老人管理模块模型（models.py）

from django.db import models

from django.utils import timezone

class Elderly(models.Model):

"""老人信息模型"""

GENDER\_CHOICES = (

(0, '女'),

(1, '男'),

name = models.CharField(max\_length=50, verbose\_name="姓名") # 老人姓名

gender = models.IntegerField(choices=GENDER\_CHOICES, verbose\_name="性别") # 性别

age = models.IntegerField(verbose\_name="年龄") # 年龄

id\_card = models.CharField(max\_length=18, unique=True, verbose\_name="身份证号") # 身份证号（唯一）

phone = models.CharField(max\_length=20, blank=True, null=True, verbose\_name="联系电话") # 联系电话

check\_in\_date = models.DateField(blank=True, null=True, verbose\_name="入住日期") # 入住日期

check\_out\_date = models.DateField(blank=True, null=True, verbose\_name="退住日期") # 退住日期

room = models.ForeignKey('Room', on\_delete=models.SET\_NULL, null=True, related\_name='elderlies', verbose\_name="所属房间") # 关联房间

care\_level = models.CharField(max\_length=20, blank=True, null=True, verbose\_name="护理等级") # 护理等级

create\_time = models.DateTimeField(default=timezone.now, verbose\_name="创建时间") # 创建时间

update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间") # 更新时间

class Meta:

verbose\_name = "老人信息"

verbose\_name\_plural = "老人信息管理"

ordering = ['-create\_time'] # 按创建时间倒序

def \_\_str\_\_(self):

return f"{self.name}（{self.id\_card}）"

老人管理模块序列化器（serializers.py）

from rest\_framework import serializers

from .models import Elderly

class ElderlySerializer(serializers.ModelSerializer):

"""老人信息序列化器"""

# 额外返回房间号

room\_no = serializers.CharField(source='room.room\_no', read\_only=True)

# 性别中文显示（将0/1转换为男女）

gender\_display = serializers.CharField(source='get\_gender\_display', read\_only=True)

class Meta:

model = Elderly

fields = [

'id', 'name', 'gender', 'gender\_display', 'age', 'id\_card',

'phone', 'check\_in\_date', 'check\_out\_date', 'room', 'room\_no',

'care\_level', 'create\_time', 'update\_time'

]

read\_only\_fields = ['id', 'create\_time', 'update\_time'] # 只读字段

def validate\_id\_card(self, value):

"""验证身份证号格式"""

if len(value) != 18:

raise serializers.ValidationError("身份证号必须为18位")

return value

老人管理模块视图（views.py）

from rest\_framework import viewsets, filters, status

from rest\_framework.response import Response

from django\_filters.rest\_framework import DjangoFilterBackend

from .models import Elderly

from .serializers import ElderlySerializer

class ElderlyViewSet(viewsets.ModelViewSet):

"""老人信息视图集（提供CRUD接口）"""

queryset = Elderly.objects.all() # 数据集

serializer\_class = ElderlySerializer # 序列化器

# 支持过滤、搜索、排序

filter\_backends = [DjangoFilterBackend, filters.SearchFilter, filters.OrderingFilter]

filterset\_fields = ['gender', 'care\_level', 'room'] # 可过滤字段

search\_fields = ['name', 'id\_card', 'phone'] # 可搜索字段

ordering\_fields = ['age', 'check\_in\_date', 'create\_time'] # 可排序字段

def destroy(self, request, \*args, \*\*kwargs):

"""重写删除方法，支持软删除（实际项目可改为逻辑删除）"""

instance = self.get\_object()

self.perform\_destroy(instance)

return Response({"message": "删除成功"}, status=status.HTTP\_200\_OK)

路由配置（urls.py）

# 应用级路由

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from .views import ElderlyViewSet

router = DefaultRouter()

router.register(r'elderlies', ElderlyViewSet) # 注册老人管理接口

urlpatterns = [

path('api/', include(router.urls)), # 接口路径：/api/elderlies/

]

# 项目级路由（主urls.py）

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('elderly.urls')), # 包含应用路由

]

老人列表组件（ElderlyList.vue）

<template>

<div class="elderly-list-container">

<!-- 页面标题和操作按钮 -->

<a-card title="老人信息管理" :bordered="false">

<div style="margin-bottom: 16px">

<a-button type="primary" @click="showAddModal">

<plus-outlined /> 新增老人

</a-button>

<a-input-search

placeholder="搜索姓名/身份证号"

style="width: 300px; margin-left: 16px"

@search="handleSearch"

/>

</div>

<!-- 老人列表表格 -->

<a-table

:columns="columns"

:data-source="dataSource"

:loading="loading"

rowKey="id"

:pagination="pagination"

@change="handleTableChange"

>

<!-- 操作列 -->

<template #operation="{ record }">

<a-button type="link" @click="showEditModal(record)">编辑</a-button>

<a-popconfirm

title="确定删除该老人信息吗？"

@confirm="handleDelete(record.id)"

