/\* eslint-disable react-hooks/exhaustive-deps \*/  
  
/\* eslint-disable no-unused-vars \*/  
  
import React, { useCallback, useEffect, useRef, useState } from 'react';  
  
import './map.less';  
  
import api from '@/desktop/model';  
  
import { Request } from '@/utils';  
  
import greenImg from './img/p-cyan.png';  
  
import orangeImg from './img/p-orange.png';  
  
import grayImg from './img/p-gray.png';  
  
import blueImg from './img/p-blue.png';  
  
import { TreeSelect } from 'antd';  
  
import SearchInput from './searchInput';  
  
import { gcj02towgs84 } from '@/utils/gis';  
  
   
  
const YichangMap = () => {  
  
 const [allList, setAllList] = useState([]);  
  
 const [list, setList] = useState([]);  
  
 const [mapScene, setMapScene] = useState(null);  
  
 const [mapViewer, setMapViewer] = useState(null);  
  
 const [regionData, setRegionData] = useState();  
  
 const [regionLabel, setRegionLabel] = useState(null);  
  
 const [filingData, setFilingData] = useState();  
  
 const [filingLabel, setFilingLabel] = useState(null);  
  
 const [typeData, setTypeData] = useState();  
  
 const [typeLabel, setTypeLabel] = useState(null);  
  
 const [connectData, setConnectData] = useState();  
  
 const [connectLabel, setConnectLabel] = useState(null);  
  
   
  
 const [selected, setSelected] = useState(0);  
  
   
  
 const regionTreeData = [  
  
​ {  
  
​ title: '西陵区',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '伍家岗区',  
  
​ value: '0-1',  
  
​ },  
  
​ {  
  
​ title: '点军区',  
  
​ value: '0-2',  
  
​ },  
  
​ {  
  
​ title: '猇亭区',  
  
​ value: '0-3',  
  
​ },  
  
​ {  
  
​ title: '夷陵区',  
  
​ value: '0-4',  
  
​ },  
  
​ {  
  
​ title: '远安县',  
  
​ value: '0-5',  
  
​ },  
  
​ {  
  
​ title: '兴山县',  
  
​ value: '0-6',  
  
​ },  
  
​ {  
  
​ title: '秭归县',  
  
​ value: '0-7',  
  
​ },  
  
​ {  
  
​ title: '长阳土家族自治县',  
  
​ value: '0-8',  
  
​ },  
  
​ {  
  
​ title: '五峰土家族自治县',  
  
​ value: '0-9',  
  
​ },  
  
​ {  
  
​ title: '枝江市',  
  
​ value: '0-10',  
  
​ },  
  
​ {  
  
​ title: '宜都市',  
  
​ value: '0-11',  
  
​ },  
  
​ {  
  
​ title: '当阳市',  
  
​ value: '0-12',  
  
​ },  
  
 ];  
  
 const filingTreeData = [  
  
​ {  
  
​ title: '未备案',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '备案审核中',  
  
​ value: '0-1',  
  
​ },  
  
​ {  
  
​ title: '备案通过',  
  
​ value: '0-2',  
  
​ },  
  
​ {  
  
​ title: '备案未通过',  
  
​ value: '0-3',  
  
​ },  
  
 ];  
  
 const typeTreeData = [  
  
​ {  
  
​ title: '公共停车场',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '配建停车场',  
  
​ value: '0-1',  
  
​ },  
  
​ {  
  
​ title: '路内停车场',  
  
​ value: '0-2',  
  
​ },  
  
 ];  
  
 const connectTreeData = [  
  
​ {  
  
​ title: '已联网',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '未联网',  
  
​ value: '0-1',  
  
​ },  
  
 ];  
  
 const containerRef = useRef();  
  
 // 创建一个View  
  
 const restUrl = 'http://10.55.49.27:8884/gtis/map/rest/69dbfedc855347eda29a2eb2011d4c48/rest/maps/dianziditu\_zhuji';  
  
   
  
 const onRegionChange = useCallback((value, label) => {  
  
​ setRegionData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setRegionLabel(null);  
  
​ } else {  
  
​ setRegionLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onfilingChange = useCallback((value, label) => {  
  
​ setFilingData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setFilingLabel(null);  
  
​ } else {  
  
​ setFilingLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onTypeChange = useCallback((value, label) => {  
  
​ setTypeData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setTypeLabel(null);  
  
​ } else {  
  
​ setTypeLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onConnectChange = useCallback((value, label) => {  
  
​ setConnectData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setConnectLabel(null);  
  
​ } else {  
  
​ setConnectLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onSelect = (value, option) => {  
  
​ setList([option.park]);  
  
​ setSelected(1);  
  
 };  
  
   
  
 const onClear = () => {  
  
​ setList(allList);  
  
​ setSelected(0);  
  
 };  
  
   
  
 const getPictureUrl = (item, item2, name) => {  
  
​ let result = item?.recordIds?.map(() => {  
  
​ return item2?.[name];  
  
​ });  
  
​ result = result  
  
​ ?.map(item => {  
  
​ return Array.isArray(item)  
  
​ ? item.map(pic => {  
  
​ if (pic?.type && pic.type.indexOf('image') > -1) {  
  
​ return pic?.url;  
  
​ } else {  
  
​ return '';  
  
​ }  
  
​ })  
  
​ : '';  
  
​ })  
  
​ .flat(Infinity);  
  
​ return result;  
  
 };  
  
   
  
 useEffect(() => {  
  
​ const initMap = () => {  
  
​ let link, script;  
  
​ link = document.createElement('link');  
  
​ link.type = 'text/css';  
  
​ link.rel = 'stylesheet';  
  
​ link.href = `${window.\_pageURL}/static/map/Widgets/widgets.css`;  
  
​ document.head.appendChild(link);  
  
​ script = document.createElement('script');  
  
​ script.type = 'text/javascript';  
  
​ script.src = `${window.\_pageURL}/static/map/Cesium.js`;  
  
