterpriseForm.id, enterpriseForm)

ElMessage.success('企业信息更新成功')

} else {

await createEnterprise(enterpriseForm)

ElMessage.success('企业创建成功')

}

dialogVisible.value = false

loadData()

loadStatistics()

} catch (error) {

ElMessage.error(error.message || '操作失败')

} finally {

submitLoading.value = false

}

}

const handleDialogClose = () => {

resetForm()

}

const deleteEnterprise = async (id) => {

try {

await deleteEnterpriseApi(id)

ElMessage.success('删除成功')

loadData()

loadStatistics()

} catch (error) {

ElMessage.error('删除失败')

}

}

const handleBatchDelete = async () => {

try {

await ElMessageBox.confirm('确定要删除选中的企业吗？', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

})

await batchDeleteEnterprise(selectedIds.value)

ElMessage.success('批量删除成功')

loadData()

loadStatistics()

} catch (error) {

if (error !== 'cancel') {

ElMessage.error('批量删除失败')

}

}

}

const viewDetails = (row) => {

currentEnterprise.value = row

detailDialogVisible.value = true

}

const refreshTable = () => {

loadData()

}

const showImportDialog = () => {

fileList.value = []

importProgress.show = false

importDialogVisible.value = true

}

const handleFileChange = (file) => {

fileList.value = [file]

}

const downloadTemplate = () => {

// 这里应该下载模板文件

const link = document.createElement('a')

link.href = '/templates/enterprise\_import\_template.xlsx'

link.download = '企业导入模板.xlsx'

link.click()

}

const handleImport = async () => {

if (!fileList.value.length) {

ElMessage.warning('请选择要导入的文件')

return

}

importLoading.value = true

importProgress.show = true

importProgress.percentage = 0

importProgress.status = 'active'

importProgress.message = '正在上传文件...'

try {

const formData = new FormData()

formData.append('file', fileList.value[0].raw)

// 模拟进度更新

const progressInterval = setInterval(() => {

if (importProgress.percentage < 90) {

importProgress.percentage += 10

}

}, 500)

await importEnterprise(formData)

clearInterval(progressInterval)

importProgress.percentage = 100

importProgress.status = 'success'

importProgress.message = '导入完成'

ElMessage.success('导入成功')

importDialogVisible.value = false

loadData()

loadStatistics()

} catch (error) {

importProgress.status = 'exception'

importProgress.message = '导入失败'

ElMessage.error('导入失败')

} finally {

importLoading.value = false

}

}

const exportData = async () => {

try {

const response = await exportEnterprise(searchForm)

const blob = new Blob([response.data])

const link = document.createElement('a')

link.href = URL.createObjectURL(blob)

link.download = `企业名录\_${new Date().toLocaleDateString()}.xlsx`

link.click()

ElMessage.success('导出成功')

} catch (error) {

ElMessage.error('导出失败')

}

}

const handleBatchExport = async () => {

try {

const response = await exportEnterprise({ ids: selectedIds.value })

const blob = new Blob([response.data])

const link = document.createElement('a')

link.href = URL.createObjectURL(blob)

link.download = `选中企业\_${new Date().toLocaleDateString()}.xlsx`

link.click()

ElMessage.success('导出成功')

} catch (error) {

ElMessage.error('导出失败')

}

}

const updateCityOptions = (province) => {

// 这里应该根据省份获取城市列表

// 示例数据

const cityMap = {

'北京市': [{ label: '东城区', value: '东城区' }, { label: '西城区', value: '西城区' }],

'上海市': [{ label: '黄浦区', value: '黄浦区' }, { label: '徐汇区', value: '徐汇区' }],

'广东省': [{ label: '广州市', value: '广州市' }, { label: '深圳市', value: '深圳市' }]

}

cityOptions.value = cityMap[province] || []

}

const handleProvinceChange = (value) => {

updateCityOptions(value)

enterpriseForm.city = ''

}

const getIndustryLabel = (value) => {

const industry = industryOptions.value.find(item => item.value === value)

return industry ? industry.label : value

}

const getIndustryTagType = (value) => {

const typeMap = {

'manufacturing': 'primary',

'construction': 'success',

'wholesale\_retail': 'warning',

'transportation': 'info',

'information\_technology': 'primary'

}

return typeMap[value] || 'default'

}

const getPowerLevelLabel = (value) => {

const labelMap = {

'large': '大型用户',

'medium': '中型用户',

'small': '小型用户'

}

return labelMap[value] || value

}

const getPowerLevelTagType = (value) => {

const typeMap = {

'large': 'danger',

'medium': 'warning',

'small': 'success'

}

return typeMap[value] || 'default'

}

const formatDate = (dateString) => {

if (!dateString) return ''

return new Date(dateString).toLocaleDateString('zh-CN', {

year: 'numeric',

month: '2-digit',

day: '2-digit',

hour: '2-digit',

minute: '2-digit'

})

}

// 生命周期

onMounted(() => {

loadData()

loadStatistics()

})

</script>

<style scoped>

.enterprise-directory {

padding: 20px;

}

.page-header {

display: flex;

justify-content: space-between;

align-items: flex-start;

margin-bottom: 20px;

background: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 8px rgba(0,0,0,0.1);

}

.header-content {

flex: 1;

}

.page-title {

font-size: 24px;

font-weight: 600;

color: #1f2329;

margin: 0 0 16px 0;

}

.header-stats {

display: flex;

gap: 40px;

}

.stat-item {

background: none;

}

.stat-item :deep(.el-statistic\_\_head) {

color: #86909c;

font-size: 14px;

}

.stat-item :deep(.el-statistic\_\_content) {

color: #1f2329;

font-size: 20px;

font-weight: 600;

}

.header-actions {

display: flex;

gap: 12px;

align-items: flex-start;

}

.search-card {

margin-bottom: 20px;

border: none;

}

.search-card :deep(.el-card\_\_body) {

padding: 20px;

}

.search-form {

display: flex;

flex-wrap: wrap;

gap: 8px 20px;

}

.search-form .el-form-item {

margin-bottom: 12px;

}

.table-card {

border: none;

}

.table-card :deep(.el-card\_\_body) {

padding: 0;

}

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 20px;

border-bottom: 1px solid #e4e7ed;

}

.card-header span {

font-size: 16px;

font-weight: 600;

color: #1f2329;

}

.table-actions {

display: flex;

gap: 8px;

}

.el-table {

border-radius: 0;

}

.el-table :deep(.el-table\_\_header) {

background-color: #f7f8fa;

}

.el-table :deep(.el-table\_\_header th) {

background-color: #f7f8fa;

color: #1f2329;

font-weight: 600;

border-bottom: 1px solid #e4e7ed;

}

.el-table :deep(.el-table\_\_row:hover) {

background-color: #f7f8fa;

}

.pagination-wrapper {

display: flex;

justify-content: flex-end;

padding: 20px;

border-top: 1px solid #e4e7ed;

}

.dialog-footer {

text-align: right;

}

.dialog-footer .el-button {

margin-left: 8px;

}

.import-content {

padding: 0;

}

.import-actions {

margin-bottom: 20px;

text-align: center;

}

.upload-area {

margin: 20px 0;

}

.upload-area :deep(.el-upload-dragger) {

width: 100%;

height: 120px;

border: 2px dashed #d9d9d9;

border-radius: 6px;

cursor: pointer;

position: relative;

overflow: hidden;

transition: border-color 0.3s;

}

.upload-area :deep(.el-upload-dragger:hover) {

border-color: #409eff;

}

.upload-area :deep(.el-icon--upload) {

font-size: 32px;

color: #c0c4cc;

margin-bottom: 8px;

}

.upload-area :deep(.el-upload\_\_text) {

color: #606266;

font-size: 14px;

text-align: center;

}

.upload-area :deep(.el-upload\_\_text em) {

color: #409eff;

font-style: normal;

}

.import-progress {

margin-top: 20px;

}

.import-progress p {

margin-top: 8px;

text-align: center;

color: #606266;

font-size: 14px;

}

.enterprise-details {

padding: 0;

}

.enterprise-details :deep(.el-descriptions) {

margin-top: 0;

}

.enterprise-details :deep(.el-descriptions\_\_header) {

display: none;

}

.enterprise-details :deep(.el-descriptions-item\_\_label) {

font-weight: 600;

color: #1f2329;

background-color: #f7f8fa;

}

.enterprise-details :deep(.el-descriptions-item\_\_content) {

color: #4e5969;

}

/\* 响应式设计 \*/

@media (max-width: 1200px) {

.header-stats {

gap: 20px;

}

.search-form {

gap: 8px 16px;

}

}

@media (max-width: 768px) {

.enterprise-directory {

padding: 16px;

}

.page-header {

flex-direction: column;

gap: 16px;

}

.header-stats {

flex-wrap: wrap;

gap: 16px;

}

.header-actions {

align-self: stretch;

justify-content: flex-end;

}

.search-form {

flex-direction: column;

gap: 12px;

}

.search-form .el-form-item {

width: 100%;

}

.card-header {

flex-direction: column;

gap: 12px;

align-items: stretch;

}

.table-actions {

justify-content: flex-end;

}

.pagination-wrapper {

padding: 16px;

}

}

/\* 表格优化 \*/

.el-table :deep(.cell) {

padding: 8px 12px;

}

.el-table :deep(.el-table\_\_empty-text) {

color: #909399;

}

/\* 标签样式优化 \*/

.el-tag {

border-radius: 4px;

font-size: 12px;

height: 22px;

line-height: 20px;

}

/\* 按钮组样式 \*/

.el-button-group {

display: inline-flex;

}

.el-button + .el-button {

margin-left: 0;

}

/\* 表单验证错误样式 \*/

.el-form-item.is-error :deep(.el-input\_\_inner) {

border-color: #f56c6c;

}

.el-form-item.is-error :deep(.el-input\_\_inner):focus {

border-color: #f56c6c;

}

/\* 加载状态优化 \*/

.el-loading-mask {

background-color: rgba(255, 255, 255, 0.8);

}

/\* 对话框样式优化 \*/

.el-dialog {

border-radius: 8px;

}

.el-dialog :deep(.el-dialog\_\_header) {

padding: 20px 20px 0;

border-bottom: 1px solid #e4e7ed;

}

.el-dialog :deep(.el-dialog\_\_body) {

padding: 20px;

max-height: 60vh;

overflow-y: auto;

}

.el-dialog :deep(.el-dialog\_\_footer) {

padding: 0 20px 20px;

border-top: 1px solid #e4e7ed;

}

/\* 进度条样式 \*/

.el-progress--line {

margin: 16px 0;

}

.el-progress :deep(.el-progress\_\_text) {

font-size: 14px;

}

/\* 统计卡片动画 \*/

.stat-item {

transition: transform 0.2s;

}

.stat-item:hover {

transform: translateY(-2px);

}

/\* 表格行操作按钮 \*/

.el-table :deep(.el-button--small) {

padding: 5px 8px;

font-size: 12px;

}

/\* 搜索表单优化 \*/

.search-form .el-select {

width: 150px;

}

.search-form .el-date-picker {

width: 200px;

}

/\* 卡片阴影效果 \*/

.el-card {

transition: box-shadow 0.3s;

}

.el-card:hover {

box-shadow: 0 4px 12px rgba(0,0,0,0.15);

}

/\* 提示信息样式 \*/

.el-alert {

border-radius: 6px;

}

.el-alert :deep(.el-alert\_\_content) {

padding: 0;

}

.el-alert p {

margin: 4px 0;

line-height: 1.5;

}

/\* 上传组件优化 \*/

.el-upload\_\_tip {

margin-top: 8px;

color: #909399;

font-size: 12px;

line-height: 1.5;

}

/\* 描述列表样式 \*/

.el-descriptions :deep(.el-descriptions\_\_body) {

background-color: #fff;

}

.el-descriptions :deep(.el-descriptions-item) {

vertical-align: top;

}

/\* 表格空状态 \*/

.el-table :deep(.el-table\_\_empty-block) {

padding: 60px 0;

}

.el-table :deep(.el-table\_\_empty-text) {

font-size: 14px;

color: #909399;

}

/\* 分页组件优化 \*/

.el-pagination {

text-align: right;

}

.el-pagination :deep(.el-pagination\_\_total) {

margin-right: auto;

}

/\* 表单标签对齐 \*/

.el-form--label-top .el-form-item\_\_label {

text-align: left;

padding-bottom: 8px;

}

/\* 响应式表格 \*/

@media (max-width: 768px) {

.el-table :deep(.el-table\_\_header-wrapper) {

overflow-x: auto;

}

.el-table :deep(.el-table\_\_body-wrapper) {

overflow-x: auto;

}

}

/\* 深色模式适配 \*/

@media (prefers-color-scheme: dark) {

.page-header {

background: #2c2c2c;

}

.page-title {

color: #e4e7ed;

}

.el-card {

background: #2c2c2c;

border-color: #4c4d4f;

}

}

</style>

现在企业名录管理前端页面已经完成，接下来开发下一个模块：**智能匹配API和小程序集成**。这个模块将实现企业名录的智能匹配功能，帮助客户经理快速找到潜在客户。

智能匹配API后端 - intelligent\_matching.py

# intelligent\_matching.py - 智能匹配API后端

# 语言：Python 3.9+ (FastAPI)

from fastapi import APIRouter, Depends, HTTPException, Query

from sqlalchemy.orm import Session

from typing import List, Optional, Dict, Any

from pydantic import BaseModel, Field

from datetime import datetime, timedelta

import math

from geopy.distance import geodesic

from database import get\_db

from models.enterprise import Enterprise

from models.user import User

from models.customer\_assignment import CustomerAssignment

from auth import get\_current\_user

router = APIRouter(prefix="/api/v1/intelligent-matching", tags=["智能匹配"])

# 请求模型

class MatchingCriteria(BaseModel):

"""匹配条件"""

industry\_categories: Optional[List[str]] = Field(None, description="行业分类列表")

power\_consumption\_levels: Optional[List[str]] = Field(None, description="用电规模列表")

provinces: Optional[List[str]] = Field(None, description="省份列表")

cities: Optional[List[str]] = Field(None, description="城市列表")

min\_customer\_count: Optional[int] = Field(None, description="最小客户数量")

max\_customer\_count: Optional[int] = Field(None, description="最大客户数量")

exclude\_assigned: bool = Field(True, description="排除已分配客户")

location\_priority: bool = Field(False, description="位置优先级")

user\_location: Optional[Dict[str, float]] = Field(None, description="用户位置 {lat, lng}")

similarity\_threshold: float = Field(0.3, description="相似度阈值")

class SmartMatchingRequest(BaseModel):

"""智能匹配请求"""

user\_id: Optional[int] = Field(None, description="用户ID，用于个性化匹配")

employee\_id: Optional[int] = Field(None, description="员工ID，用于区域匹配")

criteria: MatchingCriteria = Field(..., description="匹配条件")

limit: int = Field(50, le=200, description="返回结果数量限制")

strategy: str = Field("comprehensive", description="匹配策略: comprehensive/location/industry/potential")

class BatchMatchingRequest(BaseModel):

"""批量匹配请求"""

employee\_ids: List[int] = Field(..., description="员工ID列表")

criteria: MatchingCriteria = Field(..., description="匹配条件")

auto\_assign: bool = Field(False, description="是否自动分配")

# 响应模型

class MatchingScore(BaseModel):

"""匹配分数详情"""

total\_score: float = Field(..., description="总分")

industry\_score: float = Field(..., description="行业匹配分")

location\_score: float = Field(..., description="位置匹配分")

scale\_score: float = Field(..., description="规模匹配分")

potential\_score: float = Field(..., description="潜力分")

activity\_score: float = Field(..., description="活跃度分")

class EnterpriseMatch(BaseModel):

"""企业匹配结果"""

enterprise\_id: int

company\_name: str

company\_short\_name: Optional[str]

industry\_category: str

province: str

city: str

power\_consumption\_level: str

customer\_count: int

contact\_person: Optional[str]

contact\_phone: Optional[str]

distance: Optional[float] = Field(None, description="距离(公里)")

matching\_score: MatchingScore

match\_reasons: List[str] = Field(..., description="匹配原因")

recommended\_approach: str = Field(..., description="推荐接触方式")

class MatchingResult(BaseModel):

"""匹配结果"""

total\_matches: int

matches: List[EnterpriseMatch]

matching\_strategy: str

execution\_time: float

# 智能匹配服务类

class IntelligentMatchingService:

def \_\_init\_\_(self, db: Session):

self.db = db

def calculate\_industry\_similarity(self, industry1: str, industry2: str) -> float:

"""计算行业相似度"""

if industry1 == industry2:

return 1.0

# 行业相关性矩阵

industry\_relations = {

'manufacturing': ['construction', 'wholesale\_retail'],

'construction': ['manufacturing', 'real\_estate'],

'information\_technology': ['scientific\_research', 'finance'],

'wholesale\_retail': ['manufacturing', 'transportation'],

'transportation': ['wholesale\_retail', 'accommodation\_catering'],

'finance': ['information\_technology', 'real\_estate'],

'real\_estate': ['construction', 'finance'],

'education': ['health\_social\_work', 'culture\_entertainment'],

'health\_social\_work': ['education'],

'culture\_entertainment': ['education', 'accommodation\_catering']

}

if industry2 in industry\_relations.get(industry1, []):

return 0.6

return 0.2

def calculate\_location\_score(self, enterprise: Enterprise, user\_location: Dict[str, float]) -> tuple:

"""计算位置匹配分数"""

if not user\_location or not enterprise.latitude or not enterprise.longitude:

return 0.5, None

distance = geodesic(

(user\_location['lat'], user\_location['lng']),

(enterprise.latitude, enterprise.longitude)

).kilometers

# 距离评分：距离越近分数越高

if distance <= 10:

score = 1.0

elif distance <= 50:

score = 0.8

elif distance <= 100:

score = 0.6

elif distance <= 200:

score = 0.4

else:

score = 0.2

return score, distance

def calculate\_scale\_compatibility(self, enterprise: Enterprise, user\_power\_level: str = None) -> float:

"""计算规模匹配度"""

if not user\_power\_level:

return 0.7

if enterprise.power\_consumption\_level == user\_power\_level:

return 1.0

# 规模匹配矩阵

scale\_compatibility = {

'large': {'medium': 0.7, 'small': 0.3},

'medium': {'large': 0.6, 'small': 0.8},

'small': {'medium': 0.7, 'large': 0.2}

}

return scale\_compatibility.get(user\_power\_level, {}).get(enterprise.power\_consumption\_level, 0.5)

def calculate\_potential\_score(self, enterprise: Enterprise) -> float:

"""计算企业潜力分数"""

score = 0.5 # 基础分

# 客户数量因子

if enterprise.customer\_count > 100:

score += 0.3

elif enterprise.customer\_count > 50:

score += 0.2

elif enterprise.customer\_count > 10:

score += 0.1

# 行业潜力因子

high\_potential\_industries = ['manufacturing', 'information\_technology', 'scientific\_research']

if enterprise.industry\_category in high\_potential\_industries:

score += 0.2

return min(score, 1.0)

def calculate\_activity\_score(self, enterprise: Enterprise) -> float:

"""计算活跃度分数"""

# 基于最近创建时间和更新时间

now = datetime.now()

created\_days = (now - enterprise.created\_at).days

if created\_days <= 30:

return 0.9

elif created\_days <= 90:

return 0.7

elif created\_days <= 180:

return 0.5

else:

return 0.3

def generate\_match\_reasons(self, enterprise: Enterprise, scores: Dict[str, float]) -> List[str]:

"""生成匹配原因"""

reasons = []

if scores['industry\_score'] >= 0.8:

reasons.append(f"所属{enterprise.industry\_category}行业高度匹配")

if scores['location\_score'] >= 0.8:

reasons.append("地理位置便于拜访")

if scores['scale\_score'] >= 0.8:

reasons.append("用电规模匹配")

if scores['potential\_score'] >= 0.7:

reasons.append("企业发展潜力较大")

if enterprise.customer\_count > 50:

reasons.append("客户基数较大，业务机会多")

if not reasons:

reasons.append("综合评估具有合作潜力")

return reasons

def get\_recommended\_approach(self, enterprise: Enterprise, total\_score: float) -> str:

"""获取推荐接触方式"""

if total\_score >= 0.8:

return "重点客户，建议电话预约后上门拜访"

elif total\_score >= 0.6:

return "优质潜客，建议电话沟通了解需求"

elif total\_score >= 0.4:

return "一般潜客，可通过短信或邮件初步接触"

else:

return "低优先级，适合批量营销触达"

# API端点

@router.post("/smart-match", response\_model=MatchingResult)

async def smart\_matching(

request: SmartMatchingRequest,

current\_user = Depends(get\_current\_user),

db: Session = Depends(get\_db)

):

"""智能匹配企业"""

start\_time = datetime.now()

service = IntelligentMatchingService(db)

# 构建查询

query = db.query(Enterprise).filter(Enterprise.status == 'active')

# 应用筛选条件

criteria = request.criteria

if criteria.industry\_categories:

query = query.filter(Enterprise.industry\_category.in\_(criteria.industry\_categories))

if criteria.power\_consumption\_levels:

query = query.filter(Enterprise.power\_consumption\_level.in\_(criteria.power\_consumption\_levels))

if criteria.provinces:

query = query.filter(Enterprise.province.in\_(criteria.provinces))

if criteria.cities:

query = query.filter(Enterprise.city.in\_(criteria.cities))

if criteria.min\_customer\_count:

query = query.filter(Enterprise.customer\_count >= criteria.min\_customer\_count)

if criteria.max\_customer\_count:

query = query.filter(Enterprise.customer\_count <= criteria.max\_customer\_count)

# 排除已分配的企业

if criteria.exclude\_assigned:

assigned\_enterprise\_ids = db.query(CustomerAssignment.enterprise\_id).filter(

CustomerAssignment.status == 'active'

).subquery()

query = query.filter(~Enterprise.id.in\_(assigned\_enterprise\_ids))

enterprises = query.limit(request.limit \* 2).all() # 获取更多候选，后续筛选

# 计算匹配分数

matches = []

user\_info = None

if request.user\_id:

user\_info = db.query(User).filter(User.id == request.user\_id).first()

for enterprise in enterprises:

# 计算各项分数

industry\_score = 0.7 # 默认行业分数

if user\_info and user\_info.industry\_category:

industry\_score = service.calculate\_industry\_similarity(

enterprise.industry\_category,

user\_info.industry\_category

)

location\_score, distance = service.calculate\_location\_score(

enterprise,

criteria.user\_location

)

scale\_score = service.calculate\_scale\_compatibility(enterprise)

potential\_score = service.calculate\_potential\_score(enterprise)

activity\_score = service.calculate\_activity\_score(enterprise)

# 根据策略调整权重

if request.strategy == "location":

weights = {'industry': 0.2, 'location': 0.4, 'scale': 0.1, 'potential': 0.2, 'activity': 0.1}

elif request.strategy == "industry":

weights = {'industry': 0.4, 'location': 0.1, 'scale': 0.3, 'potential': 0.1, 'activity': 0.1}

elif request.strategy == "potential":

weights = {'industry': 0.2, 'location': 0.1, 'scale': 0.1, 'potential': 0.4, 'activity': 0.2}

else: # comprehensive

weights = {'industry': 0.25, 'location': 0.2, 'scale': 0.2, 'potential': 0.2, 'activity': 0.15}

total\_score = (

industry\_score \* weights['industry'] +

location\_score \* weights['location'] +

scale\_score \* weights['scale'] +

potential\_score \* weights['potential'] +

activity\_score \* weights['activity']

)

# 过滤低分匹配

if total\_score < criteria.similarity\_threshold:

continue

scores = {

'industry\_score': industry\_score,

'location\_score': location\_score,

'scale\_score': scale\_score,

'potential\_score': potential\_score,

'activity\_score': activity\_score

}

match\_reasons = service.generate\_match\_reasons(enterprise, scores)

recommended\_approach = service.get\_recommended\_approach(enterprise, total\_score)

matches.append(EnterpriseMatch(

enterprise\_id=enterprise.id,

company\_name=enterprise.company\_name,

company\_short\_name=enterprise.company\_short\_name,

industry\_category=enterprise.industry\_category,

province=enterprise.province,

city=enterprise.city,

power\_consumption\_level=enterprise.power\_consumption\_level,

customer\_count=enterprise.customer\_count,

contact\_person=enterprise.contact\_person,

contact\_phone=enterprise.contact\_phone,

distance=distance,

matching\_score=MatchingScore(

total\_score=round(total\_score, 3),

industry\_score=round(industry\_score, 3),

location\_score=round(location\_score, 3),

scale\_score=round(scale\_score, 3),

potential\_score=round(potential\_score, 3),

activity\_score=round(activity\_score, 3)

),

match\_reasons=match\_reasons,

recommended\_approach=recommended\_approach

))

# 按总分排序

matches.sort(key=lambda x: x.matching\_score.total\_score, reverse=True)

matches = matches[:request.limit]

execution\_time = (datetime.now() - start\_time).total\_seconds()

return MatchingResult(

total\_matches=len(matches),

matches=matches,

matching\_strategy=request.strategy,

execution\_time=round(execution\_time, 3)

)

@router.post("/batch-match", response\_model=List[Dict[str, Any]])

async def batch\_matching(

request: BatchMatchingRequest,

current\_user = Depends(get\_current\_user),

db: Session = Depends(get\_db)

):

"""批量匹配分配"""

results = []

for employee\_id in request.employee\_ids:

# 为每个员工进行匹配

matching\_request = SmartMatchingRequest(

employee\_id=employee\_id,

criteria=request.criteria,

limit=20 # 每人最多匹配20个

)

try:

result = await smart\_matching(matching\_request, current\_user, db)

employee\_result = {

"employee\_id": employee\_id,

"matches\_count": result.total\_matches,

"top\_matches": result.matches[:5], # 返回前5个最佳匹配

"success": True

}

# 如果开启自动分配

if request.auto\_assign and result.matches:

assigned\_count = 0

for match in result.matches[:3]: # 自动分配前3个最佳匹配

assignment = CustomerAssignment(

enterprise\_id=match.enterprise\_id,

employee\_id=employee\_id,

assigned\_by=current\_user.id,

assigned\_at=datetime.now(),

status='active',

assignment\_type='auto',

remarks=f"系统智能匹配分配，匹配分数: {match.matching\_score.total\_score}"

)

db.add(assignment)

assigned\_count += 1

employee\_result["auto\_assigned"] = assigned\_count

db.commit()

results.append(employee\_result)

except Exception as e:

results.append({

"employee\_id": employee\_id,

"success": False,

"error": str(e)

})

return results

@router.get("/matching-statistics")

async def get\_matching\_statistics(

employee\_id: Optional[int] = Query(None),

days: int = Query(30, description="统计天数"),

current\_user = Depends(get\_current\_user),

db: Session = Depends(get\_db)

):

"""获取匹配统计数据"""

end\_date = datetime.now()

start\_date = end\_date - timedelta(days=days)

query = db.query(CustomerAssignment).filter(

CustomerAssignment.assigned\_at >= start\_date,

CustomerAssignment.assigned\_at <= end\_date

)

if employee\_id:

query = query.filter(CustomerAssignment.employee\_id == employee\_id)

assignments = query.all()

# 统计数据

total\_assignments = len(assignments)

auto\_assignments = len([a for a in assignments if a.assignment\_type == 'auto'])

manual\_assignments = total\_assignments - auto\_assignments

# 按日期统计

daily\_stats = {}

for assignment in assignments:

date\_key = assignment.assigned\_at.strftime('%Y-%m-%d')

if date\_key not in daily\_stats:

daily\_stats[date\_key] = 0

daily\_stats[date\_key] += 1

# 按行业统计

industry\_stats = {}

for assignment in assignments:

if assignment.enterprise and assignment.enterprise.industry\_category:

industry = assignment.enterprise.industry\_category

if industry not in industry\_stats:

industry\_stats[industry] = 0

industry\_stats[industry] += 1

return {

"period": {

"start\_date": start\_date.strftime('%Y-%m-%d'),

"end\_date": end\_date.strftime('%Y-%m-%d'),

"days": days

},

"summary": {

"total\_assignments": total\_assignments,

"auto\_assignments": auto\_assignments,

"manual\_assignments": manual\_assignments,

"auto\_assignment\_rate": round(auto\_assignments / total\_assignments \* 100, 2) if total\_assignments > 0 else 0

},

"daily\_distribution": daily\_stats,

"industry\_distribution": industry\_stats

}

@router.get("/matching-suggestions")

async def get\_matching\_suggestions(

employee\_id: int = Query(..., description="员工ID"),

current\_user = Depends(get\_current\_user),

db: Session = Depends(get\_db)

):

"""获取匹配建议"""

# 获取员工信息

employee = db.query(Employee).filter(Employee.id == employee\_id).first()

if not employee:

raise HTTPException(status\_code=404, detail="员工不存在")

# 获取员工已分配的客户

assigned\_enterprises = db.query(CustomerAssignment.enterprise\_id).filter(

CustomerAssignment.employee\_id == employee\_id,

CustomerAssignment.status == 'active'

).subquery()

assigned\_count = db.query(assigned\_enterprises).count()

# 分析员工擅长的行业

industry\_analysis = db.query(

Enterprise.industry\_category,

db.func.count(Enterprise.id).label('count')

).join(

CustomerAssignment,

Enterprise.id == CustomerAssignment.enterprise\_id

).filter(

CustomerAssignment.employee\_id == employee\_id,

CustomerAssignment.status == 'active'

).group\_by(Enterprise.industry\_category).all()

preferred\_industries = [item.industry\_category for item in industry\_analysis]

# 建议匹配条件

suggestions = {

"employee\_info": {

"id": employee.id,

"name": employee.name,

"department": employee.department.name if employee.department else None,

"current\_assignments": assigned\_count

},

"workload\_analysis": {

"current\_load": assigned\_count,

"recommended\_load": 30, # 建议负载

"can\_assign\_more": assigned\_count < 30

},

"preferred\_industries": preferred\_industries,

"suggested\_criteria": {

"industry\_categories": preferred\_industries[:3] if preferred\_industries else None,

"power\_consumption\_levels": ["medium", "large"], # 建议关注中大型企业

"min\_customer\_count": 10,

"exclude\_assigned": True,

"similarity\_threshold": 0.4

},

"matching\_tips": [

"关注您擅长的行业类型",

"优先选择中大型用电企业",

"考虑地理位置便于拜访的企业",

"重点关注有发展潜力的新兴企业"

]

}

return suggestions

@router.post("/save-matching-template")

async def save\_matching\_template(

template\_name: str = Query(..., description="模板名称"),

criteria: MatchingCriteria = ...,

current\_user = Depends(get\_current\_user),

db: Session = Depends(get\_db)

):

"""保存匹配模板"""

template = MatchingTemplate(

name=template\_name,

criteria=criteria.dict(),

created\_by=current\_user.id,

created\_at=datetime.now()

)

db.add(template)

db.commit()

db.refresh(template)

return {

"message": "匹配模板保存成功",

"template\_id": template.id

}

@router.get("/matching-templates")

async def get\_matching\_templates(

current\_user = Depends(get\_current\_user),

db: Session = Depends(get\_db)

):

"""获取匹配模板列表"""

templates = db.query(MatchingTemplate).filter(

MatchingTemplate.created\_by == current\_user.id

).order\_by(MatchingTemplate.created\_at.desc()).all()

return [

{

"id": template.id,

"name": template.name,

"criteria": template.criteria,

"created\_at": template.created\_at.strftime('%Y-%m-%d %H:%M:%S')

}

for template in templates

]

@router.post("/analyze-matching-performance")

async def analyze\_matching\_performance(

employee\_ids: List[int] = Query(..., description="员工ID列表"),

days: int = Query(30, description="分析天数"),

current\_user = Depends(get\_current\_user),

db: Session = Depends(get\_db)

):

"""分析匹配效果"""

end\_date = datetime.now()

start\_date = end\_date - timedelta(days=days)

performance\_data = []

for employee\_id in employee\_ids:

employee = db.query(Employee).filter(Employee.id == employee\_id).first()

if not employee:

continue

# 获取分配记录

assignments = db.query(CustomerAssignment).filter(

CustomerAssignment.employee\_id == employee\_id,

CustomerAssignment.assigned\_at >= start\_date,

CustomerAssignment.assigned\_at <= end\_date

).all()

# 获取跟进记录

follow\_ups = db.query(CustomerFollowLog).filter(

CustomerFollowLog.employee\_id == employee\_id,

CustomerFollowLog.created\_at >= start\_date,

CustomerFollowLog.created\_at <= end\_date

).all()

# 获取成交订单

orders = db.query(Order).join(

CustomerAssignment,

Order.user\_id == CustomerAssignment.user\_id

).filter(

CustomerAssignment.employee\_id == employee\_id,

Order.created\_at >= start\_date,

Order.created\_at <= end\_date,

Order.status.in\_(['completed', 'signed'])

).all()

# 计算转化率

total\_assignments = len(assignments)

total\_orders = len(orders)

conversion\_rate = (total\_orders / total\_assignments \* 100) if total\_assignments > 0 else 0

# 计算跟进效率

follow\_up\_rate = (len(set([f.user\_id for f in follow\_ups])) / total\_assignments \* 100) if total\_assignments > 0 else 0

performance\_data.append({

"employee\_id": employee\_id,

"employee\_name": employee.name,

"department": employee.department.name if employee.department else None,

"assignments\_count": total\_assignments,

"follow\_ups\_count": len(follow\_ups),

"orders\_count": total\_orders,

"conversion\_rate": round(conversion\_rate, 2),

"follow\_up\_rate": round(follow\_up\_rate, 2),

"average\_order\_amount": round(sum([o.amount for o in orders]) / len(orders), 2) if orders else 0,

"performance\_score": round((conversion\_rate \* 0.6 + follow\_up\_rate \* 0.4), 2)

})

# 排序

performance\_data.sort(key=lambda x: x['performance\_score'], reverse=True)

return {

"analysis\_period": {

"start\_date": start\_date.strftime('%Y-%m-%d'),

"end\_date": end\_date.strftime('%Y-%m-%d'),

"days": days

},

"summary": {

"total\_employees": len(performance\_data),

"total\_assignments": sum([p['assignments\_count'] for p in performance\_data]),

"total\_orders": sum([p['orders\_count'] for p in performance\_data]),

"average\_conversion\_rate": round(sum([p['conversion\_rate'] for p in performance\_data]) / len(performance\_data), 2) if performance\_data else 0

},

"employee\_performance": performance\_data

}

# 需要添加的数据模型 (在 models/matching\_template.py 中)

class MatchingTemplate(Base):