>

<a-button type="link" danger>删除</a-button>

</a-popconfirm>

</template>

</a-table>

</a-card>

<!-- 新增/编辑弹窗 -->

<elderly-modal

:visible="modalVisible"

:initial-values="formData"

:title="modalTitle"

@cancel="modalVisible = false"

@submit="handleSubmit"

/>

</div>

</template>

<script>

import { PlusOutlined } from '@ant-design/icons-vue';

import { defineComponent, reactive, toRefs, onMounted } from 'vue';

import ElderlyModal from './ElderlyModal.vue';

import { getElderlyList, deleteElderly } from '@/api/elderly'; // API请求函数

export default defineComponent({

components: { ElderlyModal, PlusOutlined },

setup() {

// 响应式数据

const state = reactive({

dataSource: [], // 表格数据

loading: false, // 加载状态

modalVisible: false, // 弹窗显示状态

formData: {}, // 弹窗表单数据

modalTitle: '新增老人', // 弹窗标题

pagination: { // 分页配置

current: 1,

pageSize: 10,

total: 0,

showSizeChanger: true,

pageSizeOptions: ['10', '20', '50']

},

searchKey: '' // 搜索关键词

});

// 表格列配置

const columns = [

{ title: '姓名', dataIndex: 'name', key: 'name' },

{ title: '性别', dataIndex: 'gender\_display', key: 'gender' },

{ title: '年龄', dataIndex: 'age', key: 'age' },

{ title: '身份证号', dataIndex: 'id\_card', key: 'id\_card' },

{ title: '护理等级', dataIndex: 'care\_level', key: 'care\_level' },

{ title: '入住日期', dataIndex: 'check\_in\_date', key: 'check\_in\_date' },

{ title: '操作', key: 'operation', slots: { customRender: 'operation' } }

];

// 获取老人列表数据

const fetchData = async () => {

state.loading = true;

try {

const params = {

page: state.pagination.current,

page\_size: state.pagination.pageSize,

search: state.searchKey

};

const res = await getElderlyList(params);

state.dataSource = res.results;

state.pagination.total = res.count; // 总条数

} catch (error) {

console.error('获取数据失败：', error);

} finally {

state.loading = false;

}

};

// 初始化加载数据

onMounted(fetchData);

// 搜索事件

const handleSearch = (value) => {

state.searchKey = value;

state.pagination.current = 1; // 重置页码

fetchData();

};

// 分页/排序变化

const handleTableChange = (pagination) => {

state.pagination = pagination;

fetchData();

};

// 显示新增弹窗

const showAddModal = () => {

state.formData = {};

state.modalTitle = '新增老人';

state.modalVisible = true;

};

// 显示编辑弹窗

const showEditModal = (record) => {

state.formData = { ...record };

state.modalTitle = '编辑老人';

state.modalVisible = true;

};

// 提交表单（新增/编辑）

const handleSubmit = async () => {

state.modalVisible = false;

fetchData(); // 提交后刷新列表

};

// 删除老人信息

const handleDelete = async (id) => {

try {

await deleteElderly(id);

fetchData(); // 删除后刷新列表

} catch (error) {

console.error('删除失败：', error);

}

};

return {

...toRefs(state),

columns,

fetchData,

handleSearch,

handleTableChange,

showAddModal,

showEditModal,

handleSubmit,

handleDelete

};

}

});

</script>

<style scoped>

.elderly-list-container {

padding: 20px;

}

</style>

老人信息表单弹窗（ElderlyModal.vue）

<template>

<a-modal

:visible="visible"

:title="title"

@cancel="handleCancel"

@ok="handleOk"

>

<a-form

:model="form"

:rules="rules"

ref="formRef"

layout="vertical"

>

<a-form-item name="name" label="姓名">

<a-input v-model:value="form.name" placeholder="请输入姓名" />

</a-form-item>

<a-form-item name="gender" label="性别">

<a-radio-group v-model:value="form.gender">

<a-radio :value="1">男</a-radio>

<a-radio :value="0">女</a-radio>

</a-radio-group>

</a-form-item>

<a-form-item name="age" label="年龄">

<a-input-number v-model:value="form.age" min="0" max="150" placeholder="请输入年龄" />

</a-form-item>

<a-form-item name="id\_card" label="身份证号">

<a-input v-model:value="form.id\_card" placeholder="请输入18位身份证号" />

</a-form-item>

<a-form-item name="care\_level" label="护理等级">

<a-select v-model:value="form.care\_level" placeholder="请选择护理等级">

<a-select-option value="一级">一级护理</a-select-option>

<a-select-option value="二级">二级护理</a-select-option>

<a-select-option value="三级">三级护理</a-select-option>

</a-select>

</a-form-item>

</a-form>

</a-modal>

</template>

<script>

import { defineComponent, reactive, toRefs, ref, watch } from 'vue';

import { addElderly, updateElderly } from '@/api/elderly'; // API请求函数

export default defineComponent({

props: {

visible: { type: Boolean, default: false },

initialValues: { type: Object, default: () => ({}) },

title: { type: String, default: '' }

},

emits: ['cancel', 'submit'],

setup(props, { emit }) {

const formRef = ref(null);

const form = reactive({

name: '',

gender: 1,

age: null,

id\_card: '',

care\_level: ''