​ document.head.appendChild(script);  
  
​ script.onload = () => {  
  
​ const div = document.querySelector('.cesium-viewer');  
  
​ if (div) {  
  
​ return;  
  
​ }  
  
​ const viewer = new window.Cesium.Viewer('cesiumContainer', {  
  
​ sceneMode: window.Cesium.SceneMode.SCENE3D, //设置为三维地图  
  
​ });  
  
​ setMapViewer(viewer);  
  
​ const scene = viewer.scene;  
  
​ setMapScene(scene);  
  
​ // 加载图层  
  
​ // eslint-disable-next-line no-unused-vars  
  
​ const layer = viewer.imageryLayers.addImageryProvider(  
  
​ new window.Cesium.SuperMapImageryProvider({  
  
​ url: restUrl,  
  
​ })  
  
​ );  
  
​ const initPosition = window.Cesium.Cartesian3.fromDegrees(111.293779, 30.650459, 4000);  
  
​ scene.camera.setView({  
  
​ destination: initPosition, //位置  
  
​ orientation: {  
  
​ pitch: -0.6,  
  
​ },  
  
​ endTransform: window.Cesium.Matrix4.IDENTITY,  
  
​ });  
  
​ };  
  
​ };  
  
​ initMap();  
  
​ return () => {  
  
​ if (mapViewer) {  
  
​ mapViewer.destroy();  
  
​ }  
  
​ };  
  
​ // eslint-disable-next-line react-hooks/exhaustive-deps  
  
 }, []);  
  
   
  
 const getInfo = items => {  
  
​ let info = '<div class="content-wrapper"><div class="item-container">';  
  
​ items.forEach(item => {  
  
​ const { label, value } = item;  
  
​ if (label === '详细地址' || label === '车场业态' || label === '备案状态' || label === '停车场类型' || label === '充电泊位数') {  
  
​ if (value) {  
  
​ info = `${info}<div class="item"><span class="item-key">${label}: </span><span class="item-value">${  
  
​ value?.text  
  
​ ? Array.isArray(value?.text)  
  
​ ? value?.text?.join(',')  
  
​ : value?.text?.address || value?.text?.text  
  
​ ? value?.text?.address || value?.text?.text  
  
​ : value?.text  
  
​ : Array.isArray(value)  
  
​ ? value?.[0]?.url || value.join('/')  
  
​ : value || ''  
  
​ }</span></div>`;  
  
​ }  
  
​ }  
  
​ if (label === '是否联网接入宜昌智慧停车平台') {  
  
​ info = `${info}<div class="item"><span class="item-key">联网状态: </span><span class="item-value">${  
  
​ value === '是' ? '已联网' : '未联网'  
  
​ }</span></div>`;  
  
​ }  
  
​ });  
  
​ info = `${info}</div>`;  
  
​ info = `${info}<div class="num-container">`;  
  
​ items.forEach(item => {  
  
​ const { label, value } = item;  
  
​ if (label === '泊位总数' || label === '对外开放泊位个数') {  
  
​ info = `${info}<div class="num-item"><div class="item-value">${value}</div><div class="item-label">${label}</div></div>`;  
  
​ }  
  
​ });  
  
​ info = `${info}</div>`;  
  
​ items.forEach(item => {  
  
​ const { label, value } = item;  
  
​ if (label === '现场勘验视频图片' && value?.[0]) {  
  
​ info = `${info}  
  
 <div class="img-container">  
  
​ <img src="${window.\_baseURL}${value[0]}" alt="img" imgUrl="${window.\_baseURL}${value[0]}"/>  
  
​ </div>  
  
​ `;  
  
​ }  
  
​ });  
  
​ info = `${info}</div>`;  
  
​ return info;  
  
 };  
  
   
  
 const getFilingImg = (status, connectStatus) => {  
  
​ if (status === '备案通过' && connectStatus === '是') {  
  
​ return greenImg;  
  
​ } else if (status === '备案通过' && connectStatus === '否') {  
  
​ return orangeImg;  
  
​ } else if ((status === '备案未通过' || status === '备案审核中' || status === '未备案') && connectStatus === '是') {  
  
​ return blueImg;  
  
​ } else {  
  
​ return grayImg;  
  
​ }  
  
 };  
  
   
  
 const getMarker = (list, rows, viewer) => {  
  
​ const array = list.map(item => {  
  
​ try {  
  
​ const obj = { ...item };  
  
​ let latitude = 0,  
  
​ longitude = 0,  
  
​ parkName = '',  
  
​ filingStatus = '',  
  
​ parkType = '',  
  
​ region = [],  
  
​ recordIds = [],  
  
​ spaces = 0,  
  
​ outSpaces = 0,  
  
​ connectStatus = '',  
  
​ chargeSpaces = 0;  
  
​ for (const key in item) {  
  
​ if (Object.hasOwnProperty.call(item, key)) {  
  
​ if (/latitude/.test(key) || obj[key]?.text?.latitude) {  
  
​ latitude = obj[key]?.text?.latitude || obj[key];  
  
​ } else if (/longitude/.test(key) || obj[key]?.text?.longitude) {  
  
​ longitude = obj[key]?.text?.longitude || obj[key];  
  
​ }  
  
​ item[key] = {  
  
​ name: key,  
  
​ value: item[key],  
  
​ label: rows.filter(row => row.name === key)?.[0]?.label,  
  
​ };  
  
​ if (/停车场名称/.test(item[key].label)) {  
  
​ parkName = item[key].value;  
  
​ }  
  
​ if (/备案状态/.test(item[key].label)) {  
  
​ filingStatus = item[key].value || '未备案';  
  
​ }  
  
​ if (/所在行政区/.test(item[key].label)) {  
  
​ region = item[key].value;  
  
​ }  
  
​ if (/停车场类型/.test(item[key].label)) {  
  
​ parkType = item[key].value;  
  