\_\_tablename\_\_ = "matching\_templates"

id = Column(Integer, primary\_key=True, index=True)

name = Column(String(100), nullable=False, comment="模板名称")

criteria = Column(JSON, nullable=False, comment="匹配条件JSON")

created\_by = Column(Integer, ForeignKey("employees.id"), comment="创建人")

created\_at = Column(DateTime, default=datetime.now, comment="创建时间")

updated\_at = Column(DateTime, default=datetime.now, onupdate=datetime.now, comment="更新时间")

creator = relationship("Employee", backref="matching\_templates")

现在我来开发**智能匹配小程序集成**模块：

智能匹配小程序页面和组件

// 智能匹配小程序页面和组件

// 语言：JavaScript (微信小程序)

// pages/intelligent-matching/index.js - 智能匹配主页面

Page({

data: {

// 匹配条件

criteria: {

industry\_categories: [],

power\_consumption\_levels: [],

provinces: [],

cities: [],

min\_customer\_count: null,

max\_customer\_count: null,

exclude\_assigned: true,

location\_priority: false,

user\_location: null,

similarity\_threshold: 0.3

},

// 匹配策略

strategies: [

{ value: 'comprehensive', label: '综合匹配', desc: '平衡各项因素' },

{ value: 'location', label: '位置优先', desc: '优先推荐附近企业' },

{ value: 'industry', label: '行业优先', desc: '优先推荐同行业' },

{ value: 'potential', label: '潜力优先', desc: '优先推荐高潜力企业' }

],

selectedStrategy: 'comprehensive',

// 行业选项

industries: [

{ value: 'manufacturing', label: '制造业' },

{ value: 'construction', label: '建筑业' },

{ value: 'information\_technology', label: '信息技术' },

{ value: 'wholesale\_retail', label: '批发零售' },

{ value: 'transportation', label: '交通运输' },

{ value: 'finance', label: '金融业' },

{ value: 'real\_estate', label: '房地产' },

{ value: 'education', label: '教育' },

{ value: 'health\_social\_work', label: '卫生社会工作' },

{ value: 'culture\_entertainment', label: '文化娱乐' }

],

// 用电规模选项

powerLevels: [

{ value: 'small', label: '小型(≤100kW)' },

{ value: 'medium', label: '中型(100-1000kW)' },

{ value: 'large', label: '大型(>1000kW)' }

],

// 省份选项

provinces: [

'北京市', '上海市', '天津市', '重庆市',

'河北省', '山西省', '辽宁省', '吉林省', '黑龙江省',

'江苏省', '浙江省', '安徽省', '福建省', '江西省', '山东省',

'河南省', '湖北省', '湖南省', '广东省', '海南省',

'四川省', '贵州省', '云南省', '陕西省', '甘肃省', '青海省',

'内蒙古自治区', '广西壮族自治区', '西藏自治区', '宁夏回族自治区', '新疆维吾尔自治区'

],

// 匹配结果

matching: false,

matchResults: [],

totalMatches: 0,

executionTime: 0,

// 界面状态

showCriteriaPanel: false,

showResultDetail: false,

selectedResult: null,

// 用户位置

userLocation: null,

locationPermission: false

},

onLoad(options) {

// 检查是否为客户经理

const userInfo = wx.getStorageSync('userInfo') || {};

if (userInfo.role !== 'manager') {

wx.showToast({

title: '仅客户经理可使用',

icon: 'error'

});

setTimeout(() => {

wx.navigateBack();

}, 1500);

return;

}

this.loadMatchingTemplates();

this.getUserLocation();

},

// 获取用户位置

getUserLocation() {

wx.getLocation({

type: 'gcj02',

success: (res) => {

this.setData({

userLocation: {

lat: res.latitude,

lng: res.longitude

},

locationPermission: true,

'criteria.user\_location': {

lat: res.latitude,

lng: res.longitude

}

});

},

fail: () => {

wx.showModal({

title: '位置权限',

content: '开启位置权限可获得更精准的匹配结果',

confirmText: '去设置',

success: (res) => {

if (res.confirm) {

wx.openSetting();

}

}

});

}

});

},

// 加载匹配模板

loadMatchingTemplates() {

wx.request({

url: `${getApp().globalData.apiBase}/api/v1/intelligent-matching/matching-templates`,

header: {

'Authorization': `Bearer ${wx.getStorageSync('token')}`

},

success: (res) => {

if (res.statusCode === 200) {

this.setData({

templates: res.data

});

}

}

});

},

// 显示条件设置面板

showCriteriaPanel() {

this.setData({ showCriteriaPanel: true });

},

// 隐藏条件设置面板

hideCriteriaPanel() {

this.setData({ showCriteriaPanel: false });

},

// 行业选择

onIndustryChange(e) {

const values = e.detail.value;

const selectedIndustries = values.map(index => this.data.industries[index].value);

this.setData({

'criteria.industry\_categories': selectedIndustries

});

},

// 用电规模选择

onPowerLevelChange(e) {

const values = e.detail.value;

const selectedLevels = values.map(index => this.data.powerLevels[index].value);

this.setData({

'criteria.power\_consumption\_levels': selectedLevels

});

},

// 省份选择

onProvinceChange(e) {

const values = e.detail.value;

const selectedProvinces = values.map(index => this.data.provinces[index]);

this.setData({

'criteria.provinces': selectedProvinces

});

},

// 客户数量范围输入

onMinCustomerInput(e) {

this.setData({

'criteria.min\_customer\_count': parseInt(e.detail.value) || null

});

},

onMaxCustomerInput(e) {

this.setData({

'criteria.max\_customer\_count': parseInt(e.detail.value) || null

});

},

// 相似度阈值调整

onThresholdChange(e) {

this.setData({

'criteria.similarity\_threshold': e.detail.value / 100

});

},

// 策略选择

onStrategyChange(e) {

this.setData({

selectedStrategy: e.detail.value

});

},

// 位置优先开关

onLocationPriorityChange(e) {

this.setData({

'criteria.location\_priority': e.detail.value

});

},

// 排除已分配开关

onExcludeAssignedChange(e) {

this.setData({

'criteria.exclude\_assigned': e.detail.value

});

},

// 开始智能匹配

startMatching() {

if (this.data.matching) return;

// 验证条件

if (!this.validateCriteria()) {

return;

}

this.setData({ matching: true });

const requestData = {

criteria: this.data.criteria,

strategy: this.data.selectedStrategy,

limit: 50

};

wx.request({

url: `${getApp().globalData.apiBase}/api/v1/intelligent-matching/smart-match`,

method: 'POST',

header: {

'Authorization': `Bearer ${wx.getStorageSync('token')}`,

'Content-Type': 'application/json'

},

data: requestData,

success: (res) => {

if (res.statusCode === 200) {

this.setData({

matchResults: res.data.matches,

totalMatches: res.data.total\_matches,

executionTime: res.data.execution\_time,

showCriteriaPanel: false

});

wx.showToast({

title: `找到${res.data.total\_matches}个匹配企业`,

icon: 'success'

});

} else {

wx.showToast({

title: res.data.detail || '匹配失败',

icon: 'error'

});

}

},

fail: () => {

wx.showToast({

title: '网络请求失败',

icon: 'error'

});

},

complete: () => {

this.setData({ matching: false });

}

});

},

// 验证匹配条件

validateCriteria() {

const { criteria } = this.data;

if (criteria.min\_customer\_count && criteria.max\_customer\_count) {

if (criteria.min\_customer\_count > criteria.max\_customer\_count) {

wx.showToast({

title: '客户数量范围有误',

icon: 'error'

});

return false;

}

}

return true;

},

// 查看匹配结果详情

viewResultDetail(e) {

const index = e.currentTarget.dataset.index;

const result = this.data.matchResults[index];

this.setData({

selectedResult: result,

showResultDetail: true

});

},

// 隐藏结果详情

hideResultDetail() {

this.setData({ showResultDetail: false });

},

// 分配企业给自己

assignToSelf(e) {

const enterpriseId = e.currentTarget.dataset.enterpriseId;

const userInfo = wx.getStorageSync('userInfo') || {};

wx.showModal({

title: '确认分配',

content: '确定要将此企业分配给自己吗？',

success: (res) => {

if (res.confirm) {

this.performAssignment(enterpriseId, userInfo.employee\_id);

}

}

});

},

// 执行分配操作

performAssignment(enterpriseId, employeeId) {

wx.request({

url: `${getApp().globalData.apiBase}/api/v1/customer-assignment/assign`,

method: 'POST',

header: {

'Authorization': `Bearer ${wx.getStorageSync('token')}`,

'Content-Type': 'application/json'

},

data: {

enterprise\_id: enterpriseId,

employee\_id: employeeId,

assignment\_type: 'smart\_match',

remarks: '智能匹配分配'

},

success: (res) => {

if (res.statusCode === 200) {

wx.showToast({

title: '分配成功',

icon: 'success'

});

// 从结果中移除已分配的企业

const updatedResults = this.data.matchResults.filter(

item => item.enterprise\_id !== enterpriseId

);

this.setData({

matchResults: updatedResults,

totalMatches: updatedResults.length,

showResultDetail: false

});

} else {

wx.showToast({

title: res.data.detail || '分配失败',

icon: 'error'

});

}

}

});

},

// 拨打电话

makeCall(e) {

const phone = e.currentTarget.dataset.phone;

if (!phone) {

wx.showToast({

title: '暂无联系电话',

icon: 'error'

});

return;

}

wx.makePhoneCall({

phoneNumber: phone

});

},

// 保存为模板

saveAsTemplate() {

wx.showModal({

title: '保存模板',

content: '请输入模板名称',

editable: true,

success: (res) => {

if (res.confirm && res.content.trim()) {

this.performSaveTemplate(res.content.trim());

}

}

});

},

// 执行保存模板

performSaveTemplate(templateName) {

wx.request({

url: `${getApp().globalData.apiBase}/api/v1/intelligent-matching/save-matching-template`,

method: 'POST',

header: {

'Authorization': `Bearer ${wx.getStorageSync('token')}`,

'Content-Type': 'application/json'

},

data: {

template\_name: templateName,

criteria: this.data.criteria

},

success: (res) => {

if (res.statusCode === 200) {

wx.showToast({

title: '模板保存成功',

icon: 'success'

});

this.loadMatchingTemplates();

} else {

wx.showToast({

title: '保存失败',

icon: 'error'

});

}

}

});

},

// 使用模板

useTemplate(e) {

const templateId = e.currentTarget.dataset.templateId;

const template = this.data.templates.find(t => t.id === templateId);

if (template) {

this.setData({

criteria: template.criteria

});

wx.showToast({

title: '模板已应用',

icon: 'success'

});

}

},

// 获取匹配建议

getMatchingSuggestions() {

const userInfo = wx.getStorageSync('userInfo') || {};

wx.request({

url: `${getApp().globalData.apiBase}/api/v1/intelligent-matching/matching-suggestions`,

header: {

'Authorization': `Bearer ${wx.getStorageSync('token')}`

},

data: {

employee\_id: userInfo.employee\_id

},

success: (res) => {

if (res.statusCode === 200) {

const suggestions = res.data.suggested\_criteria;

this.setData({

criteria: {

...this.data.criteria,

...suggestions

}

});

wx.showToast({

title: '已应用智能建议',

icon: 'success'

});

}

}

});

},

// 批量分配

batchAssign() {

if (this.data.matchResults.length === 0) {

wx.showToast({

title: '没有匹配结果',

icon: 'error'

});

return;

}

wx.showModal({

title: '批量分配',

content: `确定要将前10个匹配结果分配给自己吗？`,

success: (res) => {

if (res.confirm) {

this.performBatchAssign();

}

}

});

},

// 执行批量分配

performBatchAssign() {

const userInfo = wx.getStorageSync('userInfo') || {};

const batchResults = this.data.matchResults.slice(0, 10);

wx.request({

url: `${getApp().globalData.apiBase}/api/v1/intelligent-matching/batch-match`,

method: 'POST',

header: {

'Authorization': `Bearer ${wx.getStorageSync('token')}`,

'Content-Type': 'application/json'

},

data: {

employee\_ids: [userInfo.employee\_id],

criteria: this.data.criteria,

auto\_assign: true

},

success: (res) => {

if (res.statusCode === 200) {

const result = res.data[0];

if (result.success) {

wx.showToast({

title: `成功分配${result.auto\_assigned}个企业`,

icon: 'success'

});

// 刷新匹配结果

this.startMatching();

} else {

wx.showToast({

title: '批量分配失败',

icon: 'error'

});

}

}

}

});

}

});

智能匹配小程序页面模板

<!-- pages/intelligent-matching/index.wxml - 智能匹配页面模板 -->

<view class="container">

<!-- 页面标题 -->

<view class="page-header">

<text class="title">智能匹配</text>

<text class="subtitle">AI推荐优质企业客户</text>

</view>

<!-- 快速操作按钮 -->

<view class="quick-actions">

<button class="action-btn primary" bindtap="showCriteriaPanel">

<text class="icon">⚙️</text>

<text>设置条件</text>

</button>

<button class="action-btn" bindtap="getMatchingSuggestions">

<text class="icon">💡</text>

<text>智能建议</text>

</button>

<button class="action-btn" bindtap="saveAsTemplate" wx:if="{{matchResults.length > 0}}">

<text class="icon">💾</text>

<text>保存模板</text>

</button>

</view>

<!-- 匹配策略选择 -->

<view class="strategy-section">

<view class="section-title">

<text class="title">匹配策略</text>

<text class="desc">选择匹配模式</text>

</view>

<view class="strategy-tabs">

<view class="tab-item {{activeStrategy === 'precise' ? 'active' : ''}}"

data-strategy="precise" bindtap="switchStrategy">

<text class="tab-text">精准匹配</text>

</view>

<view class="tab-item {{activeStrategy === 'fuzzy' ? 'active' : ''}}"

data-strategy="fuzzy" bindtap="switchStrategy">

<text class="tab-text">模糊匹配</text>

</view>

<view class="tab-item {{activeStrategy === 'ai' ? 'active' : ''}}"

data-strategy="ai" bindtap="switchStrategy">

<text class="tab-text">AI推荐</text>

</view>

</view>

</view>

<!-- 匹配条件面板 -->

<view class="criteria-panel {{showCriteria ? 'show' : ''}}" bindtap="hideCriteriaPanel">

<view class="panel-content" catchtap="stopPropagation">

<view class="panel-header">

<text class="panel-title">匹配条件设置</text>

<text class="close-btn" bindtap="hideCriteriaPanel">×</text>

</view>

<scroll-view class="criteria-form" scroll-y>

<!-- 行业类型 -->

<view class="form-group">

<text class="label">行业类型</text>

<picker mode="multiSelector" range="{{industryOptions}}" range-key="name"

value="{{selectedIndustries}}" bindchange="onIndustryChange">

<view class="picker-display">

<text class="placeholder" wx:if="{{selectedIndustries.length === 0}}">请选择行业</text>