});

// 表单验证规则

const rules = {

name: [{ required: true, message: '请输入姓名', trigger: 'blur' }],

age: [{ required: true, message: '请输入年龄', trigger: 'blur' }],

id\_card: [

{ required: true, message: '请输入身份证号', trigger: 'blur' },

{ pattern: /^\d{18}$/, message: '请输入18位身份证号', trigger: 'blur' }

],

care\_level: [{ required: true, message: '请选择护理等级', trigger: 'change' }]

};

watch(

() => props.initialValues,

(val) => {

Object.assign(form, val);

},

{ immediate: true }

);

// 取消操作

const handleCancel = () => {

emit('cancel');

formRef.value.resetFields(); // 重置表单

};

// 提交表单

const handleOk = async () => {

// 表单验证

const valid = await formRef.value.validate();

if (!valid) return;

try {

if (form.id) {

// 编辑操作

await updateElderly(form.id, form);

} else {

// 新增操作

await addElderly(form);

}

emit('submit'); // 通知父组件提交成功

} catch (error) {

console.error('提交失败：', error);

}

};

return {

formRef,

form,

rules,

handleCancel,

handleOk

};

}

});

</script>

API 请求封装（api/elderly.js）

import axios from 'axios';

import { baseUrl } from '@/constants'; // 基础API地址

// 获取老人列表

export const getElderlyList = (params) => {

return axios.get(`${baseUrl}/elderlies/`, { params });

};

// 新增老人

export const addElderly = (data) => {

return axios.post(`${baseUrl}/elderlies/`, data);

};

// 更新老人信息

export const updateElderly = (id, data) => {

return axios.put(`${baseUrl}/elderlies/${id}/`, data);

};

// 删除老人

export const deleteElderly = (id) => {

return axios.delete(`${baseUrl}/elderlies/${id}/`);

};

护工管理模块（Nurse Management）

模型定义（models.py）

from django.db import models

from django.utils import timezone

class Nurse(models.Model):

"""护工信息模型"""

GENDER\_CHOICES = (

(0, '女'),

(1, '男'),

)

STATUS\_CHOICES = (

(0, '离职'),

(1, '在职'),

)

name = models.CharField(max\_length=50, verbose\_name="姓名")

gender = models.IntegerField(choices=GENDER\_CHOICES, verbose\_name="性别")

phone = models.CharField(max\_length=20, verbose\_name="联系电话")

hire\_date = models.DateField(verbose\_name="入职日期")

certificate = models.CharField(max\_length=100, blank=True, null=True, verbose\_name="资质证书")

specialty = models.CharField(max\_length=100, blank=True, null=True, verbose\_name="擅长领域")

status = models.IntegerField(choices=STATUS\_CHOICES, default=1, verbose\_name="状态")

# 多对多关联老人（一个护工负责多个老人，一个老人可由多个护工照护）

elderly = models.ManyToManyField('Elderly', related\_name='nurses', blank=True, verbose\_name="负责老人")

create\_time = models.DateTimeField(default=timezone.now, verbose\_name="创建时间")

update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")

class Meta:

verbose\_name = "护工信息"

verbose\_name\_plural = "护工信息管理"

ordering = ['-create\_time']

def \_\_str\_\_(self):

return f"{self.name}（{self.get\_status\_display()}）"

序列化器（serializers.py）

from rest\_framework import serializers

from .models import Nurse

from .models import Elderly # 导入老人模型用于关联展示

class NurseSerializer(serializers.ModelSerializer):

"""护工信息序列化器"""

gender\_display = serializers.CharField(source='get\_gender\_display', read\_only=True) # 性别中文

status\_display = serializers.CharField(source='get\_status\_display', read\_only=True) # 状态中文

# 负责老人的姓名列表

elderly\_names = serializers.SerializerMethodField()

class Meta:

model = Nurse

fields = [

'id', 'name', 'gender', 'gender\_display', 'phone', 'hire\_date',

'certificate', 'specialty', 'status', 'status\_display',

'elderly', 'elderly\_names', 'create\_time', 'update\_time'

]

read\_only\_fields = ['id', 'create\_time', 'update\_time']

def get\_elderly\_names(self, obj):

"""获取负责老人的姓名列表"""

return [elderly.name for elderly in obj.elderly.all()]

def validate\_phone(self, value):

"""验证手机号格式"""

if not (len(value) == 11 and value.isdigit()):

raise serializers.ValidationError("手机号必须为11位数字")

return value

视图（views.py）

from rest\_framework import viewsets, filters

from django\_filters.rest\_framework import DjangoFilterBackend

from .models import Nurse

from .serializers import NurseSerializer

class NurseViewSet(viewsets.ModelViewSet):