​ }  
  
​ if (/相关备案/.test(item[key].label)) {  
  
​ recordIds = item[key].value?.ids || [];  
  
​ }  
  
​ if (/泊位总数/.test(item[key].label)) {  
  
​ spaces = item[key].value;  
  
​ }  
  
​ if (/对外开放泊位个数/.test(item[key].label)) {  
  
​ outSpaces = item[key].value;  
  
​ }  
  
​ if (/是否联网接入宜昌智慧停车平台/.test(item[key].label)) {  
  
​ connectStatus = item[key].value;  
  
​ }  
  
​ if (/充电泊位数/.test(item[key].label)) {  
  
​ chargeSpaces = item[key].value;  
  
​ }  
  
​ }  
  
​ }  
  
​ // 图标  
  
​ const temp = [];  
  
​ for (const key in item) {  
  
​ if (item[key]?.label) {  
  
​ temp.push({ ...item[key] });  
  
​ }  
  
​ }  
  
​ const des = getInfo(temp);  
  
​ // 坐标转换  
  
​ const coords = gcj02towgs84(longitude, latitude);  
  
​ const entity = new window.Cesium.Entity({  
  
​ position: window.Cesium.Cartesian3.fromDegrees(coords[0], coords[1], 10),  
  
​ billboard: {  
  
​ image: getFilingImg(filingStatus, connectStatus),  
  
​ width: 27,  
  
​ height: 60,  
  
​ disableDepthTestDistance: Number.POSITIVE\_INFINITY,  
  
​ },  
  
​ id: item?.id?.value,  
  
​ name: parkName,  
  
​ description: des,  
  
​ attributes: {  
  
​ ...item,  
  
​ latitude,  
  
​ longitude,  
  
​ parkName,  
  
​ filingStatus,  
  
​ region,  
  
​ parkType,  
  
​ recordIds,  
  
​ spaces,  
  
​ outSpaces,  
  
​ connectStatus,  
  
​ chargeSpaces,  
  
​ },  
  
​ });  
  
​ viewer.entities.add(entity);  
  
   
  
​ const res = {  
  
​ ...item,  
  
​ latitude,  
  
​ longitude,  
  
​ parkName,  
  
​ filingStatus,  
  
​ region,  
  
​ parkType,  
  
​ recordIds,  
  
​ spaces,  
  
​ outSpaces,  
  
​ connectStatus,  
  
​ chargeSpaces,  
  
​ };  
  
   
  
​ const arrayItem = {};  
  
​ for (const key in res) {  
  
​ if (Object.hasOwnProperty.call(res, key)) {  
  
​ if (!/relatedrowid/.test(key)) {  
  
​ arrayItem[key] = res[key];  
  
​ }  
  
​ }  
  
​ }  
  
​ return arrayItem;  
  
​ } catch (e) {  
  
​ console.log(e);  
  
​ return {};  
  
​ }  
  
​ });  
  
​ return array;  
  
 };  
  
   
  
 useEffect(() => {  
  
​ if (mapViewer) {  
  
​ // 交互  
  
​ const handler = new window.Cesium.ScreenSpaceEventHandler(mapViewer.scene.canvas);  
  
​ handler.setInputAction(e => {  
  
​ let pick = mapViewer.scene.pick(e.position);  
  
​ if (pick && pick.id) {  
  
​ console.log(pick);  
  
​ const fetchData = async () => {  
  
​ const res = await Request(api.tableview.getviewdatas, {  
  
​ applicationID: 21,  
  
​ viewUID: 'feb5872227874e37b6518e398c723be8',  
  
​ offset: 0,  
  
​ rows: 1,  
  
​ queryFields: JSON.stringify([  
  
​ {  
  
​ id: 'ea98f19a3ce843308d7de176a09d4bec',  
  
​ value: pick.id.\_name,  
  
​ type: 'like',  
  
​ },  
  
​ ]),  
  
​ });  
  
​ if (res.success && res.data?.list?.length === 1) {  
  
​ const obj = pick.id.\_attributes || {};  
  
​ const urlArr = getPictureUrl(obj, res.data.list[0], '1c520d440fdd40f291e919a049f95761');  
  
​ obj.exploreImg = {  
  
​ label: '现场勘验视频图片',  
  
​ value: urlArr,  
  
​ name: pick.id.\_name,  
  
​ };  
  
​ const temp = [];  
  
​ for (const key in obj) {  
  
​ if (obj[key]?.label) {  
  
​ temp.push({ ...obj[key] });  
  
​ }  
  
​ }  
  
​ const des = getInfo(temp);  
  
​ pick.id.description = des;  
  
​ }  
  
​ };  
  
​ fetchData();  
  
​ }  
  
​ }, window.Cesium.ScreenSpaceEventType.LEFT\_CLICK);  
  
   
  
​ const fetchData = async () => {  
  
​ let rows = [];  
  
​ const res1 = await Request(api.form.formdetail, {  
  
​ applicationID: 21,  
  
​ formUID: '5fcb8d72f5994488973a45fa0c08b333',  
  
​ });  
  
​ if (res1.success) {  
  
​ rows = JSON.parse(res1.data?.form?.formJson || '{}')?.rows || [];  
  
​ rows = rows.map(row => row.children).flat(Infinity);  
  
​ let count = 0,  
  
​ offset = 0,  
  
​ parks = [];  
  
​ const getDatas = async start => {  
  
​ const res = await Request(api.tableview.getviewdatas, {  
  
​ applicationID: 21,  
  
​ viewUID: 'd56f8e4c1d2b477591b62e9d44e7d515',  
  
​ offset: start,  
  
​ rows: 200,  
  
​ });  
  
​ if (res.success) {  
  
​ const length = res.data.list.length;  
  
​ const marker = getMarker(res.data.list, rows, mapViewer);  
  
​ count = start + length;  
  
​ parks = parks.concat(marker);  
  
​ if (length > 0) {  
  
​ await getDatas(start + length);  
  
​ }  
  
​ }  
  
​ };  
  