<text wx:else>已选择 {{selectedIndustries.length}} 个行业</text>

</view>

</picker>

</view>

<!-- 地区选择 -->

<view class="form-group">

<text class="label">地区范围</text>

<picker mode="region" value="{{selectedRegion}}" bindchange="onRegionChange">

<view class="picker-display">

<text class="placeholder" wx:if="{{selectedRegion.length === 0}}">请选择地区</text>

<text wx:else>{{selectedRegion.join(' - ')}}</text>

</view>

</picker>

</view>

<!-- 企业规模 -->

<view class="form-group">

<text class="label">企业规模</text>

<view class="checkbox-group">

<view class="checkbox-item" wx:for="{{scaleOptions}}" wx:key="value"

data-value="{{item.value}}" bindtap="toggleScale">

<view class="checkbox {{selectedScales.includes(item.value) ? 'checked' : ''}}">

<text class="checkmark" wx:if="{{selectedScales.includes(item.value)}}">✓</text>

</view>

<text class="checkbox-label">{{item.name}}</text>

</view>

</view>

</view>

<!-- 用电量范围 -->

<view class="form-group">

<text class="label">用电量范围 (万kWh/年)</text>

<view class="range-input">

<input type="number" placeholder="最小值" value="{{minPowerUsage}}"

bindinput="onMinPowerUsageInput" />

<text class="range-separator">-</text>

<input type="number" placeholder="最大值" value="{{maxPowerUsage}}"

bindinput="onMaxPowerUsageInput" />

</view>

</view>

<!-- 匹配权重 -->

<view class="form-group">

<text class="label">匹配权重</text>

<view class="weight-sliders">

<view class="slider-item">

<text class="slider-label">行业匹配: {{industryWeight}}%</text>

<slider value="{{industryWeight}}" min="0" max="100"

bindchange="onIndustryWeightChange" />

</view>

<view class="slider-item">

<text class="slider-label">地域匹配: {{regionWeight}}%</text>

<slider value="{{regionWeight}}" min="0" max="100"

bindchange="onRegionWeightChange" />

</view>

<view class="slider-item">

<text class="slider-label">规模匹配: {{scaleWeight}}%</text>

<slider value="{{scaleWeight}}" min="0" max="100"

bindchange="onScaleWeightChange" />

</view>

</view>

</view>

</scroll-view>

<view class="panel-footer">

<button class="btn-secondary" bindtap="resetCriteria">重置</button>

<button class="btn-primary" bindtap="applyCriteria">应用</button>

</view>

</view>

</view>

<!-- 匹配结果 -->

<view class="results-section" wx:if="{{matchResults.length > 0}}">

<view class="section-header">

<text class="title">匹配结果</text>

<text class="count">共找到 {{matchResults.length}} 个潜在客户</text>

</view>

<!-- 结果排序 -->

<view class="sort-bar">

<text class="sort-label">排序方式:</text>

<picker range="{{sortOptions}}" range-key="name" value="{{sortIndex}}"

bindchange="onSortChange">

<view class="sort-picker">

<text>{{sortOptions[sortIndex].name}}</text>

<text class="arrow">▼</text>

</view>

</picker>

</view>

<!-- 结果列表 -->

<scroll-view class="results-list" scroll-y>

<view class="result-item" wx:for="{{matchResults}}" wx:key="id"

bindtap="viewCompanyDetail" data-company="{{item}}">

<view class="company-info">

<view class="company-header">

<text class="company-name">{{item.companyName}}</text>

<view class="match-score">

<text class="score-text">匹配度</text>

<text class="score-value">{{item.matchScore}}%</text>

</view>

</view>

<view class="company-details">

<text class="detail-item">{{item.industry}}</text>

<text class="detail-separator">|</text>

<text class="detail-item">{{item.region}}</text>

<text class="detail-separator">|</text>

<text class="detail-item">{{item.scale}}</text>

</view>

<view class="company-stats">

<view class="stat-item">

<text class="stat-label">年用电量</text>

<text class="stat-value">{{item.powerUsage}}万kWh</text>

</view>

<view class="stat-item">

<text class="stat-label">潜在价值</text>

<text class="stat-value">{{item.potentialValue}}万元</text>

</view>

</view>

<view class="match-reasons">

<text class="reasons-title">匹配原因:</text>

<text class="reason-item" wx:for="{{item.matchReasons}}" wx:key="\*this">

{{item}}

</text>

</view>

</view>

<view class="action-buttons">

<button class="btn-outline" bindtap="addToFollowList"

data-company="{{item}}" catchtap="stopPropagation">

<text class="icon">📋</text>

<text>加入跟进</text>

</button>

<button class="btn-primary" bindtap="contactCompany"

data-company="{{item}}" catchtap="stopPropagation">

<text class="icon">📞</text>

<text>联系客户</text>

</button>

</view>

</view>

</scroll-view>

</view>

<!-- 空状态 -->

<view class="empty-state" wx:if="{{!loading && matchResults.length === 0 && hasSearched}}">

<image class="empty-icon" src="/images/empty-search.png" mode="aspectFit" />

<text class="empty-title">暂无匹配结果</text>

<text class="empty-desc">尝试调整匹配条件或使用其他策略</text>

<button class="btn-primary" bindtap="showCriteriaPanel">调整条件</button>

</view>

<!-- 加载状态 -->

<view class="loading-state" wx:if="{{loading}}">

<view class="loading-spinner"></view>

<text class="loading-text">AI正在为您匹配客户...</text>

</view>

<!-- 智能建议面板 -->

<view class="suggestions-panel {{showSuggestions ? 'show' : ''}}" bindtap="hideSuggestionsPanel">

<view class="panel-content" catchtap="stopPropagation">

<view class="panel-header">

<text class="panel-title">智能建议</text>

<text class="close-btn" bindtap="hideSuggestionsPanel">×</text>

</view>

<scroll-view class="suggestions-content" scroll-y>

<view class="suggestion-item" wx:for="{{suggestions}}" wx:key="id">

<view class="suggestion-header">

<text class="suggestion-title">{{item.title}}</text>

<text class="suggestion-type">{{item.type}}</text>

</view>

<text class="suggestion-desc">{{item.description}}</text>

<button class="btn-small" bindtap="applySuggestion" data-suggestion="{{item}}">

应用建议

</button>

</view>

</scroll-view>

</view>

</view>

<!-- 保存模板面板 -->

<view class="template-panel {{showTemplate ? 'show' : ''}}" bindtap="hideTemplatePanel">

<view class="panel-content" catchtap="stopPropagation">

<view class="panel-header">

<text class="panel-title">保存为模板</text>

<text class="close-btn" bindtap="hideTemplatePanel">×</text>

</view>

<view class="template-form">

<view class="form-group">

<text class="label">模板名称</text>

<input class="input" placeholder="请输入模板名称" value="{{templateName}}"

bindinput="onTemplateNameInput" />

</view>

<view class="form-group">

<text class="label">模板描述</text>

<textarea class="textarea" placeholder="请输入模板描述" value="{{templateDesc}}"

bindinput="onTemplateDescInput"></textarea>

</view>

</view>

<view class="panel-footer">

<button class="btn-secondary" bindtap="hideTemplatePanel">取消</button>

<button class="btn-primary" bindtap="saveTemplate">保存</button>

</view>

</view>

</view>

</view>

现在继续开发下一个模块：**消息通知系统** (站内信、短信、微信通知)。这个系统包含后端API和前端管理界面。

消息通知系统 API 后端 - notification\_system.py

"""

消息通知系统 API

支持站内信、短信、微信通知的统一管理

"""

from fastapi import APIRouter, Depends, HTTPException, BackgroundTasks, Query

from sqlalchemy.orm import Session

from sqlalchemy import and\_, or\_, desc, func

from typing import List, Optional, Dict, Any

from datetime import datetime, timedelta

from database import get\_db

from models.notification import (

Notification, NotificationTemplate, NotificationLog,

NotificationSetting, NotificationQueue

)

from models.user import User

from schemas.notification import (

NotificationCreate, NotificationUpdate, NotificationResponse,

NotificationTemplateCreate, NotificationTemplateUpdate, NotificationTemplateResponse,

NotificationSettingUpdate, NotificationSettingResponse,

NotificationStatsResponse, NotificationBatchCreate

)

from auth import get\_current\_user, require\_permissions

from services.notification\_service import NotificationService

from utils.pagination import paginate

import logging

logger = logging.getLogger(\_\_name\_\_)

router = APIRouter(prefix="/api/v1/notifications", tags=["notifications"])

# 初始化通知服务

notification\_service = NotificationService()

@router.post("/send", response\_model=Dict[str, Any])

async def send\_notification(

notification: NotificationCreate,

background\_tasks: BackgroundTasks,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""发送单个通知"""

try:

# 创建通知记录

db\_notification = Notification(

user\_id=notification.user\_id,

title=notification.title,

content=notification.content,

type=notification.type,

channel=notification.channel,

priority=notification.priority,

template\_id=notification.template\_id,

extra\_data=notification.extra\_data,

created\_by=current\_user.id

)

db.add(db\_notification)

db.commit()

db.refresh(db\_notification)

# 添加到发送队列

background\_tasks.add\_task(

notification\_service.send\_notification\_async,

db\_notification.id,

db

)

return {

"success": True,

"message": "通知发送请求已提交",

"notification\_id": db\_notification.id

}

except Exception as e:

logger.error(f"发送通知失败: {str(e)}")

raise HTTPException(status\_code=500, detail="发送通知失败")

@router.post("/send-batch", response\_model=Dict[str, Any])

async def send\_batch\_notifications(

batch\_notification: NotificationBatchCreate,

background\_tasks: BackgroundTasks,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""批量发送通知"""

try:

notification\_ids = []

# 获取目标用户

target\_users = []

if batch\_notification.user\_ids:

target\_users = db.query(User).filter(

User.id.in\_(batch\_notification.user\_ids)

).all()

elif batch\_notification.user\_filter:

# 根据过滤条件获取用户

query = db.query(User)

if batch\_notification.user\_filter.get("role"):

query = query.filter(User.role == batch\_notification.user\_filter["role"])

if batch\_notification.user\_filter.get("department\_id"):

query = query.filter(User.department\_id == batch\_notification.user\_filter["department\_id"])

if batch\_notification.user\_filter.get("status"):

query = query.filter(User.status == batch\_notification.user\_filter["status"])

target\_users = query.all()

# 创建通知记录

for user in target\_users:

db\_notification = Notification(

user\_id=user.id,

title=batch\_notification.title,

content=batch\_notification.content,

type=batch\_notification.type,

channel=batch\_notification.channel,

priority=batch\_notification.priority,

template\_id=batch\_notification.template\_id,

extra\_data=batch\_notification.extra\_data,

created\_by=current\_user.id

)

db.add(db\_notification)

db.flush()

notification\_ids.append(db\_notification.id)

db.commit()

# 添加到发送队列

for notification\_id in notification\_ids:

background\_tasks.add\_task(

notification\_service.send\_notification\_async,

notification\_id,

db

)

return {

"success": True,

"message": f"批量通知发送请求已提交，共 {len(notification\_ids)} 条",

"notification\_count": len(notification\_ids)

}

except Exception as e:

logger.error(f"批量发送通知失败: {str(e)}")

raise HTTPException(status\_code=500, detail="批量发送通知失败")

@router.get("/", response\_model=Dict[str, Any])

def get\_notifications(

page: int = Query(1, ge=1),

limit: int = Query(20, ge=1, le=100),

user\_id: Optional[int] = None,

type: Optional[str] = None,

channel: Optional[str] = None,

status: Optional[str] = None,

priority: Optional[str] = None,

start\_date: Optional[datetime] = None,

end\_date: Optional[datetime] = None,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""获取通知列表"""

query = db.query(Notification)

# 权限过滤

if current\_user.role not in ["admin", "manager"]:

query = query.filter(Notification.user\_id == current\_user.id)

elif user\_id:

query = query.filter(Notification.user\_id == user\_id)

# 条件过滤

if type:

query = query.filter(Notification.type == type)

if channel:

query = query.filter(Notification.channel == channel)

if status:

query = query.filter(Notification.status == status)

if priority:

query = query.filter(Notification.priority == priority)

if start\_date:

query = query.filter(Notification.created\_at >= start\_date)

if end\_date:

query = query.filter(Notification.created\_at <= end\_date)

# 排序

query = query.order\_by(desc(Notification.created\_at))

return paginate(query, page, limit)

@router.get("/{notification\_id}", response\_model=NotificationResponse)

def get\_notification(

notification\_id: int,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""获取通知详情"""

notification = db.query(Notification).filter(

Notification.id == notification\_id

).first()

if not notification:

raise HTTPException(status\_code=404, detail="通知不存在")

# 权限检查

if (current\_user.role not in ["admin", "manager"] and

notification.user\_id != current\_user.id):

raise HTTPException(status\_code=403, detail="无权限访问此通知")

return notification

@router.put("/{notification\_id}/read", response\_model=Dict[str, str])

def mark\_as\_read(

notification\_id: int,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""标记通知为已读"""

notification = db.query(Notification).filter(

and\_(

Notification.id == notification\_id,

Notification.user\_id == current\_user.id

)

).first()

if not notification:

raise HTTPException(status\_code=404, detail="通知不存在")

notification.read\_at = datetime.utcnow()

notification.status = "read"

db.commit()

return {"message": "通知已标记为已读"}

@router.put("/read-all", response\_model=Dict[str, str])

def mark\_all\_as\_read(

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""标记所有通知为已读"""

db.query(Notification).filter(

and\_(

Notification.user\_id == current\_user.id,

Notification.status == "unread"

)

).update({

"read\_at": datetime.utcnow(),

"status": "read"

})

db.commit()

return {"message": "所有通知已标记为已读"}

@router.delete("/{notification\_id}")

def delete\_notification(

notification\_id: int,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""删除通知"""

notification = db.query(Notification).filter(

Notification.id == notification\_id

).first()

if not notification:

raise HTTPException(status\_code=404, detail="通知不存在")

# 权限检查

if (current\_user.role not in ["admin", "manager"] and

notification.user\_id != current\_user.id):

raise HTTPException(status\_code=403, detail="无权限删除此通知")

db.delete(notification)

db.commit()

return {"message": "通知已删除"}

@router.get("/stats/overview", response\_model=NotificationStatsResponse)

def get\_notification\_stats(

days: int = Query(30, ge=1, le=365),

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""获取通知统计信息"""

start\_date = datetime.utcnow() - timedelta(days=days)

query = db.query(Notification).filter(

Notification.created\_at >= start\_date

)

# 权限过滤

if current\_user.role not in ["admin", "manager"]:

query = query.filter(Notification.user\_id == current\_user.id)

# 基础统计

total\_count = query.count()

sent\_count = query.filter(Notification.status == "sent").count()

failed\_count = query.filter(Notification.status == "failed").count()

pending\_count = query.filter(Notification.status == "pending").count()

# 按渠道统计

channel\_stats = db.query(

Notification.channel,

func.count(Notification.id).label('count')

).filter(

Notification.created\_at >= start\_date

).group\_by(Notification.channel).all()

# 按类型统计

type\_stats = db.query(

Notification.type,

func.count(Notification.id).label('count')

).filter(

Notification.created\_at >= start\_date

).group\_by(Notification.type).all()

# 每日发送量统计

daily\_stats = db.query(

func.date(Notification.created\_at).label('date'),

func.count(Notification.id).label('count')

).filter(

Notification.created\_at >= start\_date

).group\_by(func.date(Notification.created\_at)).all()

return {

"total\_count": total\_count,

"sent\_count": sent\_count,

"failed\_count": failed\_count,

"pending\_count": pending\_count,

"success\_rate": round(sent\_count / total\_count \* 100, 2) if total\_count > 0 else 0,

"channel\_stats": [{"channel": item[0], "count": item[1]} for item in channel\_stats],

"type\_stats": [{"type": item[0], "count": item[1]} for item in type\_stats],

"daily\_stats": [{"date": item[0].isoformat(), "count": item[1]} for item in daily\_stats]

}

# 通知模板管理

@router.post("/templates", response\_model=NotificationTemplateResponse)

@require\_permissions(["admin", "notification\_manager"])

def create\_template(

template: NotificationTemplateCreate,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""创建通知模板"""

db\_template = NotificationTemplate(

name=template.name,

type=template.type,

channel=template.channel,

title\_template=template.title\_template,

content\_template=template.content\_template,

variables=template.variables,

is\_active=template.is\_active,

created\_by=current\_user.id

)

db.add(db\_template)

db.commit()

db.refresh(db\_template)

return db\_template

@router.get("/templates", response\_model=Dict[str, Any])

def get\_templates(

page: int = Query(1, ge=1),

limit: int = Query(20, ge=1, le=100),

type: Optional[str] = None,

channel: Optional[str] = None,

is\_active: Optional[bool] = None,

db: Session = Depends(get\_db)

):

"""获取通知模板列表"""

query = db.query(NotificationTemplate)

if type:

query = query.filter(NotificationTemplate.type == type)

if channel:

query = query.filter(NotificationTemplate.channel == channel)

if is\_active is not None:

query = query.filter(NotificationTemplate.is\_active == is\_active)

query = query.order\_by(desc(NotificationTemplate.created\_at))

return paginate(query, page, limit)

@router.put("/templates/{template\_id}", response\_model=NotificationTemplateResponse)

@require\_permissions(["admin", "notification\_manager"])

def update\_template(

template\_id: int,

template: NotificationTemplateUpdate,

db: Session = Depends(get\_db)

):

"""更新通知模板"""

db\_template = db.query(NotificationTemplate).filter(

NotificationTemplate.id == template\_id

).first()

if not db\_template:

raise HTTPException(status\_code=404, detail="模板不存在")

update\_data = template.dict(exclude\_unset=True)

for field, value in update\_data.items():

setattr(db\_template, field, value)

db.commit()

db.refresh(db\_template)

return db\_template

@router.delete("/templates/{template\_id}")

@require\_permissions(["admin", "notification\_manager"])

def delete\_template(

template\_id: int,

db: Session = Depends(get\_db)

):

"""删除通知模板"""

db\_template = db.query(NotificationTemplate).filter(

NotificationTemplate.id == template\_id

).first()

if not db\_template:

raise HTTPException(status\_code=404, detail="模板不存在")

db.delete(db\_template)

db.commit()

return {"message": "模板已删除"}

# 通知设置管理

@router.get("/settings", response\_model=List[NotificationSettingResponse])

def get\_notification\_settings(

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""获取用户通知设置"""

settings = db.query(NotificationSetting).filter(

NotificationSetting.user\_id == current\_user.id

).all()

# 如果没有设置，创建默认设置

if not settings:

default\_settings = [

{"type": "order", "channel": "system", "enabled": True},

{"type": "order", "channel": "sms", "enabled": True},

{"type": "order", "channel": "wechat", "enabled": False},

{"type": "marketing", "channel": "system", "enabled": True},

{"type": "marketing", "channel": "sms", "enabled": False},

{"type": "marketing", "channel": "wechat", "enabled": False},

{"type": "system", "channel": "system", "enabled": True},

{"type": "system", "channel": "sms", "enabled": False},

{"type": "system", "channel": "wechat", "enabled": False},

]

for setting in default\_settings:

db\_setting = NotificationSetting(

user\_id=current\_user.id,

\*\*setting

)

db.add(db\_setting)

settings.append(db\_setting)

db.commit()

return settings

@router.put("/settings", response\_model=List[NotificationSettingResponse])

def update\_notification\_settings(

settings: List[NotificationSettingUpdate],

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""更新用户通知设置"""

for setting\_update in settings:

# 查找或创建设置

db\_setting = db.query(NotificationSetting).filter(

and\_(

NotificationSetting.user\_id == current\_user.id,

NotificationSetting.type == setting\_update.type,

NotificationSetting.channel == setting\_update.channel

)