"""护工信息视图集"""

queryset = Nurse.objects.all()

serializer\_class = NurseSerializer

# 支持按状态、性别过滤，按姓名/证书搜索

filter\_backends = [DjangoFilterBackend, filters.SearchFilter, filters.OrderingFilter]

filterset\_fields = ['status', 'gender']

search\_fields = ['name', 'certificate', 'specialty']

ordering\_fields = ['hire\_date', 'create\_time']

路由配置（urls.py）

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from .views import NurseViewSet

router = DefaultRouter()

router.register(r'nurses', NurseViewSet) # 护工接口路径：/api/nurses/

urlpatterns = [

path('api/', include(router.urls)),

]

护工列表组件（NurseList.vue）

<template>

<div class="nurse-list-container">

<a-card title="护工信息管理" :bordered="false">

<div style="margin-bottom: 16px">

<a-button type="primary" @click="showAddModal">

<plus-outlined /> 新增护工

</a-button>

<a-select

placeholder="按状态筛选"

style="width: 150px; margin-left: 16px"

@change="handleStatusChange"

allow-clear

>

<a-select-option value="1">在职</a-select-option>

<a-select-option value="0">离职</a-select-option>

</a-select>

</div>

<!-- 护工列表表格 -->

<a-table

:columns="columns"

:data-source="dataSource"

:loading="loading"

rowKey="id"

:pagination="pagination"

@change="handleTableChange"

>

<template #specialty="{ record }">

<tag color="blue">{{ record.specialty || '无' }}</tag>

</template>

<template #operation="{ record }">

<a-button type="link" @click="showEditModal(record)">编辑</a-button>

<a-button type="link" @click="showElderlyModal(record)">负责老人</a-button>

<a-popconfirm

title="确定删除该护工信息吗？"

@confirm="handleDelete(record.id)"

>

<a-button type="link" danger>删除</a-button>

</a-popconfirm>

</template>

</a-table>

</a-card>

<!-- 新增/编辑弹窗 -->

<nurse-modal

:visible="modalVisible"

:initial-values="formData"

:title="modalTitle"

@cancel="modalVisible = false"

@submit="handleSubmit"

/>

<!-- 负责老人弹窗 -->

<nurse-elderly-modal

:visible="elderlyModalVisible"

:nurse-id="currentNurseId"

@cancel="elderlyModalVisible = false"

/>

</div>

</template>

<script>

import { PlusOutlined } from '@ant-design/icons-vue';

import { Tag } from 'ant-design-vue';

import { defineComponent, reactive, toRefs, onMounted } from 'vue';

import NurseModal from './NurseModal.vue';

import NurseElderlyModal from './NurseElderlyModal.vue';

import { getNurseList, deleteNurse } from '@/api/nurse';

export default defineComponent({

components: { NurseModal, NurseElderlyModal, PlusOutlined, Tag },

setup() {

const state = reactive({

dataSource: [],

loading: false,

modalVisible: false,

elderlyModalVisible: false,

formData: {},

currentNurseId: null,

modalTitle: '新增护工',

pagination: {

current: 1,

pageSize: 10,

total: 0,

showSizeChanger: true

},

statusFilter: null

});

// 表格列配置

const columns = [

{ title: '姓名', dataIndex: 'name', key: 'name' },

{ title: '性别', dataIndex: 'gender\_display', key: 'gender' },

{ title: '电话', dataIndex: 'phone', key: 'phone' },

{ title: '入职日期', dataIndex: 'hire\_date', key: 'hire\_date' },

{ title: '擅长领域', key: 'specialty', slots: { customRender: 'specialty' } },

{ title: '状态', dataIndex: 'status\_display', key: 'status' },

{ title: '操作', key: 'operation', slots: { customRender: 'operation' } }

];

// 获取护工列表

const fetchData = async () => {

state.loading = true;

try {

const params = {

page: state.pagination.current,

page\_size: state.pagination.pageSize,

status: state.statusFilter

};

const res = await getNurseList(params);

state.dataSource = res.results;

state.pagination.total = res.count;

} catch (error) {

console.error('获取护工数据失败：', error);

} finally {

state.loading = false;

}

};

onMounted(fetchData);

// 状态筛选

const handleStatusChange = (value) => {

state.statusFilter = value;

state.pagination.current = 1;

fetchData();

};

// 分页变化

const handleTableChange = (pagination) => {

state.pagination = pagination;

fetchData();

};

// 新增/编辑弹窗

const showAddModal = () => {

state.formData = {};

state.modalTitle = '新增护工';

state.modalVisible = true;

};

const showEditModal = (record) => {

state.formData = { ...record };

state.modalTitle = '编辑护工';

state.modalVisible = true;

};

// 负责老人弹窗

const showElderlyModal = (record) => {

state.currentNurseId = record.id;

state.elderlyModalVisible = true;

};

// 提交表单

const handleSubmit = () => {

state.modalVisible = false;

fetchData();

};

// 删除护工

const handleDelete = async (id) => {

try {

await deleteNurse(id);

fetchData();

} catch (error) {

console.error('删除失败：', error);