​ await getDatas(offset);  
  
​ console.log(parks);  
  
​ setAllList(parks);  
  
​ }  
  
​ };  
  
​ fetchData();  
  
​ }  
  
​ // eslint-disable-next-line react-hooks/exhaustive-deps  
  
 }, [mapViewer]);  
  
   
  
 const transformLabel = label => {  
  
​ switch (label) {  
  
​ case '是':  
  
​ return '已联网';  
  
​ case '否':  
  
​ return '未联网';  
  
​ default:  
  
​ break;  
  
​ }  
  
 };  
  
   
  
 useEffect(() => {  
  
​ let temp = allList || [];  
  
​ if (regionLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.region && Array.isArray(item.region)) {  
  
​ return item.region[item.region.length - 1] === regionLabel;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ if (filingLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.filingStatus) {  
  
​ return item.filingStatus === filingLabel;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ if (typeLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.parkType) {  
  
​ return item.parkType.text.indexOf(typeLabel) > -1;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ if (connectLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.connectStatus) {  
  
​ return transformLabel(item.connectStatus) === connectLabel;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ setList(temp);  
  
 }, [allList, regionLabel, filingLabel, typeLabel, connectLabel]);  
  
   
  
 useEffect(() => {  
  
​ if (mapViewer) {  
  
​ mapViewer.entities.removeAll();  
  
​ }  
  
​ if (window.Cesium) {  
  
​ list.forEach(item => {  
  
​ const { longitude, latitude, filingStatus, parkName, connectStatus } = item;  
  
​ const temp = [];  
  
​ for (const key in item) {  
  
​ if (item[key]?.label) {  
  
​ temp.push({ ...item[key] });  
  
​ }  
  
​ }  
  
​ const des = getInfo(temp);  
  
​ // 坐标转换  
  
​ const coords = gcj02towgs84(longitude, latitude);  
  
​ const entity = new window.Cesium.Entity({  
  
​ position: window.Cesium.Cartesian3.fromDegrees(coords[0], coords[1], 10),  
  
​ billboard: {  
  
​ image: getFilingImg(filingStatus, connectStatus),  
  
​ width: 27,  
  
​ height: 60,  
  
​ disableDepthTestDistance: Number.POSITIVE\_INFINITY,  
  
​ },  
  
​ name: parkName,  
  
​ description: des,  
  
​ id: item?.id?.value,  
  
​ attributes: {  
  
​ ...item,  
  
​ },  
  
​ });  
  
​ mapViewer.entities.add(entity);  
  
​ });  
  
   
  
​ if (selected) {  
  
​ const coords = gcj02towgs84(list[0]?.longitude, list[0]?.latitude);  
  
​ const initPosition = window.Cesium.Cartesian3.fromDegrees(coords[0], coords[1], 3000);  
  
​ mapScene.camera.setView({  
  
​ destination: initPosition, //位置  
  
​ // orientation: initialOrientation,//方向  
  
​ endTransform: window.Cesium.Matrix4.IDENTITY,  
  
​ });  
  
​ setSelected(0);  
  
​ }  
  
​ }  
  
   
  
​ // eslint-disable-next-line react-hooks/exhaustive-deps  
  
 }, [list, window.Cesium]);  
  
   
  
 return (  
  
 <div className="yichang-map no-scale">  
  
 <div className="search-panel">  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={regionData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={regionTreeData}  
  
​ placeholder="查找区县市"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onRegionChange}  
  
​ />  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={filingData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={filingTreeData}  
  
​ placeholder="查找备案状态"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onfilingChange}  
  
​ />  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={typeData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={typeTreeData}  
  
​ placeholder="查找停车场类型"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onTypeChange}  
  
​ />  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={connectData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={connectTreeData}  
  
​ placeholder="联网状态"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onConnectChange}  
  
​ />  
  
 <div className="search-input">  
  
​ <SearchInput data={allList} onSelect={onSelect} onClear={onClear} />  
  
​ </div>  
  
​ </div>  
  
 <div id="cesiumContainer" ref={containerRef}></div>  
  
​ </div>  
  
 );  
  
};  
  
   
  
export default YichangMap;  
  
   
  
import { useState, useCallback, useEffect } from 'react';  
  
import { Selector } from '@/components';  
  
import { Button, Modal, Form, Select, InputNumber, message } from 'antd';  
  
import api from '@/parkaccess-manager/model';  
  
import { request } from '@/utils';  
  
   
  
const PlanModal = props => {  
  
 const [form] = Form.useForm();  
  
 const [districtOptions, setDistrictOptions] = useState([]);  
  
   
  
 const fetchDistrict = useCallback(async () => {  
  
​ const result = await request(api.park.getyichangdistrictdic);  
  
​ if (result.success) {  
  
​ setDistrictOptions(result.data?.data || []);  
  
​ } else {  
  
​ message.error(result.message || '获取城区数据失败');  
  
​ }  
  
 }, []);  
  
 useEffect(() => {  
  
​ fetchDistrict();  
  
 }, [fetchDistrict]);  
  
   
  
 const handleSubmit = () => {  
  
​ form.validateFields().then(values => {  
  
​ props.handleSubmit(values);  
  
​ props.handleCancel();  
  
​ });  
  
 };  
  
   
  
 return (  
  
​ <Modal title="制定抽检计划" visible={props.visible} destroyOnClose onCancel={() => props.handleCancel()} onOk={handleSubmit}>  
  
​ <Form form={form} initialValues={{ cityName: '宜昌市' }}>  
  
​ <Form.Item name="cityName" label="所在城市" rules={[{ required: true }]}>  
  
​ <Select>  
  
​ <Select.Option value="宜昌市">宜昌市</Select.Option>  
  
​ </Select>  
  
​ </Form.Item>  
  
​ <Form.Item name="districtName" label="所在区县" rules={[{ required: true }]}>  
  
​ <Select>  
  
​ {districtOptions.map((item, index) => (  
  
​ <Select.Option key={index} value={item}>  
  
​ {item}  
  
​ </Select.Option>  
  
​ ))}  
  
​ </Select>  
  
​ </Form.Item>  
  
​ <Form.Item  
  
​ name="limitCount"  
  