).first()

if db\_setting:

# 更新现有设置

db\_setting.enabled = setting\_update.enabled

if hasattr(setting\_update, 'schedule'):

db\_setting.schedule = setting\_update.schedule

else:

# 创建新设置

db\_setting = NotificationSetting(

user\_id=current\_user.id,

type=setting\_update.type,

channel=setting\_update.channel,

enabled=setting\_update.enabled,

schedule=getattr(setting\_update, 'schedule', None)

)

db.add(db\_setting)

db.commit()

# 返回更新后的设置

updated\_settings = db.query(NotificationSetting).filter(

NotificationSetting.user\_id == current\_user.id

).all()

return updated\_settings

@router.get("/logs", response\_model=Dict[str, Any])

@require\_permissions(["admin", "notification\_manager"])

def get\_notification\_logs(

page: int = Query(1, ge=1),

limit: int = Query(20, ge=1, le=100),

notification\_id: Optional[int] = None,

status: Optional[str] = None,

channel: Optional[str] = None,

start\_date: Optional[datetime] = None,

end\_date: Optional[datetime] = None,

db: Session = Depends(get\_db)

):

"""获取通知发送日志"""

query = db.query(NotificationLog)

if notification\_id:

query = query.filter(NotificationLog.notification\_id == notification\_id)

if status:

query = query.filter(NotificationLog.status == status)

if channel:

query = query.filter(NotificationLog.channel == channel)

if start\_date:

query = query.filter(NotificationLog.sent\_at >= start\_date)

if end\_date:

query = query.filter(NotificationLog.sent\_at <= end\_date)

query = query.order\_by(desc(NotificationLog.sent\_at))

return paginate(query, page, limit)

@router.post("/test-send", response\_model=Dict[str, Any])

@require\_permissions(["admin", "notification\_manager"])

def test\_send\_notification(

test\_data: Dict[str, Any],

background\_tasks: BackgroundTasks,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""测试发送通知"""

try:

# 创建测试通知

test\_notification = Notification(

user\_id=test\_data.get("user\_id", current\_user.id),

title=test\_data.get("title", "测试通知"),

content=test\_data.get("content", "这是一条测试通知"),

type="system",

channel=test\_data.get("channel", "system"),

priority="normal",

created\_by=current\_user.id

)

db.add(test\_notification)

db.commit()

db.refresh(test\_notification)

# 立即发送测试通知

background\_tasks.add\_task(

notification\_service.send\_notification\_async,

test\_notification.id,

db

)

return {

"success": True,

"message": "测试通知已发送",

"notification\_id": test\_notification.id

}

except Exception as e:

logger.error(f"测试发送通知失败: {str(e)}")

raise HTTPException(status\_code=500, detail="测试发送失败")

@router.get("/queue/status", response\_model=Dict[str, Any])

@require\_permissions(["admin", "notification\_manager"])

def get\_queue\_status(db: Session = Depends(get\_db)):

"""获取通知队列状态"""

try:

# 获取队列统计

pending\_count = db.query(NotificationQueue).filter(

NotificationQueue.status == "pending"

).count()

processing\_count = db.query(NotificationQueue).filter(

NotificationQueue.status == "processing"

).count()

failed\_count = db.query(NotificationQueue).filter(

and\_(

NotificationQueue.status == "failed",

NotificationQueue.updated\_at >= datetime.utcnow() - timedelta(hours=24)

)

).count()

# 获取最近的队列任务

recent\_tasks = db.query(NotificationQueue).order\_by(

desc(NotificationQueue.updated\_at)

).limit(10).all()

return {

"queue\_stats": {

"pending\_count": pending\_count,

"processing\_count": processing\_count,

"failed\_count\_24h": failed\_count

},

"recent\_tasks": [

{

"id": task.id,

"notification\_id": task.notification\_id,

"status": task.status,

"retry\_count": task.retry\_count,

"updated\_at": task.updated\_at.isoformat()

}

for task in recent\_tasks

]

}

except Exception as e:

logger.error(f"获取队列状态失败: {str(e)}")

raise HTTPException(status\_code=500, detail="获取队列状态失败")

@router.post("/queue/retry", response\_model=Dict[str, str])

@require\_permissions(["admin", "notification\_manager"])

def retry\_failed\_notifications(

background\_tasks: BackgroundTasks,

db: Session = Depends(get\_db)

):

"""重试失败的通知"""

try:

# 获取失败的通知队列任务

failed\_tasks = db.query(NotificationQueue).filter(

and\_(

NotificationQueue.status == "failed",

NotificationQueue.retry\_count < 3 # 最多重试3次

)

).all()

retry\_count = 0

for task in failed\_tasks:

# 重置状态为pending

task.status = "pending"

task.retry\_count += 1

task.error\_message = None

task.updated\_at = datetime.utcnow()

# 添加到后台任务

background\_tasks.add\_task(

notification\_service.send\_notification\_async,

task.notification\_id,

db

)

retry\_count += 1

db.commit()

return {"message": f"已重试 {retry\_count} 个失败的通知任务"}

except Exception as e:

logger.error(f"重试失败通知出错: {str(e)}")

raise HTTPException(status\_code=500, detail="重试失败")

@router.delete("/cleanup", response\_model=Dict[str, str])

@require\_permissions(["admin"])

def cleanup\_old\_notifications(

days: int = Query(30, ge=7, le=365),

db: Session = Depends(get\_db)

):

"""清理旧通知数据"""

try:

cutoff\_date = datetime.utcnow() - timedelta(days=days)

# 删除旧的通知记录

deleted\_notifications = db.query(Notification).filter(

and\_(

Notification.created\_at < cutoff\_date,

Notification.status.in\_(["sent", "failed"])

)

).delete()

# 删除旧的发送日志

deleted\_logs = db.query(NotificationLog).filter(

NotificationLog.sent\_at < cutoff\_date

).delete()

# 删除已完成的队列任务

deleted\_queue = db.query(NotificationQueue).filter(

and\_(

NotificationQueue.updated\_at < cutoff\_date,

NotificationQueue.status.in\_(["completed", "failed"])

)

).delete()

db.commit()

return {

"message": f"清理完成：删除了 {deleted\_notifications} 条通知记录，"

f"{deleted\_logs} 条发送日志，{deleted\_queue} 条队列任务"

}

except Exception as e:

logger.error(f"清理旧通知数据失败: {str(e)}")

raise HTTPException(status\_code=500, detail="清理失败")

# WebSocket 实时通知

@router.websocket("/ws/{user\_id}")

async def websocket\_notifications(

websocket: WebSocket,

user\_id: int,

db: Session = Depends(get\_db)

):

"""WebSocket实时通知推送"""

await notification\_service.handle\_websocket\_connection(websocket, user\_id, db)

现在继续开发下一个模块：**通知服务类 (NotificationService)**

通知服务类 - services/notification\_service.py

"""

通知服务类

处理各种通知的发送逻辑

"""

import asyncio

import json

import logging

from datetime import datetime

from typing import Dict, List, Optional, Any

from sqlalchemy.orm import Session

from sqlalchemy import and\_

from fastapi import WebSocket, WebSocketDisconnect

import aiohttp

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

from jinja2 import Template

import requests

from models.notification import (

Notification, NotificationTemplate, NotificationLog,

NotificationSetting, NotificationQueue

)

from models.user import User

from config import settings

logger = logging.getLogger(\_\_name\_\_)

class NotificationService:

def \_\_init\_\_(self):

self.active\_connections: Dict[int, List[WebSocket]] = {}

async def send\_notification\_async(self, notification\_id: int, db: Session):

"""异步发送通知"""

try:

# 获取通知信息

notification = db.query(Notification).filter(

Notification.id == notification\_id

).first()

if not notification:

logger.error(f"通知 {notification\_id} 不存在")

return

# 获取用户信息

user = db.query(User).filter(User.id == notification.user\_id).first()

if not user:

logger.error(f"用户 {notification.user\_id} 不存在")

return

# 检查用户通知设置

setting = db.query(NotificationSetting).filter(

and\_(

NotificationSetting.user\_id == user.id,

NotificationSetting.type == notification.type,

NotificationSetting.channel == notification.channel

)

).first()

if setting and not setting.enabled:

logger.info(f"用户 {user.id} 已关闭 {notification.type} - {notification.channel} 通知")

notification.status = "skipped"

db.commit()

return

# 创建队列任务

queue\_task = NotificationQueue(

notification\_id=notification\_id,

status="processing",

retry\_count=0

)

db.add(queue\_task)

db.commit()

# 根据渠道发送通知

success = False

error\_message = None

try:

if notification.channel == "system":

success = await self.\_send\_system\_notification(notification, user, db)

elif notification.channel == "sms":

success = await self.\_send\_sms\_notification(notification, user, db)

elif notification.channel == "wechat":

success = await self.\_send\_wechat\_notification(notification, user, db)

elif notification.channel == "email":

success = await self.\_send\_email\_notification(notification, user, db)

else:

error\_message = f"不支持的通知渠道: {notification.channel}"

except Exception as e:

error\_message = str(e)

logger.error(f"发送通知失败: {error\_message}")

# 更新通知状态

if success:

notification.status = "sent"

notification.sent\_at = datetime.utcnow()

queue\_task.status = "completed"

else:

notification.status = "failed"

queue\_task.status = "failed"

queue\_task.error\_message = error\_message

# 记录发送日志

log = NotificationLog(

notification\_id=notification\_id,

channel=notification.channel,

status="sent" if success else "failed",

sent\_at=datetime.utcnow(),

error\_message=error\_message

)

db.add(log)

db.commit()

except Exception as e:

logger.error(f"处理通知 {notification\_id} 时出错: {str(e)}")

async def \_send\_system\_notification(self, notification: Notification, user: User, db: Session) -> bool:

"""发送站内信通知"""

try:

# 站内信直接存储在数据库中，状态设为已发送

# 如果有WebSocket连接，实时推送

await self.\_push\_websocket\_notification(user.id, {

"id": notification.id,

"title": notification.title,

"content": notification.content,

"type": notification.type,

"created\_at": notification.created\_at.isoformat()

})

return True

except Exception as e:

logger.error(f"发送站内信失败: {str(e)}")

return False

async def \_send\_sms\_notification(self, notification: Notification, user: User, db: Session) -> bool:

"""发送短信通知"""

if not user.phone:

logger.warning(f"用户 {user.id} 没有手机号，无法发送短信")

return False

try:

# 这里需要集成短信服务商API，如阿里云短信、腾讯云短信等

# 示例使用假设的短信API

sms\_content = await self.\_render\_template(

notification.content,

notification.extra\_data or {}

)

# 调用短信API

response = await self.\_call\_sms\_api(

phone=user.phone,

content=sms\_content,

template\_code=getattr(notification, 'sms\_template\_code', None)

)

return response.get('success', False)

except Exception as e:

logger.error(f"发送短信失败: {str(e)}")

return False

async def \_send\_wechat\_notification(self, notification: Notification, user: User, db: Session) -> bool:

"""发送微信通知"""

if not user.openid:

logger.warning(f"用户 {user.id} 没有openid，无法发送微信通知")

return False

try:

# 发送微信模板消息

access\_token = await self.\_get\_wechat\_access\_token()

template\_data = {

"first": {"value": notification.title},

"keyword1": {"value": notification.content},

"keyword2": {"value": notification.created\_at.strftime("%Y-%m-%d %H:%M")},

"remark": {"value": "点击查看详情"}

}

# 如果有额外数据，添加到模板中

if notification.extra\_data:

template\_data.update(notification.extra\_data)

response = await self.\_send\_wechat\_template\_message(

access\_token=access\_token,

openid=user.openid,

template\_id=getattr(notification, 'wechat\_template\_id', settings.WECHAT\_DEFAULT\_TEMPLATE\_ID),

data=template\_data,

url=getattr(notification, 'jump\_url', None)

)

return response.get('errcode') == 0

except Exception as e:

logger.error(f"发送微信通知失败: {str(e)}")

return False

async def \_send\_email\_notification(self, notification: Notification, user: User, db: Session) -> bool:

"""发送邮件通知"""

if not user.email:

logger.warning(f"用户 {user.id} 没有邮箱，无法发送邮件")

return False

try:

# 渲染邮件内容

subject = await self.\_render\_template(

notification.title,

notification.extra\_data or {}

)

content = await self.\_render\_template(

notification.content,

notification.extra\_data or {}

)

# 发送邮件

msg = MIMEMultipart()

msg['From'] = settings.SMTP\_FROM\_EMAIL

msg['To'] = user.email

msg['Subject'] = subject

msg.attach(MIMEText(content, 'html', 'utf-8'))

with smtplib.SMTP(settings.SMTP\_HOST, settings.SMTP\_PORT) as server:

if settings.SMTP\_USE\_TLS:

server.starttls()

server.login(settings.SMTP\_USERNAME, settings.SMTP\_PASSWORD)

server.send\_message(msg)

return True

except Exception as e:

logger.error(f"发送邮件失败: {str(e)}")

return False

async def \_render\_template(self, template\_str: str, data: Dict[str, Any]) -> str:

"""渲染模板"""

try:

template = Template(template\_str)

return template.render(\*\*data)

except Exception as e:

logger.error(f"渲染模板失败: {str(e)}")

return template\_str

async def \_call\_sms\_api(self, phone: str, content: str, template\_code: Optional[str] = None) -> Dict[str, Any]:

"""调用短信API"""

# 这里是示例实现，需要根据实际短信服务商进行调整

try:

async with aiohttp.ClientSession() as session:

payload = {

"phone": phone,

"content": content,

"template\_code": template\_code or settings.SMS\_DEFAULT\_TEMPLATE\_CODE

}

async with session.post(

settings.SMS\_API\_URL,

json=payload,

headers={"Authorization": f"Bearer {settings.SMS\_API\_KEY}"}

) as response:

return await response.json()

except Exception as e:

logger.error(f"调用短信API失败: {str(e)}")

return {"success": False, "error": str(e)}

async def \_get\_wechat\_access\_token(self) -> str:

"""获取微信访问令牌"""

try:

async with aiohttp.ClientSession() as session:

url = f"https://api.weixin.qq.com/cgi-bin/token"

params = {

"grant\_type": "client\_credential",

"appid": settings.WECHAT\_APP\_ID,

"secret": settings.WECHAT\_APP\_SECRET

}

async with session.get(url, params=params) as response:

data = await response.json()

return data.get("access\_token", "")

except Exception as e:

logger.error(f"获取微信访问令牌失败: {str(e)}")

return ""

async def \_send\_wechat\_template\_message(

self,

access\_token: str,

openid: str,

template\_id: str,

data: Dict[str, Any],

url: Optional[str] = None

) -> Dict[str, Any]:

"""发送微信模板消息"""

try:

async with aiohttp.ClientSession() as session:

api\_url = f"https://api.weixin.qq.com/cgi-bin/message/template/send?access\_token={access\_token}"

payload = {

"touser": openid,

"template\_id": template\_id,

"data": data

}

if url:

payload["url"] = url

async with session.post(api\_url, json=payload) as response:

return await response.json()

except Exception as e:

logger.error(f"发送微信模板消息失败: {str(e)}")

return {"errcode": -1, "errmsg": str(e)}

async def \_push\_websocket\_notification(self, user\_id: int, notification\_data: Dict[str, Any]):

"""通过WebSocket推送实时通知"""

if user\_id in self.active\_connections:

# 向该用户的所有连接推送通知

connections\_to\_remove = []

for websocket in self.active\_connections[user\_id]:

try:

await websocket.send\_json({

"type": "notification",

"data": notification\_data

})

except Exception as e:

logger.warning(f"WebSocket连接已断开: {str(e)}")

connections\_to\_remove.append(websocket)

# 清理断开的连接

for websocket in connections\_to\_remove:

self.active\_connections[user\_id].remove(websocket)

# 如果用户没有活跃连接，清理记录

if not self.active\_connections[user\_id]:

del self.active\_connections[user\_id]

async def handle\_websocket\_connection(self, websocket: WebSocket, user\_id: int, db: Session):

"""处理WebSocket连接"""

await websocket.accept()

# 添加到活跃连接

if user\_id not in self.active\_connections:

self.active\_connections[user\_id] = []

self.active\_connections[user\_id].append(websocket)

try:

# 发送未读通知

unread\_notifications = db.query(Notification).filter(

and\_(

Notification.user\_id == user\_id,

Notification.status == "sent",

Notification.read\_at.is\_(None)

)

).order\_by(Notification.created\_at.desc()).limit(10).all()

for notification in unread\_notifications:

await websocket.send\_json({

"type": "unread\_notification",

"data": {

"id": notification.id,

"title": notification.title,

"content": notification.content,

"type": notification.type,

"created\_at": notification.created\_at.isoformat()

}

})

# 保持连接

while True:

try:

# 等待客户端消息

message = await websocket.receive\_json()

# 处理客户端消息

if message.get("type") == "mark\_read":

notification\_id = message.get("notification\_id")

if notification\_id:

# 标记通知为已读

notification = db.query(Notification).filter(

and\_(

Notification.id == notification\_id,

Notification.user\_id == user\_id

)

).first()

if notification:

notification.read\_at = datetime.utcnow()

notification.status = "read"

db.commit()

await websocket.send\_json({

"type": "mark\_read\_success",

"data": {"notification\_id": notification\_id}

})

except WebSocketDisconnect:

break

except Exception as e:

logger.error(f"WebSocket处理消息时出错: {str(e)}")

break

except Exception as e:

logger.error(f"WebSocket连接处理出错: {str(e)}")

finally:

# 清理连接

if user\_id in self.active\_connections:

if websocket in self.active\_connections[user\_id]:

self.active\_connections[user\_id].remove(websocket)

if not self.active\_connections[user\_id]:

del self.active\_connections[user\_id]

async def send\_bulk\_notification(

self,

user\_ids: List[int],

title: str,

content: str,

notification\_type: str = "system",

channel: str = "system",

priority: str = "normal",

template\_id: Optional[int] = None,

extra\_data: Optional[Dict[str, Any]] = None,

created\_by: int = None,

db: Session = None

) -> List[int]:

"""批量发送通知"""

notification\_ids = []

try:

for user\_id in user\_ids:

notification = Notification(

user\_id=user\_id,

title=title,

content=content,

type=notification\_type,

channel=channel,

priority=priority,

template\_id=template\_id,

extra\_data=extra\_data,

created\_by=created\_by

)

db.add(notification)

db.flush()

notification\_ids.append(notification.id)

db.commit()

# 异步发送所有通知

tasks = []

for notification\_id in notification\_ids:

task = asyncio.create\_task(

self.send\_notification\_async(notification\_id, db)

)

tasks.append(task)

await asyncio.gather(\*tasks)

except Exception as e:

logger.error(f"批量发送通知失败: {str(e)}")

db.rollback()

return notification\_ids

现在继续开发下一个模块：**消息通知系统 Vue.js 前端页面**

消息通知系统管理页面 - NotificationManagement.vue

<template>

<div class="notification-management">

<el-card class="page-card">

<template #header>

<div class="card-header">

<span class="title">消息通知管理</span>

<div class="header-actions">

<el-button

type="primary"

@click="showSendDialog = true"

:icon="Plus"

>

发送通知

</el-button>

<el-button

type="success"

@click="showBatchSendDialog = true"

:icon="Message"

>

批量发送

</el-button>

<el-button

type="info"

@click="showTemplateDialog = true"

:icon="Document"

>

模板管理

</el-button>

</div>

</div>

</template>

<!-- 统计卡片 -->

<div class="stats-row">

<el-row :gutter="20">

<el-col :span="6">

<el-card class="stats-card">

<div class="stats-content">

<div class="stats-value">{{ stats.total\_count }}</div>

<div class="stats-label">总通知数</div>

</div>

<el-progress

:percentage="100"

:show-text="false"

stroke-width="4"

color="#409eff"

/>

</el-card>

</el-col>

<el-col :span="6">

<el-card class="stats-card">

<div class="stats-content">

<div class="stats-value">{{ stats.sent\_count }}</div>

<div class="stats-label">发送成功</div>

</div>

<el-progress

:percentage="stats.success\_rate"

:show-text="false"

stroke-width="4"

color="#67c23a"

/>

</el-card>

</el-col>

<el-col :span="6">

<el-card class="stats-card">

<div class="stats-content">

<div class="stats-value">{{ stats.failed\_count }}</div>

<div class="stats-label">发送失败</div>

</div>

<el-progress

:percentage="stats.total\_count > 0 ? (stats.failed\_count / stats.total\_count \* 100) : 0"

:show-text="false"

stroke-width="4"

color="#f56c6c"

/>

</el-card>

</el-col>

<el-col :span="6">

<el-card class="stats-card">

<div class="stats-content">

<div class="stats-value">{{ stats.pending\_count }}</div>

<div class="stats-label">待发送</div>

</div>

<el-progress

:percentage="stats.total\_count > 0 ? (stats.pending\_count / stats.total\_count \* 100) : 0"

:show-text="false"

stroke-width="4"

color="#e6a23c"

/>

</el-card>

</el-col>

</el-row>

</div>

<!-- 筛选条件 -->

<div class="filter-row">

<el-form :inline="true" :model="filterForm" class="filter-form">

<el-form-item label="通知类型">

<el-select v-model="filterForm.type" placeholder="选择类型" clearable>

<el-option label="系统通知" value="system" />

<el-option label="营销活动" value="marketing" />

<el-option label="订单状态" value="order" />

<el-option label="合同提醒" value="contract" />

</el-select>

</el-form-item>

<el-form-item label="发送状态">

<el-select v-model="filterForm.status" placeholder="选择状态" clearable>

<el-option label="待发送" value="pending" />

<el-option label="发送中" value="sending" />

<el-option label="发送成功" value="sent" />

<el-option label="发送失败" value="failed" />

</el-select>

</el-form-item>

<el-form-item label="发送渠道">

<el-select v-model="filterForm.channel" placeholder="选择渠道" clearable>

<el-option label="站内信" value="internal" />

<el-option label="短信" value="sms" />

<el-option label="微信" value="wechat" />

<el-option label="邮件" value="email" />

</el-select>

</el-form-item>

<el-form-item label="发送时间">

<el-date-picker

v-model="filterForm.dateRange"

type="daterange"

range-separator="至"

start-placeholder="开始日期"

end-placeholder="结束日期"

format="YYYY-MM-DD"

value-format="YYYY-MM-DD"

/>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="getNotifications" :icon="Search">

搜索

</el-button>

<el-button @click="resetFilter" :icon="Refresh">

重置

</el-button>

</el-form-item>

</el-form>

</div>

<!-- 通知列表 -->

<div class="table-container">

<el-table

:data="notifications"

v-loading="loading"

style="width: 100%"

@selection-change="handleSelectionChange"

>

<el-table-column type="selection" width="55" />

<el-table-column prop="id" label="ID" width="80" />

<el-table-column prop="title" label="标题" min-width="200" />

<el-table-column prop="type" label="类型" width="100">

<template #default="scope">

<el-tag :type="getTypeTagType(scope.row.type)">

{{ getTypeLabel(scope.row.type) }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="channel" label="发送渠道" width="120">

<template #default="scope">

<el-tag :type="getChannelTagType(scope.row.channel)">

{{ getChannelLabel(scope.row.channel) }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="target\_count" label="目标用户数" width="120" />

<el-table-column prop="sent\_count" label="发送成功" width="100" />

<el-table-column prop="failed\_count" label="发送失败" width="100" />

<el-table-column prop="status" label="状态" width="100">

<template #default="scope">

<el-tag :type="getStatusTagType(scope.row.status)">

{{ getStatusLabel(scope.row.status) }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="created\_at" label="创建时间" width="160">

<template #default="scope">

{{ formatDateTime(scope.row.created\_at) }}

</template>

</el-table-column>

<el-table-column label="操作" width="180" fixed="right">

<template #default="scope">

<el-button

size="small"

@click="viewNotification(scope.row)"

:icon="View"

>

查看

</el-button>

<el-button

v-if="scope.row.status === 'pending'"

size="small"

type="primary"

@click="sendNotification(scope.row)"

:icon="Promotion"

>

发送

</el-button>

<el-button

v-if="scope.row.status === 'failed'"

size="small"

type="warning"

@click="retrySendNotification(scope.row)"

:icon="RefreshRight"

>

重试

</el-button>

</template>

</el-table-column>

</el-table>

<!-- 分页 -->

<div class="pagination-container">

<el-pagination

v-model:current-page="pagination.page"

v-model:page-size="pagination.size"

:page-sizes="[10, 20, 50, 100]"

:total="pagination.total"

layout="total, sizes, prev, pager, next, jumper"

@size-change="getNotifications"

@current-change="getNotifications"

/>

</div>

</div>

</el-card>

<!-- 发送通知对话框 -->

<el-dialog

v-model="showSendDialog"

title="发送通知"

width="600px"

@close="resetSendForm"

>

<el-form

ref="sendFormRef"

:model="sendForm"

:rules="sendRules"

label-width="100px"

>

<el-form-item label="通知类型" prop="type">

<el-select v-model="sendForm.type" placeholder="选择通知类型">

<el-option label="系统通知" value="system" />

<el-option label="营销活动" value="marketing" />

<el-option label="订单状态" value="order" />

<el-option label="合同提醒" value="contract" />

</el-select>

</el-form-item>

<el-form-item label="发送渠道" prop="channel">

<el-checkbox-group v-model="sendForm.channel">

<el-checkbox label="internal">站内信</el-checkbox>

<el-checkbox label="sms">短信</el-checkbox>

<el-checkbox label="wechat">微信</el-checkbox>

<el-checkbox label="email">邮件</el-checkbox>

</el-checkbox-group>

</el-form-item>

<el-form-item label="通知标题" prop="title">

<el-input v-model="sendForm.title" placeholder="请输入通知标题" />

</el-form-item>

<el-form-item label="通知内容" prop="content">

<el-input

v-model="sendForm.content"

type="textarea"

:rows="4"

placeholder="请输入通知内容"

/>

</el-form-item>

<el-form-item label="目标用户" prop="target\_users">

<el-select

v-model="sendForm.target\_users"

multiple

filterable

remote

reserve-keyword

placeholder="请选择目标用户"

:remote-method="searchUsers"

:loading="userSearchLoading"

>

<el-option

v-for="user in userOptions"

:key="user.id"

:label="`${user.name} (${user.phone})`"

:value="user.id"

/>

</el-select>

</el-form-item>

<el-form-item label="定时发送">

<el-switch v-model="sendForm.scheduled" />

<el-date-picker

v-if="sendForm.scheduled"

v-model="sendForm.scheduled\_time"

type="datetime"

placeholder="选择发送时间"

style="margin-left: 10px"

/>

</el-form-item>

</el-form>

<template #footer>

<el-button @click="showSendDialog = false">取消</el-button>

<el-button type="primary" @click="submitSendForm" :loading="submitLoading">

{{ sendForm.scheduled ? '定时发送' : '立即发送' }}

</el-button>

</template>

</el-dialog>

<!-- 批量发送对话框 -->

<el-dialog

v-model="showBatchSendDialog"

title="批量发送通知"

width="700px"

@close="resetBatchSendForm"

>

<el-form

ref="batchSendFormRef"

:model="batchSendForm"

:rules="batchSendRules"

label-width="100px"

>

<el-form-item label="选择模板" prop="template\_id">

<el-select

v-model="batchSendForm.template\_id"

placeholder="选择通知模板"

@change="onTemplateChange"

>

<el-option

v-for="template in templates"

:key="template.id"

:label="template.name"

:value="template.id"

/>

</el-select>

</el-form-item>

<el-form-item label="用户筛选" prop="user\_filter">

<el-tabs v-model="batchSendForm.filter\_type">

<el-tab-pane label="按条件筛选" name="condition">

<el-form :inline="true">

<el-form-item label="用户状态">

<el-select v-model="batchSendForm.user\_status" multiple placeholder="选择用户状态">

<el-option label="正常" value="active" />

<el-option label="待认证" value="pending" />

<el-option label="已认证" value="verified" />

</el-select>

</el-form-item>

<el-form-item label="注册时间">

<el-date-picker

v-model="batchSendForm.register\_date\_range"

type="daterange"

range-separator="至"

start-placeholder="开始日期"

end-placeholder="结束日期"

/>

</el-form-item>

<el-form-item label="用电规模">

<el-select v-model="batchSendForm.power\_scale" multiple placeholder="选择用电规模">

<el-option label="小微企业" value="small" />

<el-option label="中型企业" value="medium" />

<el-option label="大型企业" value="large" />

</el-select>

</el-form-item>

</el-form>

</el-tab-pane>

<el-tab-pane label="指定用户" name="specific">

<el-select

v-model="batchSendForm.specific\_users"

multiple

filterable

remote

reserve-keyword

placeholder="请选择目标用户"

:remote-method="searchUsers"

:loading="userSearchLoading"

style="width: 100%"

>

<el-option

v-for="user in userOptions"

:key="user.id"

:label="`${user.name} (${user.phone})`"

:value="user.id"

/>

</el-select>

</el-tab-pane>

</el-tabs>

</el-form-item>

<el-form-item label="预览用户">

<el-button @click="previewTargetUsers" :loading="previewLoading">

预览目标用户 ({{ targetUserCount }})

</el-button>

</el-form-item>

</el-form>

<template #footer>

<el-button @click="showBatchSendDialog = false">取消</el-button>

<el-button type="primary" @click="submitBatchSendForm" :loading="submitLoading">

批量发送

</el-button>

</template>

</el-dialog>

<!-- 模板管理对话框 -->

<el-dialog

v-model="showTemplateDialog"

title="模板管理"

width="800px"

>

<div class="template-management">

<div class="template-header">

<el-button type="primary" @click="showCreateTemplateDialog = true" :icon="Plus">

新建模板

</el-button>

</div>

<el-table :data="templates" style="width: 100%">

<el-table-column prop="name" label="模板名称" />

<el-table-column prop="type" label="类型">

<template #default="scope">

<el-tag>{{ getTypeLabel(scope.row.type) }}</el-tag>

</template>

</el-table-column>

<el-table-column prop="channel" label="适用渠道">

<template #default="scope">

<el-tag

v-for="channel in scope.row.channel"

:key="channel"

style="margin-right: 5px"

>

{{ getChannelLabel(channel) }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="created\_at" label="创建时间">

<template #default="scope">

{{ formatDateTime(scope.row.created\_at) }}

</template>

</el-table-column>

<el-table-column label="操作" width="150">

<template #default="scope">

<el-button size="small" @click="editTemplate(scope.row)">

编辑

</el-button>

<el-button

size="small"

type="danger"

@click="deleteTemplate(scope.row)"

>

删除

</el-button>

</template>

</el-table-column>

</el-table>

</div>

</el-dialog>

<!-- 创建/编辑模板对话框 -->

<el-dialog

v-model="showCreateTemplateDialog"

:title="editingTemplate ? '编辑模板' : '创建模板'"

width="600px"

@close="resetTemplateForm"

>

<el-form

ref="templateFormRef"

:model="templateForm"

:rules="templateRules"

label-width="100px"

>

<el-form-item label="模板名称" prop="name">

<el-input v-model="templateForm.name" placeholder="请输入模板名称" />

</el-form-item>

<el-form-item label="模板类型" prop="type">

<el-select v-model="templateForm.type" placeholder="选择模板类型">

<el-option label="系统通知" value="system" />

<el-option label="营销活动" value="marketing" />

<el-option label="订单状态" value="order" />

<el-option label="合同提醒" value="contract" />

</el-select>

</el-form-item>

<el-form-item label="适用渠道" prop="channel">

<el-checkbox-group v-model="templateForm.channel">

<el-checkbox label="internal">站内信</el-checkbox>

<el-checkbox label="sms">短信</el-checkbox>

<el-checkbox label="wechat">微信</el-checkbox>

<el-checkbox label="email">邮件</el-checkbox>

</el-checkbox-group>

</el-form-item>

<el-form-item label="模板标题" prop="title">

<el-input

v-model="templateForm.title"

placeholder="请输入模板标题，支持变量：{name}, {company}"

/>

</el-form-item>

<el-form-item label="模板内容" prop="content">

<el-input

v-model="templateForm.content"

type="textarea"

:rows="6"

placeholder="请输入模板内容，支持变量：{name}, {company}, {phone}, {amount} 等"

/>

</el-form-item>

<el-form-item label="变量说明">

<el-alert

title="可用变量"

type="info"

show-icon

:closable="false"

>

<p>{name} - 用户姓名</p>

<p>{company} - 公司名称</p>

<p>{phone} - 手机号码</p>

<p>{amount} - 金额</p>

<p>{date} - 日期</p>

</el-alert>

</el-form-item>

</el-form>

<template #footer>

<el-button @click="showCreateTemplateDialog = false">取消</el-button>

<el-button type="primary" @click="submitTemplateForm" :loading="submitLoading">

{{ editingTemplate ? '更新' : '创建' }}

</el-button>

</template>

</el-dialog>

<!-- 通知详情对话框 -->

<el-dialog

v-model="showDetailDialog"

title="通知详情"

width="800px"

>

<div v-if="currentNotification" class="notification-detail">

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

<el-descriptions-item label="通知标题">

{{ currentNotification.title }}

</el-descriptions-item>

<el-descriptions-item label="通知类型">

<el-tag :type="getTypeTagType(currentNotification.type)">

{{ getTypeLabel(currentNotification.type) }}

</el-tag>

</el-descriptions-item>

<el-descriptions-item label="发送渠道">

<el-tag

v-for="channel in currentNotification.channel"

:key="channel"

:type="getChannelTagType(channel)"

style="margin-right: 5px"