}

};

return {

...toRefs(state),

columns,

handleStatusChange,

handleTableChange,

showAddModal,

showEditModal,

showElderlyModal,

handleSubmit,

handleDelete

};

}

});

</script>

护工表单弹窗（NurseModal.vue）

<template>

<a-modal

:visible="visible"

:title="title"

@cancel="handleCancel"

@ok="handleOk"

>

<a-form

:model="form"

:rules="rules"

ref="formRef"

layout="vertical"

>

<a-form-item name="name" label="姓名">

<a-input v-model:value="form.name" placeholder="请输入姓名" />

</a-form-item>

<a-form-item name="gender" label="性别">

<a-radio-group v-model:value="form.gender">

<a-radio :value="1">男</a-radio>

<a-radio :value="0">女</a-radio>

</a-radio-group>

</a-form-item>

<a-form-item name="phone" label="联系电话">

<a-input v-model:value="form.phone" placeholder="请输入11位手机号" />

</a-form-item>

<a-form-item name="hire\_date" label="入职日期">

<a-date-picker v-model:value="form.hire\_date" format="YYYY-MM-DD" />

</a-form-item>

<a-form-item name="specialty" label="擅长领域">

<a-input v-model:value="form.specialty" placeholder="如：失能照护、康复辅助" />

</a-form-item>

<a-form-item name="status" label="状态">

<a-radio-group v-model:value="form.status">

<a-radio :value="1">在职</a-radio>

<a-radio :value="0">离职</a-radio>

</a-radio-group>

</a-form-item>

</a-form>

</a-modal>

</template>

<script>

import { defineComponent, reactive, toRefs, ref, watch } from 'vue';

import { addNurse, updateNurse } from '@/api/nurse';

export default defineComponent({

props: {

visible: { type: Boolean, default: false },

initialValues: { type: Object, default: () => ({}) },

title: { type: String, default: '' }

},

emits: ['cancel', 'submit'],

setup(props, { emit }) {

const formRef = ref(null);

const form = reactive({

name: '',

gender: 1,

phone: '',

hire\_date: null,

specialty: '',

status: 1

});

const rules = {

name: [{ required: true, message: '请输入姓名', trigger: 'blur' }],

phone: [

{ required: true, message: '请输入手机号', trigger: 'blur' },

{ pattern: /^1\d{10}$/, message: '请输入正确的手机号', trigger: 'blur' }

],

hire\_date: [{ required: true, message: '请选择入职日期', trigger: 'change' }]

};

// 回显数据

watch(() => props.initialValues, (val) => {

Object.assign(form, val);

}, { immediate: true });

const handleCancel = () => {

emit('cancel');

formRef.value.resetFields();

};

const handleOk = async () => {

const valid = await formRef.value.validate();

if (!valid) return;

try {

if (form.id) {

await updateNurse(form.id, form);

} else {

await addNurse(form);

}

emit('submit');

} catch (error) {

console.error('提交失败：', error);

}

};

return {

formRef,

form,

rules,

handleCancel,

handleOk

};

}

});

</script>

护工 - 老人关联弹窗（NurseElderlyModal.vue，管理护工负责的老人）

<template>

<a-modal

title="关联负责老人"

:visible="visible"

@cancel="handleCancel"

@ok="handleSave"

width="700px"

>

<a-transfer

:data-source="allElderlies"

:target-keys="selectedElderlyIds"

:titles="['可选老人', '已选老人']"

:render="renderElderly"

@change="handleTransferChange"

/>

</a-modal>

</template>

<script>

import { defineComponent, reactive, toRefs, onMounted, ref } from 'vue';

import { getElderlyList } from '@/api/elderly';

import { getNurseElderlies, updateNurseElderlies } from '@/api/nurse';

export default defineComponent({

props: {

visible: { type: Boolean, default: false },

nurseId: { type: Number, required: true }

},

emits: ['cancel'],

setup(props, { emit }) {

const state = reactive({

allElderlies: [], // 所有老人列表

selectedElderlyIds: [] // 已选老人ID

});

// 渲染老人显示内容

const renderElderly = (item) => {

return `${item.name}（${item.age}岁）`;

};

// 获取所有老人

const fetchAllElderlies = async () => {

try {

const res = await getElderlyList({ page\_size: 100 }); // 分页取100条

state.allElderlies = res.results.map(item => ({

key: item.id,

...item

}));

} catch (error) {

console.error('获取老人列表失败：', error);

}

};

// 获取护工已关联的老人

const fetchSelectedElderlies = async () => {

try {

const res = await getNurseElderlies(props.nurseId);

state.selectedElderlyIds = res.elderly.map(item => item.id);

} catch (error) {

console.error('获取关联老人失败：', error);

}

};

// 初始化数据

onMounted(() => {

fetchAllElderlies();

if (props.nurseId) {

fetchSelectedElderlies();

}

});

// 转移选择变化

const handleTransferChange = (targetKeys) => {

state.selectedElderlyIds = targetKeys;

};

// 保存关联关系

const handleSave = async () => {

try {

await updateNurseElderlies(props.nurseId, { elderly: state.selectedElderlyIds });

emit('cancel');