​ label="抽检数量"  
  
​ rules={[  
  
​ { required: true },  
  
​ {  
  
​ validator: (rule, value) => {  
  
​ if (!value && value !== 0) {  
  
​ return Promise.resolve();  
  
​ } else if (value > 100) {  
  
​ return Promise.reject('抽检数量不能大于100');  
  
​ } else if (value < 1) {  
  
​ return Promise.reject('抽检数量不能小于1');  
  
​ } else {  
  
​ return Promise.resolve();  
  
​ }  
  
​ },  
  
​ },  
  
​ ]}  
  
​ \>  
  
​ <InputNumber precision={0} style={{ width: '100%' }} />  
  
​ </Form.Item>  
  
​ </Form>  
  
​ </Modal>  
  
 );  
  
};  
  
   
  
export default props => {  
  
 const [param, setParam] = useState({});  
  
 const [visible, setVisible] = useState(false);  
  
   
  
 const columns = [  
  
​ {  
  
​ title: '车场编号',  
  
​ dataIndex: 'parkCode',  
  
​ },  
  
​ {  
  
​ title: '车场名称',  
  
​ dataIndex: 'parkName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ width: 220,  
  
​ },  
  
​ {  
  
​ title: '所在城市',  
  
​ dataIndex: 'cityName',  
  
​ },  
  
​ {  
  
​ title: '所在区县',  
  
​ dataIndex: 'districtName',  
  
​ },  
  
​ {  
  
​ title: '总泊位数',  
  
​ dataIndex: 'totalCount',  
  
​ },  
  
​ {  
  
​ title: '剩余泊位数',  
  
​ dataIndex: 'intrinsictotal',  
  
​ },  
  
​ {  
  
​ title: '经营单位',  
  
​ dataIndex: 'userGroupName',  
  
​ width: 200,  
  
​ },  
  
​ {  
  
​ title: '操作',  
  
​ key: 'action',  
  
​ render: (value, record) => {  
  
​ return (  
  
​ <Button  
  
​ type="link"  
  
​ onClick={() =>  
  
​ props.openTab(  
  
​ {  
  
​ rightID: `access\_info\_${record.parkCode}`,  
  
​ displayName: `接入情况详情-${record.parkName}`,  
  
​ path: 'ParkAccessManagerYiChangInfo',  
  
​ },  
  
​ { record: record }  
  
​ )  
  
​ }  
  
​ \>  
  
​ 接入情况详情  
  
​ </Button>  
  
​ );  
  
​ },  
  
​ },  
  
 ];  
  
   
  
 return (  
  
​ <>  
  
​ <Selector  
  
​ extraTagAfter={  
  
​ <Button  
  
​ type="primary"  
  
​ onClick={() => {  
  
​ setVisible(true);  
  
​ }}  
  
​ \>  
  
​ 制定抽检计划  
  
​ </Button>  
  
​ }  
  
​ param={param}  
  
​ columns={columns}  
  
​ api={api.park.getyichangparklist}  
  
​ enabledExport={false}  
  
​ refresh={false}  
  
​ />  
  
​ {visible && <PlanModal visible={visible} handleCancel={() => setVisible(false)} handleSubmit={values => setParam(values)} />}  
  
​ </>  
  
 );  
  
};  
  
   
  
/\* eslint-disable react-hooks/exhaustive-deps \*/  
  
/\* eslint-disable no-unused-vars \*/  
  
import React, { useCallback, useEffect, useRef, useState } from 'react';  
  
import './map.less';  
  
import api from '@/desktop/model';  
  
import { Request } from '@/utils';  
  
import greenImg from './img/p-cyan.png';  
  
import orangeImg from './img/p-orange.png';  
  
import grayImg from './img/p-gray.png';  
  
import blueImg from './img/p-blue.png';  
  
import { TreeSelect } from 'antd';  
  
import SearchInput from './searchInput';  
  
import { gcj02towgs84 } from '@/utils/gis';  
  
   
  
const YichangMap = () => {  
  
 const [allList, setAllList] = useState([]);  
  
 const [list, setList] = useState([]);  
  
 const [mapScene, setMapScene] = useState(null);  
  
 const [mapViewer, setMapViewer] = useState(null);  
  
 const [regionData, setRegionData] = useState();  
  
 const [regionLabel, setRegionLabel] = useState(null);  
  
 const [filingData, setFilingData] = useState();  
  
 const [filingLabel, setFilingLabel] = useState(null);  
  
 const [typeData, setTypeData] = useState();  
  
 const [typeLabel, setTypeLabel] = useState(null);  
  
 const [connectData, setConnectData] = useState();  
  
 const [connectLabel, setConnectLabel] = useState(null);  
  
   
  
 const [selected, setSelected] = useState(0);  
  
   
  
 const regionTreeData = [  
  
​ {  
  
​ title: '西陵区',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '伍家岗区',  
  
​ value: '0-1',  
  
​ },  
  
​ {  
  
​ title: '点军区',  
  
​ value: '0-2',  
  
​ },  
  
​ {  
  
​ title: '猇亭区',  
  
​ value: '0-3',  
  
​ },  
  
​ {  
  
​ title: '夷陵区',  
  
​ value: '0-4',  
  
​ },  
  
​ {  
  
​ title: '远安县',  
  
​ value: '0-5',  
  
​ },  
  
​ {  
  
​ title: '兴山县',  
  
​ value: '0-6',  
  
​ },  
  
​ {  
  
​ title: '秭归县',  
  
​ value: '0-7',  
  
​ },  
  
​ {  
  
​ title: '长阳土家族自治县',  
  
​ value: '0-8',  
  
​ },  
  
​ {  
  
​ title: '五峰土家族自治县',  
  
​ value: '0-9',  
  
​ },  
  
​ {  
  
​ title: '枝江市',  
  
​ value: '0-10',  
  
​ },  
  
​ {  
  
​ title: '宜都市',  
  
​ value: '0-11',  
  
​ },  
  
​ {  
  
​ title: '当阳市',  
  
​ value: '0-12',  
  
​ },  
  
 ];  
  