>

{{ getChannelLabel(channel) }}

</el-tag>

</el-descriptions-item>

<el-descriptions-item label="发送状态">

<el-tag :type="getStatusTagType(currentNotification.status)">

{{ getStatusLabel(currentNotification.status) }}

</el-tag>

</el-descriptions-item>

<el-descriptions-item label="目标用户数">

{{ currentNotification.target\_count }}

</el-descriptions-item>

<el-descriptions-item label="发送成功">

{{ currentNotification.sent\_count }}

</el-descriptions-item>

<el-descriptions-item label="发送失败">

{{ currentNotification.failed\_count }}

</el-descriptions-item>

<el-descriptions-item label="创建时间">

{{ formatDateTime(currentNotification.created\_at) }}

</el-descriptions-item>

<el-descriptions-item label="通知内容" :span="2">

<div class="notification-content">

{{ currentNotification.content }}

</div>

</el-descriptions-item>

</el-descriptions>

<!-- 发送记录 -->

<div class="send-records" style="margin-top: 20px">

<h4>发送记录</h4>

<el-table :data="sendRecords" size="small">

<el-table-column prop="user\_name" label="用户" />

<el-table-column prop="channel" label="渠道">

<template #default="scope">

<el-tag size="small">{{ getChannelLabel(scope.row.channel) }}</el-tag>

</template>

</el-table-column>

<el-table-column prop="status" label="状态">

<template #default="scope">

<el-tag size="small" :type="getStatusTagType(scope.row.status)">

{{ getStatusLabel(scope.row.status) }}

</el-tag>

</template>

</el-table-column>

<el-table-column prop="error\_message" label="错误信息" />

<el-table-column prop="sent\_at" label="发送时间">

<template #default="scope">

{{ formatDateTime(scope.row.sent\_at) }}

</template>

</el-table-column>

</el-table>

</div>

</div>

</el-dialog>

</div>

</template>

<script setup>

import { ref, reactive, onMounted, computed } from 'vue'

import { ElMessage, ElMessageBox } from 'element-plus'

import {

Plus,

Message,

Document,

Search,

Refresh,

View,

Promotion,

RefreshRight

} from '@element-plus/icons-vue'

import { notificationAPI } from '@/api/notification'

import { userAPI } from '@/api/user'

// 数据定义

const loading = ref(false)

const submitLoading = ref(false)

const userSearchLoading = ref(false)

const previewLoading = ref(false)

// 统计数据

const stats = reactive({

total\_count: 0,

sent\_count: 0,

failed\_count: 0,

pending\_count: 0,

success\_rate: 0

})

// 筛选表单

const filterForm = reactive({

type: '',

status: '',

channel: '',

dateRange: []

})

// 通知列表

const notifications = ref([])

const selectedNotifications = ref([])

// 分页

const pagination = reactive({

page: 1,

size: 20,

total: 0

})

// 对话框状态

const showSendDialog = ref(false)

const showBatchSendDialog = ref(false)

const showTemplateDialog = ref(false)

const showCreateTemplateDialog = ref(false)

const showDetailDialog = ref(false)

// 发送表单

const sendFormRef = ref()

const sendForm = reactive({

type: '',

channel: [],

title: '',

content: '',

target\_users: [],

scheduled: false,

scheduled\_time: ''

})

const sendRules = {

type: [{ required: true, message: '请选择通知类型', trigger: 'change' }],

channel: [{ required: true, message: '请选择发送渠道', trigger: 'change' }],

title: [{ required: true, message: '请输入通知标题', trigger: 'blur' }],

content: [{ required: true, message: '请输入通知内容', trigger: 'blur' }],

target\_users: [{ required: true, message: '请选择目标用户', trigger: 'change' }]

}

// 批量发送表单

const batchSendFormRef = ref()

const batchSendForm = reactive({

template\_id: '',

filter\_type: 'condition',

user\_status: [],

register\_date\_range: [],

power\_scale: [],

specific\_users: []

})

const batchSendRules = {

template\_id: [{ required: true, message: '请选择模板', trigger: 'change' }]

}

// 模板相关

const templates = ref([])

const templateFormRef = ref()

const templateForm = reactive({

name: '',

type: '',

channel: [],

title: '',

content: ''

})

const templateRules = {

name: [{ required: true, message: '请输入模板名称', trigger: 'blur' }],

type: [{ required: true, message: '请选择模板类型', trigger: 'change' }],

channel: [{ required: true, message: '请选择适用渠道', trigger: 'change' }],

title: [{ required: true, message: '请输入模板标题', trigger: 'blur' }],

content: [{ required: true, message: '请输入模板内容', trigger: 'blur' }]

}

const editingTemplate = ref(null)

// 用户选项

const userOptions = ref([])

const targetUserCount = ref(0)

// 当前查看的通知

const currentNotification = ref(null)

const sendRecords = ref([])

// 计算属性

const successRate = computed(() => {

if (stats.total\_count === 0) return 0

return Math.round((stats.sent\_count / stats.total\_count) \* 100)

})

// 方法定义

const getNotifications = async () => {

loading.value = true

try {

const params = {

page: pagination.page,

size: pagination.size,

...filterForm

}

if (filterForm.dateRange && filterForm.dateRange.length === 2) {

params.start\_date = filterForm.dateRange[0]

params.end\_date = filterForm.dateRange[1]

}

const response = await notificationAPI.getNotifications(params)

notifications.value = response.data.items

pagination.total = response.data.total

// 更新统计数据

const statsResponse = await notificationAPI.getNotificationStats()

Object.assign(stats, statsResponse.data)

stats.success\_rate = successRate.value

} catch (error) {

ElMessage.error('获取通知列表失败')

}

loading.value = false

}

const resetFilter = () => {

Object.assign(filterForm, {

type: '',

status: '',

channel: '',

dateRange: []

})

getNotifications()

}

const handleSelectionChange = (selection) => {

selectedNotifications.value = selection

}

const searchUsers = async (query) => {

if (!query) return

userSearchLoading.value = true

try {

const response = await userAPI.searchUsers({ keyword: query })

userOptions.value = response.data

} catch (error) {

console.error('搜索用户失败:', error)

}

userSearchLoading.value = false

}

const previewTargetUsers = async () => {

previewLoading.value = true

try {

const params = {

filter\_type: batchSendForm.filter\_type,

user\_status: batchSendForm.user\_status,

register\_date\_range: batchSendForm.register\_date\_range,

power\_scale: batchSendForm.power\_scale,

specific\_users: batchSendForm.specific\_users

}

const response = await notificationAPI.previewTargetUsers(params)

targetUserCount.value = response.data.count

ElMessage.success(`预览完成，共 ${response.data.count} 个目标用户`)

} catch (error) {

ElMessage.error('预览失败')

}

previewLoading.value = false

}

const submitSendForm = async () => {

if (!sendFormRef.value) return

await sendFormRef.value.validate()

submitLoading.value = true

try {

await notificationAPI.sendNotification(sendForm)

ElMessage.success('通知发送成功')

showSendDialog.value = false

resetSendForm()

getNotifications()

} catch (error) {

ElMessage.error('发送失败')

}

submitLoading.value = false

}

const submitBatchSendForm = async () => {

if (!batchSendFormRef.value) return

await batchSendFormRef.value.validate()

submitLoading.value = true

try {

await notificationAPI.batchSendNotification(batchSendForm)

ElMessage.success('批量发送成功')

showBatchSendDialog.value = false

resetBatchSendForm()

getNotifications()

} catch (error) {

ElMessage.error('批量发送失败')

}

submitLoading.value = false

}

const resetSendForm = () => {

Object.assign(sendForm, {

type: '',

channel: [],

title: '',

content: '',

target\_users: [],

scheduled: false,

scheduled\_time: ''

})

sendFormRef.value?.clearValidate()

}

const resetBatchSendForm = () => {

Object.assign(batchSendForm, {

template\_id: '',

filter\_type: 'condition',

user\_status: [],

register\_date\_range: [],

power\_scale: [],

specific\_users: []

})

batchSendFormRef.value?.clearValidate()

targetUserCount.value = 0

}

const getTemplates = async () => {

try {

const response = await notificationAPI.getTemplates()

templates.value = response.data

} catch (error) {

ElMessage.error('获取模板列表失败')

}

}

const onTemplateChange = (templateId) => {

const template = templates.value.find(t => t.id === templateId)

if (template) {

// 预填充模板信息

ElMessage.info(`已选择模板: ${template.name}`)

}

}

const submitTemplateForm = async () => {

if (!templateFormRef.value) return

await templateFormRef.value.validate()

submitLoading.value = true

try {

if (editingTemplate.value) {

await notificationAPI.updateTemplate(editingTemplate.value.id, templateForm)

ElMessage.success('模板更新成功')

} else {

await notificationAPI.createTemplate(templateForm)

ElMessage.success('模板创建成功')

}

showCreateTemplateDialog.value = false

resetTemplateForm()

getTemplates()

} catch (error) {

ElMessage.error(editingTemplate.value ? '模板更新失败' : '模板创建失败')

}

submitLoading.value = false

}

const resetTemplateForm = () => {

Object.assign(templateForm, {

name: '',

type: '',

channel: [],

title: '',

content: ''

})

templateFormRef.value?.clearValidate()

editingTemplate.value = null

}

const editTemplate = (template) => {

editingTemplate.value = template

Object.assign(templateForm, {

name: template.name,

type: template.type,

channel: template.channel,

title: template.title,

content: template.content

})

showCreateTemplateDialog.value = true

}

const deleteTemplate = async (template) => {

try {

await ElMessageBox.confirm(

`确定要删除模板 "${template.name}" 吗？`,

'确认删除',

{

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}

)

await notificationAPI.deleteTemplate(template.id)

ElMessage.success('模板删除成功')

getTemplates()

} catch (error) {

if (error !== 'cancel') {

ElMessage.error('模板删除失败')

}

}

}

const viewNotification = async (notification) => {

currentNotification.value = notification

showDetailDialog.value = true

// 获取发送记录

try {

const response = await notificationAPI.getSendRecords(notification.id)

sendRecords.value = response.data

} catch (error) {

ElMessage.error('获取发送记录失败')

}

}

const sendNotification = async (notification) => {

try {

await ElMessageBox.confirm(

`确定要发送通知 "${notification.title}" 吗？`,

'确认发送',

{

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}

)

await notificationAPI.sendNotificationById(notification.id)

ElMessage.success('通知发送成功')

getNotifications()

} catch (error) {

if (error !== 'cancel') {

ElMessage.error('通知发送失败')

}

}

}

const retrySendNotification = async (notification) => {

try {

await ElMessageBox.confirm(

`确定要重试发送通知 "${notification.title}" 吗？`,

'确认重试',

{

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}

)

await notificationAPI.retrySendNotification(notification.id)

ElMessage.success('重试发送成功')

getNotifications()

} catch (error) {

if (error !== 'cancel') {

ElMessage.error('重试发送失败')

}

}

}

// 标签类型获取方法

const getTypeTagType = (type) => {

const typeMap = {

system: '',

marketing: 'success',

order: 'warning',

contract: 'info'

}

return typeMap[type] || ''

}

const getTypeLabel = (type) => {

const typeMap = {

system: '系统通知',

marketing: '营销活动',

order: '订单状态',

contract: '合同提醒'

}

return typeMap[type] || type

}

const getChannelTagType = (channel) => {

const channelMap = {

internal: 'info',

sms: 'warning',

wechat: 'success',

email: ''

}

return channelMap[channel] || ''

}

const getChannelLabel = (channel) => {

const channelMap = {

internal: '站内信',

sms: '短信',

wechat: '微信',

email: '邮件'

}

return channelMap[channel] || channel

}

const getStatusTagType = (status) => {

const statusMap = {

pending: 'info',

sending: 'warning',

sent: 'success',

failed: 'danger'

}

return statusMap[status] || ''

}

const getStatusLabel = (status) => {

const statusMap = {

pending: '待发送',

sending: '发送中',

sent: '发送成功',

failed: '发送失败'

}

return statusMap[status] || status

}

const formatDateTime = (dateTime) => {

if (!dateTime) return ''

return new Date(dateTime).toLocaleString('zh-CN')

}

// 生命周期

onMounted(() => {

getNotifications()

getTemplates()

})

</script>

<style lang="scss" scoped>

.notification-management {

.page-card {

margin: 20px;

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

.title {

font-size: 18px;

font-weight: 500;

}

.header-actions {

display: flex;

gap: 10px;

}

}

}

.stats-row {

margin-bottom: 20px;

.stats-card {

.stats-content {

display: flex;

flex-direction: column;

align-items: center;

margin-bottom: 10px;

.stats-value {

font-size: 32px;

font-weight: bold;

color: #409eff;

line-height: 1;

}

.stats-label {

font-size: 14px;

color: #666;

margin-top: 5px;

}

}

}

}

.filter-row {

margin-bottom: 20px;

padding: 20px;

background: #f5f7fa;

border-radius: 4px;

.filter-form {

.el-form-item {

margin-bottom: 0;

}

}

}

.table-container {

.pagination-container {

display: flex;

justify-content: center;

margin-top: 20px;

}

}

.template-management {

.template-header {

margin-bottom: 20px;

display: flex;

justify-content: flex-end;

}

}

.notification-detail {

.notification-content {

padding: 10px;

background: #f5f7fa;

border-radius: 4px;

white-space: pre-wrap;

word-break: break-all;

}

.send-records {

h4 {

margin: 0 0 10px 0;

color: #333;

}

}

}

}

// 响应式设计

@media (max-width: 768px) {

.notification-management {

.page-card {

margin: 10px;

}

.card-header {

flex-direction: column;

gap: 10px;

.header-actions {

width: 100%;

justify-content: center;

}

}

.stats-row {

:deep(.el-col) {

margin-bottom: 10px;

}

}

.filter-row {

.filter-form {

:deep(.el-form-item) {

display: block;

margin-bottom: 10px;

}

}

}

}

}

</style>

现在我来开发下一个模块：**文件管理系统 API 后端**

文件管理系统 API - file\_management.py

"""

文件管理系统 API

处理合同文档、附件管理等文件相关业务

"""

from fastapi import APIRouter, Depends, HTTPException, UploadFile, File, Form

from fastapi.responses import StreamingResponse

from sqlalchemy.orm import Session

from typing import List, Optional, Dict, Any

import os

import uuid

import hashlib

from datetime import datetime

import mimetypes

import aiofiles

from pathlib import Path

from app.database import get\_db

from app.models.file import FileRecord, FileCategory, FilePermission

from app.models.user import User

from app.models.order import Order

from app.schemas.file import (

FileRecordCreate, FileRecordUpdate, FileRecordResponse,

FileCategoryCreate, FileCategoryResponse,

FilePermissionCreate, FilePermissionResponse,

FileListResponse, FileUploadResponse, FileSearchParams

)

from app.core.security import get\_current\_user

from app.core.config import settings

from app.services.file\_service import FileService

from app.services.auth\_service import AuthService

router = APIRouter()

file\_service = FileService()

auth\_service = AuthService()

# 文件上传配置

MAX\_FILE\_SIZE = 100 \* 1024 \* 1024 # 100MB

ALLOWED\_EXTENSIONS = {

'image': ['.jpg', '.jpeg', '.png', '.gif', '.bmp', '.webp'],

'document': ['.pdf', '.doc', '.docx', '.xls', '.xlsx', '.ppt', '.pptx', '.txt'],

'contract': ['.pdf', '.doc', '.docx'],

'certificate': ['.jpg', '.jpeg', '.png', '.pdf'],

'other': ['.zip', '.rar', '.7z']

}

@router.post("/upload", response\_model=FileUploadResponse)

async def upload\_file(

file: UploadFile = File(...),

category: str = Form(...),

related\_type: Optional[str] = Form(None),

related\_id: Optional[int] = Form(None),

description: Optional[str] = Form(None),

is\_public: bool = Form(False),

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

上传文件

"""

try:

# 验证文件大小

if file.size > MAX\_FILE\_SIZE:

raise HTTPException(status\_code=400, detail="文件大小超出限制")

# 验证文件类型

file\_ext = Path(file.filename).suffix.lower()

if not \_is\_allowed\_file\_type(file\_ext, category):

raise HTTPException(

status\_code=400,

detail=f"不支持的文件类型: {file\_ext}"

)

# 生成文件信息

file\_id = str(uuid.uuid4())

file\_hash = await \_calculate\_file\_hash(file)

# 检查文件是否已存在

existing\_file = db.query(FileRecord).filter(

FileRecord.file\_hash == file\_hash,

FileRecord.uploader\_id == current\_user.id

).first()

if existing\_file:

return FileUploadResponse(

id=existing\_file.id,

file\_id=existing\_file.file\_id,

filename=existing\_file.original\_filename,

file\_url=existing\_file.file\_url,

message="文件已存在，返回已有文件信息"

)

# 保存文件

file\_path = await \_save\_uploaded\_file(file, file\_id, category)

# 创建文件记录

file\_record = FileRecord(

file\_id=file\_id,

original\_filename=file.filename,

stored\_filename=f"{file\_id}{file\_ext}",

file\_path=file\_path,

file\_url=f"/api/v1/files/download/{file\_id}",

file\_size=file.size,

file\_type=file.content\_type,

file\_hash=file\_hash,

category=category,

related\_type=related\_type,

related\_id=related\_id,

description=description,

is\_public=is\_public,

uploader\_id=current\_user.id,

status='active'

)

db.add(file\_record)

db.commit()

db.refresh(file\_record)

# 记录上传日志

await file\_service.log\_file\_activity(

file\_record.id, 'upload', current\_user.id,

f"用户上传文件: {file.filename}"

)

return FileUploadResponse(

id=file\_record.id,

file\_id=file\_record.file\_id,

filename=file\_record.original\_filename,

file\_url=file\_record.file\_url,

message="文件上传成功"

)

except Exception as e:

db.rollback()

raise HTTPException(status\_code=500, detail=f"文件上传失败: {str(e)}")

@router.get("/list", response\_model=FileListResponse)

async def get\_file\_list(

category: Optional[str] = None,

related\_type: Optional[str] = None,

related\_id: Optional[int] = None,

keyword: Optional[str] = None,

page: int = 1,

size: int = 20,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

获取文件列表

"""

try:

# 构建查询条件

query = db.query(FileRecord).filter(FileRecord.status == 'active')