} catch (error) {

console.error('保存关联失败：', error);

}

};

const handleCancel = () => {

emit('cancel');

};

return {

...toRefs(state),

renderElderly,

handleTransferChange,

handleSave,

handleCancel

};

}

});

</script>

API 封装（api/nurse.js）

import axios from 'axios';

import { baseUrl } from '@/constants';

// 获取护工列表

export const getNurseList = (params) => {

return axios.get(`${baseUrl}/nurses/`, { params });

};

// 新增护工

export const addNurse = (data) => {

return axios.post(`${baseUrl}/nurses/`, data);

};

// 更新护工

export const updateNurse = (id, data) => {

return axios.put(`${baseUrl}/nurses/${id}/`, data);

};

// 删除护工

export const deleteNurse = (id) => {

return axios.delete(`${baseUrl}/nurses/${id}/`);

};

// 获取护工关联的老人

export const getNurseElderlies = (nurseId) => {

return axios.get(`${baseUrl}/nurses/${nurseId}/`);

};

// 更新护工关联的老人

export const updateNurseElderlies = (nurseId, data) => {

return axios.patch(`${baseUrl}/nurses/${nurseId}/`, data);

};

房间管理模块（Room Management）

模型定义（models.py）

from django.db import models

from django.utils import timezone

class Room(models.Model):

"""房间信息模型"""

STATUS\_CHOICES = (

(0, '停用'),

(1, '在用'),

)

room\_no = models.CharField(max\_length=20, unique=True, verbose\_name="房间号") # 房间号（唯一）

type = models.CharField(max\_length=30, blank=True, null=True, verbose\_name="房间类型") # 单人间/双人间

bed\_count = models.IntegerField(verbose\_name="总床位数") # 总床位数

available\_bed = models.IntegerField(verbose\_name="剩余床位数") # 剩余床位数

facilities = models.TextField(blank=True, null=True, verbose\_name="设施配置") # 设施描述

status = models.IntegerField(choices=STATUS\_CHOICES, default=1, verbose\_name="状态") # 在用/停用

create\_time = models.DateTimeField(default=timezone.now, verbose\_name="创建时间")

update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")

class Meta:

verbose\_name = "房间信息"

verbose\_name\_plural = "房间信息管理"

ordering = ['room\_no'] # 按房间号排序

def \_\_str\_\_(self):

return f"{self.room\_no}（{self.type}）"

def clean(self):

"""验证剩余床位数不能大于总床位数"""

if self.available\_bed > self.bed\_count:

from django.core.exceptions import ValidationError

raise ValidationError("剩余床位数不能大于总床位数")

序列化器（serializers.py）

from rest\_framework import serializers

from .models import Room

class RoomSerializer(serializers.ModelSerializer):

"""房间信息序列化器"""

status\_display = serializers.CharField(source='get\_status\_display', read\_only=True)

# 入住人数（总床位 - 剩余床位）

occupied\_count = serializers.SerializerMethodField()

# 关联的老人数量（通过反向查询）

elderly\_count = serializers.SerializerMethodField()

class Meta:

model = Room

fields = [

'id', 'room\_no', 'type', 'bed\_count', 'available\_bed',

'occupied\_count', 'elderly\_count', 'facilities',

'status', 'status\_display', 'create\_time', 'update\_time'

]

read\_only\_fields = ['id', 'create\_time', 'update\_time']

def get\_occupied\_count(self, obj):

"""计算已占用床位数"""

return obj.bed\_count - obj.available\_bed

def get\_elderly\_count(self, obj):

"""获取房间内的老人数量"""

return obj.elderlies.count() # 反向关联查询（老人表中room字段关联到房间）

def validate(self, data):

"""验证剩余床位数"""

if data.get('available\_bed') is not None and data.get('bed\_count') is not None:

if data['available\_bed'] > data['bed\_count']:

raise serializers.ValidationError("剩余床位数不能大于总床位数")

return data

视图（views.py）

from rest\_framework import viewsets, filters

from django\_filters.rest\_framework import DjangoFilterBackend

from .models import Room

from .serializers import RoomSerializer

class RoomViewSet(viewsets.ModelViewSet):

"""房间信息视图集"""

queryset = Room.objects.all()

serializer\_class = RoomSerializer

filter\_backends = [DjangoFilterBackend, filters.SearchFilter, filters.OrderingFilter]

filterset\_fields = ['status', 'type'] # 按状态、类型筛选

search\_fields = ['room\_no', 'facilities'] # 按房间号、设施搜索

ordering\_fields = ['room\_no', 'bed\_count', 'create\_time']

路由配置（urls.py）

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from .views import RoomViewSet

router = DefaultRouter()

router.register(r'rooms', RoomViewSet) # 房间接口路径：/api/rooms/

urlpatterns = [

path('api/', include(router.urls)),

]