 const filingTreeData = [  
  
​ {  
  
​ title: '未备案',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '备案审核中',  
  
​ value: '0-1',  
  
​ },  
  
​ {  
  
​ title: '备案通过',  
  
​ value: '0-2',  
  
​ },  
  
​ {  
  
​ title: '备案未通过',  
  
​ value: '0-3',  
  
​ },  
  
 ];  
  
 const typeTreeData = [  
  
​ {  
  
​ title: '公共停车场',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '配建停车场',  
  
​ value: '0-1',  
  
​ },  
  
​ {  
  
​ title: '路内停车场',  
  
​ value: '0-2',  
  
​ },  
  
 ];  
  
 const connectTreeData = [  
  
​ {  
  
​ title: '已联网',  
  
​ value: '0-0',  
  
​ },  
  
​ {  
  
​ title: '未联网',  
  
​ value: '0-1',  
  
​ },  
  
 ];  
  
 const containerRef = useRef();  
  
 // 创建一个View  
  
 const restUrl = 'http://10.55.49.27:8884/gtis/map/rest/69dbfedc855347eda29a2eb2011d4c48/rest/maps/dianziditu\_zhuji';  
  
   
  
 const onRegionChange = useCallback((value, label) => {  
  
​ setRegionData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setRegionLabel(null);  
  
​ } else {  
  
​ setRegionLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onfilingChange = useCallback((value, label) => {  
  
​ setFilingData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setFilingLabel(null);  
  
​ } else {  
  
​ setFilingLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onTypeChange = useCallback((value, label) => {  
  
​ setTypeData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setTypeLabel(null);  
  
​ } else {  
  
​ setTypeLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onConnectChange = useCallback((value, label) => {  
  
​ setConnectData(value);  
  
​ if (typeof value === 'undefined') {  
  
​ setConnectLabel(null);  
  
​ } else {  
  
​ setConnectLabel(label[0]);  
  
​ }  
  
 }, []);  
  
   
  
 const onSelect = (value, option) => {  
  
​ setList([option.park]);  
  
​ setSelected(1);  
  
 };  
  
   
  
 const onClear = () => {  
  
​ setList(allList);  
  
​ setSelected(0);  
  
 };  
  
   
  
 const getPictureUrl = (item, item2, name) => {  
  
​ let result = item?.recordIds?.map(() => {  
  
​ return item2?.[name];  
  
​ });  
  
​ result = result  
  
​ ?.map(item => {  
  
​ return Array.isArray(item)  
  
​ ? item.map(pic => {  
  
​ if (pic?.type && pic.type.indexOf('image') > -1) {  
  
​ return pic?.url;  
  
​ } else {  
  
​ return '';  
  
​ }  
  
​ })  
  
​ : '';  
  
​ })  
  
​ .flat(Infinity);  
  
​ return result;  
  
 };  
  
   
  
 useEffect(() => {  
  
​ const initMap = () => {  
  
​ let link, script;  
  
​ link = document.createElement('link');  
  
​ link.type = 'text/css';  
  
​ link.rel = 'stylesheet';  
  
​ link.href = `${window.\_pageURL}/static/map/Widgets/widgets.css`;  
  
​ document.head.appendChild(link);  
  
​ script = document.createElement('script');  
  
​ script.type = 'text/javascript';  
  
​ script.src = `${window.\_pageURL}/static/map/Cesium.js`;  
  
​ document.head.appendChild(script);  
  
​ script.onload = () => {  
  
​ const div = document.querySelector('.cesium-viewer');  
  
​ if (div) {  
  
​ return;  
  
​ }  
  
​ const viewer = new window.Cesium.Viewer('cesiumContainer', {  
  
​ sceneMode: window.Cesium.SceneMode.SCENE3D, //设置为三维地图  
  
​ });  
  
​ setMapViewer(viewer);  
  
​ const scene = viewer.scene;  
  
​ setMapScene(scene);  
  
​ // 加载图层  
  
​ // eslint-disable-next-line no-unused-vars  
  
​ const layer = viewer.imageryLayers.addImageryProvider(  
  
​ new window.Cesium.SuperMapImageryProvider({  
  
​ url: restUrl,  
  
​ })  
  
​ );  
  
​ const initPosition = window.Cesium.Cartesian3.fromDegrees(111.293779, 30.650459, 4000);  
  
​ scene.camera.setView({  
  
​ destination: initPosition, //位置  
  
​ orientation: {  
  
​ pitch: -0.6,  
  
​ },  
  
​ endTransform: window.Cesium.Matrix4.IDENTITY,  
  
​ });  
  
​ };  
  
​ };  
  
​ initMap();  
  
​ return () => {  
  
​ if (mapViewer) {  
  
​ mapViewer.destroy();  
  
​ }  
  
​ };  
  
​ // eslint-disable-next-line react-hooks/exhaustive-deps  
  
 }, []);  
  
   
  
 const getInfo = items => {  
  
​ let info = '<div class="content-wrapper"><div class="item-container">';  
  
​ items.forEach(item => {  
  
​ const { label, value } = item;  
  
​ if (label === '详细地址' || label === '车场业态' || label === '备案状态' || label === '停车场类型' || label === '充电泊位数') {  
  
​ if (value) {  
  
​ info = `${info}<div class="item"><span class="item-key">${label}: </span><span class="item-value">${  
  
​ value?.text  
  
​ ? Array.isArray(value?.text)  
  
​ ? value?.text?.join(',')  
  
​ : value?.text?.address || value?.text?.text  
  
​ ? value?.text?.address || value?.text?.text  
  
​ : value?.text  
  
​ : Array.isArray(value)  
  