# 权限过滤

if not auth\_service.has\_permission(current\_user, 'file:view\_all'):

query = query.filter(

(FileRecord.uploader\_id == current\_user.id) |

(FileRecord.is\_public == True)

)

# 添加筛选条件

if category:

query = query.filter(FileRecord.category == category)

if related\_type:

query = query.filter(FileRecord.related\_type == related\_type)

if related\_id:

query = query.filter(FileRecord.related\_id == related\_id)

if keyword:

query = query.filter(

FileRecord.original\_filename.contains(keyword) |

FileRecord.description.contains(keyword)

)

# 排序和分页

total = query.count()

files = query.order\_by(FileRecord.created\_at.desc()).offset(

(page - 1) \* size

).limit(size).all()

return FileListResponse(

items=[FileRecordResponse.from\_orm(file) for file in files],

total=total,

page=page,

size=size,

pages=(total + size - 1) // size

)

except Exception as e:

raise HTTPException(status\_code=500, detail=f"获取文件列表失败: {str(e)}")

@router.get("/download/{file\_id}")

async def download\_file(

file\_id: str,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

下载文件

"""

try:

# 查找文件记录

file\_record = db.query(FileRecord).filter(

FileRecord.file\_id == file\_id,

FileRecord.status == 'active'

).first()

if not file\_record:

raise HTTPException(status\_code=404, detail="文件不存在")

# 权限检查

if not \_check\_file\_permission(file\_record, current\_user):

raise HTTPException(status\_code=403, detail="无权访问该文件")

# 检查文件是否存在

if not os.path.exists(file\_record.file\_path):

raise HTTPException(status\_code=404, detail="文件不存在")

# 记录下载日志

await file\_service.log\_file\_activity(

file\_record.id, 'download', current\_user.id,

f"用户下载文件: {file\_record.original\_filename}"

)

# 更新下载次数

file\_record.download\_count += 1

file\_record.last\_download\_at = datetime.utcnow()

db.commit()

# 返回文件流

def file\_generator():

with open(file\_record.file\_path, 'rb') as file:

while True:

chunk = file.read(8192)

if not chunk:

break

yield chunk

# 获取文件MIME类型

content\_type = mimetypes.guess\_type(file\_record.original\_filename)[0]

if not content\_type:

content\_type = 'application/octet-stream'

return StreamingResponse(

file\_generator(),

media\_type=content\_type,

headers={

'Content-Disposition': f'attachment; filename="{file\_record.original\_filename}"'

}

)

except Exception as e:

if "无权访问" in str(e) or "不存在" in str(e):

raise e

raise HTTPException(status\_code=500, detail=f"文件下载失败: {str(e)}")

@router.get("/{file\_id}", response\_model=FileRecordResponse)

async def get\_file\_info(

file\_id: str,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

获取文件信息

"""

try:

file\_record = db.query(FileRecord).filter(

FileRecord.file\_id == file\_id,

FileRecord.status == 'active'

).first()

if not file\_record:

raise HTTPException(status\_code=404, detail="文件不存在")

# 权限检查

if not \_check\_file\_permission(file\_record, current\_user):

raise HTTPException(status\_code=403, detail="无权访问该文件")

return FileRecordResponse.from\_orm(file\_record)

except Exception as e:

if "无权访问" in str(e) or "不存在" in str(e):

raise e

raise HTTPException(status\_code=500, detail=f"获取文件信息失败: {str(e)}")

@router.put("/{file\_id}", response\_model=FileRecordResponse)

async def update\_file\_info(

file\_id: str,

file\_update: FileRecordUpdate,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

更新文件信息

"""

try:

file\_record = db.query(FileRecord).filter(

FileRecord.file\_id == file\_id,

FileRecord.status == 'active'

).first()

if not file\_record:

raise HTTPException(status\_code=404, detail="文件不存在")

# 权限检查

if not \_check\_file\_edit\_permission(file\_record, current\_user):

raise HTTPException(status\_code=403, detail="无权修改该文件")

# 更新文件信息

update\_data = file\_update.dict(exclude\_unset=True)

for field, value in update\_data.items():

setattr(file\_record, field, value)

file\_record.updated\_at = datetime.utcnow()

db.commit()

db.refresh(file\_record)

# 记录操作日志

await file\_service.log\_file\_activity(

file\_record.id, 'update', current\_user.id,

f"用户更新文件信息: {file\_record.original\_filename}"

)

return FileRecordResponse.from\_orm(file\_record)

except Exception as e:

db.rollback()

if "无权" in str(e) or "不存在" in str(e):

raise e

raise HTTPException(status\_code=500, detail=f"更新文件信息失败: {str(e)}")

@router.delete("/{file\_id}")

async def delete\_file(

file\_id: str,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

删除文件

"""

try:

file\_record = db.query(FileRecord).filter(

FileRecord.file\_id == file\_id,

FileRecord.status == 'active'

).first()

if not file\_record:

raise HTTPException(status\_code=404, detail="文件不存在")

# 权限检查

if not \_check\_file\_delete\_permission(file\_record, current\_user):

raise HTTPException(status\_code=403, detail="无权删除该文件")

# 软删除

file\_record.status = 'deleted'

file\_record.deleted\_at = datetime.utcnow()

file\_record.deleted\_by = current\_user.id

db.commit()

# 记录操作日志

await file\_service.log\_file\_activity(

file\_record.id, 'delete', current\_user.id,

f"用户删除文件: {file\_record.original\_filename}"

)

return {"message": "文件删除成功"}

except Exception as e:

db.rollback()

if "无权" in str(e) or "不存在" in str(e):

raise e

raise HTTPException(status\_code=500, detail=f"删除文件失败: {str(e)}")

@router.post("/categories", response\_model=FileCategoryResponse)

async def create\_file\_category(

category: FileCategoryCreate,

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

创建文件分类

"""

try:

# 权限检查

if not auth\_service.has\_permission(current\_user, 'file:manage\_category'):

raise HTTPException(status\_code=403, detail="无权管理文件分类")

# 检查分类是否已存在

existing = db.query(FileCategory).filter(

FileCategory.name == category.name

).first()

if existing:

raise HTTPException(status\_code=400, detail="分类名称已存在")

# 创建分类

file\_category = FileCategory(

name=category.name,

description=category.description,

allowed\_extensions=category.allowed\_extensions,

max\_file\_size=category.max\_file\_size,

is\_active=True,

created\_by=current\_user.id

)

db.add(file\_category)

db.commit()

db.refresh(file\_category)

return FileCategoryResponse.from\_orm(file\_category)

except Exception as e:

db.rollback()

if "无权" in str(e) or "已存在" in str(e):

raise e

raise HTTPException(status\_code=500, detail=f"创建文件分类失败: {str(e)}")

@router.get("/categories", response\_model=List[FileCategoryResponse])

async def get\_file\_categories(

db: Session = Depends(get\_db),

current\_user: User = Depends(get\_current\_user)

):

"""

获取文件分类列表

"""

try:

categories = db.query(FileCategory).filter(

FileCategory.is\_active == True

).order\_by(FileCategory.sort\_order).all()

return [FileCategoryResponse.from\_orm(cat) for cat in categories]

except Exception as e:

raise HTTPException(status\_code=500, detail=f"获取文件分类失败: {str(e)}")

# 辅助函数

def \_is\_allowed\_file\_type(file\_ext: str, category: str) -> bool:

"""检查文件类型是否允许"""

allowed = ALLOWED\_EXTENSIONS.get(category, [])

return file\_ext in allowed

async def \_calculate\_file\_hash(file: UploadFile) -> str:

"""计算文件哈希值"""

hasher = hashlib.md5()

content = await file.read()

hasher.update(content)

await file.seek(0) # 重置文件指针

return hasher.hexdigest()

async def \_save\_uploaded\_file(file: UploadFile, file\_id: str, category: str) -> str:

"""保存上传的文件"""

# 创建存储目录

upload\_dir = Path(settings.UPLOAD\_DIR) / category / datetime.now().strftime('%Y/%m')

upload\_dir.mkdir(parents=True, exist\_ok=True)

# 生成文件路径

file\_ext = Path(file.filename).suffix.lower()

file\_path = upload\_dir / f"{file\_id}{file\_ext}"

# 保存文件

async with aiofiles.open(file\_path, 'wb') as f:

content = await file.read()

await f.write(content)

return str(file\_path)

def \_check\_file\_permission(file\_record: FileRecord, user: User) -> bool:

"""检查文件访问权限"""

# 公开文件

if file\_record.is\_public:

return True

# 文件上传者

if file\_record.uploader\_id == user.id:

return True

# 管理员权限

if auth\_service.has\_permission(user, 'file:view\_all'):

return True

# 相关业务权限（如订单相关文件）

if file\_record.related\_type == 'order' and file\_record.related\_id:

# 检查是否为订单相关人员

return auth\_service.has\_order\_access(user, file\_record.related\_id)

return False

def \_check\_file\_edit\_permission(file\_record: FileRecord, user: User) -> bool:

"""检查文件编辑权限"""

# 文件上传者

if file\_record.uploader\_id == user.id:

return True

# 管理员权限

if auth\_service.has\_permission(user, 'file:edit\_all'):

return True

return False

def \_check\_file\_delete\_permission(file\_record: FileRecord, user: User) -> bool:

"""检查文件删除权限"""

# 文件上传者

if file\_record.uploader\_id == user.id:

return True

# 管理员权限

if auth\_service.has\_permission(user, 'file:delete\_all'):

return True

return False

现在我来开发文件管理的数据库模型：

文件管理数据库模型 - models/file.py

"""

文件管理数据库模型

"""

from sqlalchemy import Column, Integer, String, Text, Boolean, DateTime, BigInteger, ForeignKey, JSON

from sqlalchemy.orm import relationship

from sqlalchemy.sql import func

from app.database import Base

class FileRecord(Base):

"""文件记录表"""

\_\_tablename\_\_ = "file\_records"

id = Column(Integer, primary\_key=True, index=True)

file\_name = Column(String(255), nullable=False, comment="原始文件名")

file\_path = Column(String(500), nullable=False, comment="文件存储路径")

file\_size = Column(BigInteger, nullable=False, comment="文件大小(字节)")

file\_type = Column(String(100), nullable=False, comment="文件类型")

file\_extension = Column(String(20), nullable=False, comment="文件扩展名")

mime\_type = Column(String(100), nullable=False, comment="MIME类型")

md5\_hash = Column(String(32), nullable=False, comment="文件MD5哈希值")

# 业务相关字段

business\_type = Column(String(50), nullable=False, comment="业务类型：contract, certificate, attachment")

business\_id = Column(Integer, nullable=True, comment="关联业务ID")

upload\_user\_id = Column(Integer, ForeignKey("users.id"), nullable=False, comment="上传用户ID")

# 文件状态

status = Column(String(20), default="active", comment="文件状态：active, deleted, expired")

is\_public = Column(Boolean, default=False, comment="是否公开访问")

download\_count = Column(Integer, default=0, comment="下载次数")

# 元数据

metadata = Column(JSON, nullable=True, comment="文件元数据")

tags = Column(JSON, nullable=True, comment="文件标签")

description = Column(Text, nullable=True, comment="文件描述")

# 时间戳

created\_at = Column(DateTime(timezone=True), server\_default=func.now(), comment="创建时间")

updated\_at = Column(DateTime(timezone=True), server\_default=func.now(), onupdate=func.now(), comment="更新时间")

expired\_at = Column(DateTime(timezone=True), nullable=True, comment="过期时间")

# 关系

upload\_user = relationship("User", back\_populates="uploaded\_files")

file\_permissions = relationship("FilePermission", back\_populates="file\_record")

class FilePermission(Base):

"""文件权限表"""

\_\_tablename\_\_ = "file\_permissions"

id = Column(Integer, primary\_key=True, index=True)

file\_id = Column(Integer, ForeignKey("file\_records.id"), nullable=False, comment="文件ID")

user\_id = Column(Integer, ForeignKey("users.id"), nullable=True, comment="用户ID")

role\_name = Column(String(50), nullable=True, comment="角色名称")

permission\_type = Column(String(20), nullable=False, comment="权限类型：read, write, delete, share")

# 时间戳

created\_at = Column(DateTime(timezone=True), server\_default=func.now(), comment="创建时间")

# 关系

file\_record = relationship("FileRecord", back\_populates="file\_permissions")

user = relationship("User")

class FileDownloadLog(Base):

"""文件下载日志表"""

\_\_tablename\_\_ = "file\_download\_logs"

id = Column(Integer, primary\_key=True, index=True)

file\_id = Column(Integer, ForeignKey("file\_records.id"), nullable=False, comment="文件ID")

user\_id = Column(Integer, ForeignKey("users.id"), nullable=True, comment="下载用户ID")

download\_ip = Column(String(45), nullable=True, comment="下载IP地址")

user\_agent = Column(String(500), nullable=True, comment="用户代理")

download\_method = Column(String(20), default="direct", comment="下载方式：direct, preview, share")

# 时间戳

created\_at = Column(DateTime(timezone=True), server\_default=func.now(), comment="下载时间")

# 关系

file\_record = relationship("FileRecord")

user = relationship("User")

class ContractTemplate(Base):

"""合同模板表"""

\_\_tablename\_\_ = "contract\_templates"

id = Column(Integer, primary\_key=True, index=True)

template\_name = Column(String(100), nullable=False, comment="模板名称")

template\_type = Column(String(50), nullable=False, comment="模板类型")

content = Column(Text, nullable=False, comment="模板内容")

variables = Column(JSON, nullable=True, comment="模板变量定义")

version = Column(String(20), default="1.0", comment="模板版本")

# 状态管理

status = Column(String(20), default="active", comment="状态：active, inactive, archived")

is\_default = Column(Boolean, default=False, comment="是否默认模板")

# 审核信息

created\_by = Column(Integer, ForeignKey("users.id"), nullable=False, comment="创建者")

approved\_by = Column(Integer, ForeignKey("users.id"), nullable=True, comment="审核者")

approved\_at = Column(DateTime(timezone=True), nullable=True, comment="审核时间")

# 时间戳

created\_at = Column(DateTime(timezone=True), server\_default=func.now(), comment="创建时间")

updated\_at = Column(DateTime(timezone=True), server\_default=func.now(), onupdate=func.now(), comment="更新时间")

# 关系

creator = relationship("User", foreign\_keys=[created\_by])

approver = relationship("User", foreign\_keys=[approved\_by])

contracts = relationship("Contract", back\_populates="template")

class Contract(Base):

"""合同表"""

\_\_tablename\_\_ = "contracts"

id = Column(Integer, primary\_key=True, index=True)

contract\_no = Column(String(50), unique=True, nullable=False, comment="合同编号")

template\_id = Column(Integer, ForeignKey("contract\_templates.id"), nullable=False, comment="模板ID")

order\_id = Column(Integer, ForeignKey("orders.id"), nullable=False, comment="订单ID")

customer\_id = Column(Integer, ForeignKey("users.id"), nullable=False, comment="客户ID")

# 合同内容

contract\_title = Column(String(200), nullable=False, comment="合同标题")

contract\_content = Column(Text, nullable=False, comment="合同内容")

contract\_amount = Column(Integer, nullable=False, comment="合同金额(分)")

service\_start\_date = Column(DateTime(timezone=True), nullable=False, comment="服务开始日期")

service\_end\_date = Column(DateTime(timezone=True), nullable=False, comment="服务结束日期")

# 签署状态

status = Column(String(20), default="draft", comment="状态：draft, pending, signed, expired, terminated")

customer\_signed = Column(Boolean, default=False, comment="客户是否已签署")

customer\_signed\_at = Column(DateTime(timezone=True), nullable=True, comment="客户签署时间")

company\_signed = Column(Boolean, default=False, comment="公司是否已签署")

company\_signed\_at = Column(DateTime(timezone=True), nullable=True, comment="公司签署时间")

company\_signer\_id = Column(Integer, ForeignKey("users.id"), nullable=True, comment="公司签署人")

# 文件信息

pdf\_file\_id = Column(Integer, ForeignKey("file\_records.id"), nullable=True, comment="PDF文件ID")

# 时间戳

created\_at = Column(DateTime(timezone=True), server\_default=func.now(), comment="创建时间")

updated\_at = Column(DateTime(timezone=True), server\_default=func.now(), onupdate=func.now(), comment="更新时间")

# 关系

template = relationship("ContractTemplate", back\_populates="contracts")

order = relationship("Order")

customer = relationship("User", foreign\_keys=[customer\_id])

company\_signer = relationship("User", foreign\_keys=[company\_signer\_id])

pdf\_file = relationship("FileRecord")

class ContractSignLog(Base):

"""合同签署日志表"""

\_\_tablename\_\_ = "contract\_sign\_logs"

id = Column(Integer, primary\_key=True, index=True)

contract\_id = Column(Integer, ForeignKey("contracts.id"), nullable=False, comment="合同ID")

signer\_id = Column(Integer, ForeignKey("users.id"), nullable=False, comment="签署人ID")

signer\_type = Column(String(20), nullable=False, comment="签署人类型：customer, company")

signature\_data = Column(Text, nullable=True, comment="签名数据")

sign\_ip = Column(String(45), nullable=True, comment="签署IP")

sign\_location = Column(String(100), nullable=True, comment="签署地点")

# 时间戳

created\_at = Column(DateTime(timezone=True), server\_default=func.now(), comment="签署时间")

# 关系

contract = relationship("Contract")

signer = relationship("User")