房间列表组件（RoomList.vue）

<template>

<div class="room-list-container">

<a-card title="房间信息管理" :bordered="false">

<div style="margin-bottom: 16px">

<a-button type="primary" @click="showAddModal">

<plus-outlined /> 新增房间

</a-button>

<a-select

placeholder="按状态筛选"

style="width: 150px; margin-left: 16px"

@change="handleStatusChange"

allow-clear

>

<a-select-option value="1">在用</a-select-option>

<a-select-option value="0">停用</a-select-option>

</a-select>

</div>

<!-- 房间卡片列表（直观展示床位状态） -->

<div class="room-cards">

<a-card

v-for="room in dataSource"

:key="room.id"

:title="`${room.room\_no}（${room.type || '未知类型'}）`"

:bordered="true"

class="room-card"

:style="{ background: room.status === 0 ? '#f5f5f5' : 'white' }"

>

<p>总床位：{{ room.bed\_count }} 张</p>

<p>剩余床位：<span :style="{ color: room.available\_bed === 0 ? 'red' : 'green' }">

{{ room.available\_bed }} 张

</span></p>

<p>入住人数：{{ room.occupied\_count }} 人</p>

<p>状态：<tag :color="room.status === 1 ? 'green' : 'orange'">

{{ room.status\_display }}

</tag></p>

<div class="room-actions">

<a-button type="link" @click="showEditModal(room)">编辑</a-button>

<a-button type="link" @click="showElderlyList(room.id)">查看入住老人</a-button>

</div>

</a-card>

</div>

</a-card>

<!-- 新增/编辑弹窗 -->

<room-modal

:visible="modalVisible"

:initial-values="formData"

:title="modalTitle"

@cancel="modalVisible = false"

@submit="handleSubmit"

/>

<!-- 房间老人列表弹窗 -->

<room-elderly-modal

:visible="elderlyModalVisible"

:room-id="currentRoomId"

@cancel="elderlyModalVisible = false"

/>

</div>

</template>

<script>

import { PlusOutlined } from '@ant-design/icons-vue';

import { Tag } from 'ant-design-vue';

import { defineComponent, reactive, toRefs, onMounted } from 'vue';

import RoomModal from './RoomModal.vue';

import RoomElderlyModal from './RoomElderlyModal.vue';

import { getRoomList } from '@/api/room';

export default defineComponent({

components: { RoomModal, RoomElderlyModal, PlusOutlined, Tag },

setup() {

const state = reactive({

dataSource: [],

loading: false,

modalVisible: false,

elderlyModalVisible: false,

formData: {},

currentRoomId: null,

modalTitle: '新增房间',

statusFilter: null

});

// 获取房间列表

const fetchData = async () => {

state.loading = true;

try {

const params = { status: state.statusFilter };

const res = await getRoomList(params);

state.dataSource = res.results;

} catch (error) {

console.error('获取房间数据失败：', error);

} finally {

state.loading = false;

}

};

onMounted(fetchData);

// 状态筛选

const handleStatusChange = (value) => {

state.statusFilter = value;

fetchData();

};

// 新增/编辑弹窗

const showAddModal = () => {

state.formData = {};

state.modalTitle = '新增房间';

state.modalVisible = true;

};

const showEditModal = (record) => {

state.formData = { ...record };

state.modalTitle = '编辑房间';

state.modalVisible = true;

};

// 查看入住老人

const showElderlyList = (roomId) => {

state.currentRoomId = roomId;

state.elderlyModalVisible = true;

};

// 提交表单

const handleSubmit = () => {

state.modalVisible = false;

fetchData();

};

return {

...toRefs(state),

handleStatusChange,

showAddModal,

showEditModal,

showElderlyList,

handleSubmit

};

}

});

</script>

<style scoped>

.room-list-container {

padding: 20px;

}

.room-cards {

display: grid;

grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));

gap: 16px;

margin-top: 16px;

}

.room-card {

height: 100%;

display: flex;

flex-direction: column;

}

.room-actions {

margin-top: 16px;

align-self: flex-end;

}

</style>

房间表单弹窗（RoomModal.vue）

<template>

<a-modal

:visible="visible"

:title="title"

@cancel="handleCancel"

@ok="handleOk"

>

<a-form

:model="form"

:rules="rules"

ref="formRef"

layout="vertical"

>

<a-form-item name="room\_no" label="房间号">

<a-input v-model:value="form.room\_no" placeholder="如：101、202" />

</a-form-item>

<a-form-item name="type" label="房间类型">

<a-select v-model:value="form.type" placeholder="请选择房间类型">

<a-select-option value="单人间">单人间</a-select-option>

<a-select-option value="双人间">双人间</a-select-option>

<a-select-option value="多人间">多人间</a-select-option>

<a-select-option value="护理间">护理间</a-select-option>

</a-select>

</a-form-item>

<a-form-item name="bed\_count" label="总床位数">

<a-input-number

v-model:value="form.bed\_count"

:min="1"

:step="1"

placeholder="请输入总床位数"

/>

</a-form-item>

<a-form-item name="available\_bed" label="剩余床位数">

<a-input-number

v-model:value="form.available\_bed"

:min="0"