​ ? value?.[0]?.url || value.join('/')  
  
​ : value || ''  
  
​ }</span></div>`;  
  
​ }  
  
​ }  
  
​ if (label === '是否联网接入宜昌智慧停车平台') {  
  
​ info = `${info}<div class="item"><span class="item-key">联网状态: </span><span class="item-value">${  
  
​ value === '是' ? '已联网' : '未联网'  
  
​ }</span></div>`;  
  
​ }  
  
​ });  
  
​ info = `${info}</div>`;  
  
​ info = `${info}<div class="num-container">`;  
  
​ items.forEach(item => {  
  
​ const { label, value } = item;  
  
​ if (label === '泊位总数' || label === '对外开放泊位个数') {  
  
​ info = `${info}<div class="num-item"><div class="item-value">${value}</div><div class="item-label">${label}</div></div>`;  
  
​ }  
  
​ });  
  
​ info = `${info}</div>`;  
  
​ items.forEach(item => {  
  
​ const { label, value } = item;  
  
​ if (label === '现场勘验视频图片' && value?.[0]) {  
  
​ info = `${info}  
  
 <div class="img-container">  
  
​ <img src="${window.\_baseURL}${value[0]}" alt="img" imgUrl="${window.\_baseURL}${value[0]}"/>  
  
​ </div>  
  
​ `;  
  
​ }  
  
​ });  
  
​ info = `${info}</div>`;  
  
​ return info;  
  
 };  
  
   
  
 const getFilingImg = (status, connectStatus) => {  
  
​ if (status === '备案通过' && connectStatus === '是') {  
  
​ return greenImg;  
  
​ } else if (status === '备案通过' && connectStatus === '否') {  
  
​ return orangeImg;  
  
​ } else if ((status === '备案未通过' || status === '备案审核中' || status === '未备案') && connectStatus === '是') {  
  
​ return blueImg;  
  
​ } else {  
  
​ return grayImg;  
  
​ }  
  
 };  
  
   
  
 const getMarker = (list, rows, viewer) => {  
  
​ const array = list.map(item => {  
  
​ try {  
  
​ const obj = { ...item };  
  
​ let latitude = 0,  
  
​ longitude = 0,  
  
​ parkName = '',  
  
​ filingStatus = '',  
  
​ parkType = '',  
  
​ region = [],  
  
​ recordIds = [],  
  
​ spaces = 0,  
  
​ outSpaces = 0,  
  
​ connectStatus = '',  
  
​ chargeSpaces = 0;  
  
​ for (const key in item) {  
  
​ if (Object.hasOwnProperty.call(item, key)) {  
  
​ if (/latitude/.test(key) || obj[key]?.text?.latitude) {  
  
​ latitude = obj[key]?.text?.latitude || obj[key];  
  
​ } else if (/longitude/.test(key) || obj[key]?.text?.longitude) {  
  
​ longitude = obj[key]?.text?.longitude || obj[key];  
  
​ }  
  
​ item[key] = {  
  
​ name: key,  
  
​ value: item[key],  
  
​ label: rows.filter(row => row.name === key)?.[0]?.label,  
  
​ };  
  
​ if (/停车场名称/.test(item[key].label)) {  
  
​ parkName = item[key].value;  
  
​ }  
  
​ if (/备案状态/.test(item[key].label)) {  
  
​ filingStatus = item[key].value || '未备案';  
  
​ }  
  
​ if (/所在行政区/.test(item[key].label)) {  
  
​ region = item[key].value;  
  
​ }  
  
​ if (/停车场类型/.test(item[key].label)) {  
  
​ parkType = item[key].value;  
  
​ }  
  
​ if (/相关备案/.test(item[key].label)) {  
  
​ recordIds = item[key].value?.ids || [];  
  
​ }  
  
​ if (/泊位总数/.test(item[key].label)) {  
  
​ spaces = item[key].value;  
  
​ }  
  
​ if (/对外开放泊位个数/.test(item[key].label)) {  
  
​ outSpaces = item[key].value;  
  
​ }  
  
​ if (/是否联网接入宜昌智慧停车平台/.test(item[key].label)) {  
  
​ connectStatus = item[key].value;  
  
​ }  
  
​ if (/充电泊位数/.test(item[key].label)) {  
  
​ chargeSpaces = item[key].value;  
  
​ }  
  
​ }  
  
​ }  
  
​ // 图标  
  
​ const temp = [];  
  
​ for (const key in item) {  
  
​ if (item[key]?.label) {  
  
​ temp.push({ ...item[key] });  
  
​ }  
  
​ }  
  
​ const des = getInfo(temp);  
  
​ // 坐标转换  
  
​ const coords = gcj02towgs84(longitude, latitude);  
  
​ const entity = new window.Cesium.Entity({  
  
​ position: window.Cesium.Cartesian3.fromDegrees(coords[0], coords[1], 10),  
  
​ billboard: {  
  
​ image: getFilingImg(filingStatus, connectStatus),  
  
​ width: 27,  
  
​ height: 60,  
  
​ disableDepthTestDistance: Number.POSITIVE\_INFINITY,  
  
​ },  
  
​ id: item?.id?.value,  
  
​ name: parkName,  
  
​ description: des,  
  
​ attributes: {  
  
​ ...item,  
  
​ latitude,  
  
​ longitude,  
  
​ parkName,  
  
​ filingStatus,  
  
​ region,  
  
​ parkType,  
  
​ recordIds,  
  
​ spaces,  
  
​ outSpaces,  
  
​ connectStatus,  
  
​ chargeSpaces,  
  
​ },  
  
​ });  
  
​ viewer.entities.add(entity);  
  
   
  
​ const res = {  
  
​ ...item,  
  
​ latitude,  
  
​ longitude,  
  
​ parkName,  
  
​ filingStatus,  
  
​ region,  
  
​ parkType,  
  
​ recordIds,  
  
​ spaces,  
  
​ outSpaces,  
  
​ connectStatus,  
  
​ chargeSpaces,  
  
​ };  
  