:step="1"

placeholder="请输入剩余床位数"

/>

</a-form-item>

<a-form-item name="facilities" label="设施配置">

<a-textarea

v-model:value="form.facilities"

rows="4"

placeholder="请输入房间设施配置，如：独立卫生间、呼叫器、空调等"

/>

</a-form-item>

<a-form-item name="status" label="状态">

<a-radio-group v-model:value="form.status">

<a-radio :value="1">在用</a-radio>

<a-radio :value="0">停用</a-radio>

</a-radio-group>

</a-form-item>

</a-form>

</a-modal>

</template>

<script>

import { defineComponent, reactive, toRefs, ref, watch } from 'vue';

import { addRoom, updateRoom } from '@/api/room';

export default defineComponent({

props: {

visible: { type: Boolean, default: false },

initialValues: { type: Object, default: () => ({}) },

title: { type: String, default: '' }

},

emits: ['cancel', 'submit'],

setup(props, { emit }) {

const formRef = ref(null);

const form = reactive({

room\_no: '',

type: '',

bed\_count: 1,

available\_bed: 1,

facilities: '',

status: 1

});

const rules = {

room\_no: [

{ required: true, message: '请输入房间号', trigger: 'blur' },

{ pattern: /^[0-9A-Za-z]+$/, message: '房间号只能包含数字和字母', trigger: 'blur' }

],

type: [{ required: true, message: '请选择房间类型', trigger: 'change' }],

bed\_count: [{ required: true, message: '请输入总床位数', trigger: 'blur' }],

available\_bed: [{ required: true, message: '请输入剩余床位数', trigger: 'blur' }]

};

// 回显数据

watch(() => props.initialValues, (val) => {

Object.assign(form, val);

}, { immediate: true });

// 监听总床位数变化，自动调整剩余床位数

watch(() => form.bed\_count, (val) => {

if (form.available\_bed > val) {

form.available\_bed = val;

}

});

const handleCancel = () => {

emit('cancel');

formRef.value.resetFields();

};

const handleOk = async () => {

const valid = await formRef.value.validate();

if (!valid) return;

// 验证剩余床位数不能大于总床位数

if (form.available\_bed > form.bed\_count) {

message.error('剩余床位数不能大于总床位数');

return;

}

try {

if (form.id) {

await updateRoom(form.id, form);

} else {

await addRoom(form);

}

emit('submit');

} catch (error) {

console.error('提交失败：', error);

message.error(error.response?.data?.detail || '操作失败，请重试');

}

};

return {

formRef,

form,

rules,

handleCancel,

handleOk

};

}

});

</script>

护理计划模块（Nursing Plan）

from django.db import models

from django.utils import timezone

class NursingPlan(models.Model):

"""护理计划模型"""

elderly = models.ForeignKey('Elderly', on\_delete=models.CASCADE, related\_name='nursing\_plans', verbose\_name="老人")

nurse = models.ForeignKey('Nurse', on\_delete=models.SET\_NULL, null=True, related\_name='nursing\_plans', verbose\_name="护工")

title = models.CharField(max\_length=100, verbose\_name="计划标题")

content = models.TextField(verbose\_name="计划内容")

start\_date = models.DateField(verbose\_name="开始日期")

end\_date = models.DateField(null=True, blank=True, verbose\_name="结束日期")

is\_completed = models.BooleanField(default=False, verbose\_name="是否完成")

create\_time = models.DateTimeField(default=timezone.now, verbose\_name="创建时间")

update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")

class Meta:

verbose\_name = "护理计划"

verbose\_name\_plural = "护理计划管理"

ordering = ['-create\_time']

def \_\_str\_\_(self):

return f"{self.elderly.name}的护理计划：{self.title}"

健康记录模块（Health Record）

from django.db import models

from django.utils import timezone

class HealthRecord(models.Model):

"""健康记录模型"""

elderly = models.ForeignKey('Elderly', on\_delete=models.CASCADE, related\_name='health\_records', verbose\_name="老人")

record\_date = models.DateTimeField(verbose\_name="记录时间")

temperature = models.DecimalField(max\_digits=4, decimal\_places=1, null=True, blank=True, verbose\_name="体温(℃)")

blood\_pressure = models.CharField(max\_length=20, null=True, blank=True, verbose\_name="血压")

heart\_rate = models.IntegerField(null=True, blank=True, verbose\_name="心率(次/分)")

symptom = models.TextField(null=True, blank=True, verbose\_name="症状描述")

treatment = models.TextField(null=True, blank=True, verbose\_name="处理措施")

recorded\_by = models.ForeignKey('Nurse', on\_delete=models.SET\_NULL, null=True, verbose\_name="记录人")

create\_time = models.DateTimeField(default=timezone.now, verbose\_name="创建时间")

update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")

class Meta:

verbose\_name = "健康记录"

verbose\_name\_plural = "健康记录管理"

ordering = ['-record\_date']

def \_\_str\_\_(self):

return f"{self.elderly.name}的健康记录({self.record\_date.strftime('%Y-%m-%d')})"