   
  
​ const arrayItem = {};  
  
​ for (const key in res) {  
  
​ if (Object.hasOwnProperty.call(res, key)) {  
  
​ if (!/relatedrowid/.test(key)) {  
  
​ arrayItem[key] = res[key];  
  
​ }  
  
​ }  
  
​ }  
  
​ return arrayItem;  
  
​ } catch (e) {  
  
​ console.log(e);  
  
​ return {};  
  
​ }  
  
​ });  
  
​ return array;  
  
 };  
  
   
  
 useEffect(() => {  
  
​ if (mapViewer) {  
  
​ // 交互  
  
​ const handler = new window.Cesium.ScreenSpaceEventHandler(mapViewer.scene.canvas);  
  
​ handler.setInputAction(e => {  
  
​ let pick = mapViewer.scene.pick(e.position);  
  
​ if (pick && pick.id) {  
  
​ console.log(pick);  
  
​ const fetchData = async () => {  
  
​ const res = await Request(api.tableview.getviewdatas, {  
  
​ applicationID: 21,  
  
​ viewUID: 'feb5872227874e37b6518e398c723be8',  
  
​ offset: 0,  
  
​ rows: 1,  
  
​ queryFields: JSON.stringify([  
  
​ {  
  
​ id: 'ea98f19a3ce843308d7de176a09d4bec',  
  
​ value: pick.id.\_name,  
  
​ type: 'like',  
  
​ },  
  
​ ]),  
  
​ });  
  
​ if (res.success && res.data?.list?.length === 1) {  
  
​ const obj = pick.id.\_attributes || {};  
  
​ const urlArr = getPictureUrl(obj, res.data.list[0], '1c520d440fdd40f291e919a049f95761');  
  
​ obj.exploreImg = {  
  
​ label: '现场勘验视频图片',  
  
​ value: urlArr,  
  
​ name: pick.id.\_name,  
  
​ };  
  
​ const temp = [];  
  
​ for (const key in obj) {  
  
​ if (obj[key]?.label) {  
  
​ temp.push({ ...obj[key] });  
  
​ }  
  
​ }  
  
​ const des = getInfo(temp);  
  
​ pick.id.description = des;  
  
​ }  
  
​ };  
  
​ fetchData();  
  
​ }  
  
​ }, window.Cesium.ScreenSpaceEventType.LEFT\_CLICK);  
  
   
  
​ const fetchData = async () => {  
  
​ let rows = [];  
  
​ const res1 = await Request(api.form.formdetail, {  
  
​ applicationID: 21,  
  
​ formUID: '5fcb8d72f5994488973a45fa0c08b333',  
  
​ });  
  
​ if (res1.success) {  
  
​ rows = JSON.parse(res1.data?.form?.formJson || '{}')?.rows || [];  
  
​ rows = rows.map(row => row.children).flat(Infinity);  
  
​ let count = 0,  
  
​ offset = 0,  
  
​ parks = [];  
  
​ const getDatas = async start => {  
  
​ const res = await Request(api.tableview.getviewdatas, {  
  
​ applicationID: 21,  
  
​ viewUID: 'd56f8e4c1d2b477591b62e9d44e7d515',  
  
​ offset: start,  
  
​ rows: 200,  
  
​ });  
  
​ if (res.success) {  
  
​ const length = res.data.list.length;  
  
​ const marker = getMarker(res.data.list, rows, mapViewer);  
  
​ count = start + length;  
  
​ parks = parks.concat(marker);  
  
​ if (length > 0) {  
  
​ await getDatas(start + length);  
  
​ }  
  
​ }  
  
​ };  
  
​ await getDatas(offset);  
  
​ console.log(parks);  
  
​ setAllList(parks);  
  
​ }  
  
​ };  
  
​ fetchData();  
  
​ }  
  
​ // eslint-disable-next-line react-hooks/exhaustive-deps  
  
 }, [mapViewer]);  
  
   
  
 const transformLabel = label => {  
  
​ switch (label) {  
  
​ case '是':  
  
​ return '已联网';  
  
​ case '否':  
  
​ return '未联网';  
  
​ default:  
  
​ break;  
  
​ }  
  
 };  
  
   
  
 useEffect(() => {  
  
​ let temp = allList || [];  
  
​ if (regionLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.region && Array.isArray(item.region)) {  
  
​ return item.region[item.region.length - 1] === regionLabel;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ if (filingLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.filingStatus) {  
  
​ return item.filingStatus === filingLabel;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ if (typeLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.parkType) {  
  
​ return item.parkType.text.indexOf(typeLabel) > -1;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ if (connectLabel) {  
  
​ temp = temp.filter(item => {  
  
​ if (item.connectStatus) {  
  
​ return transformLabel(item.connectStatus) === connectLabel;  
  
​ } else {  
  
​ return false;  
  
​ }  
  
​ });  
  
​ }  
  
​ setList(temp);  
  
 }, [allList, regionLabel, filingLabel, typeLabel, connectLabel]);  
  
   
  
 useEffect(() => {  
  
​ if (mapViewer) {  
  
​ mapViewer.entities.removeAll();  
  
​ }  
  
​ if (window.Cesium) {  
  
​ list.forEach(item => {  
  
​ const { longitude, latitude, filingStatus, parkName, connectStatus } = item;  
  
​ const temp = [];  
  
​ for (const key in item) {  
  
​ if (item[key]?.label) {  
  
​ temp.push({ ...item[key] });  
  
​ }  
  
​ }  
  
​ const des = getInfo(temp);  
  
​ // 坐标转换  
  
​ const coords = gcj02towgs84(longitude, latitude);  
  
​ const entity = new window.Cesium.Entity({  
  
​ position: window.Cesium.Cartesian3.fromDegrees(coords[0], coords[1], 10),  
  
​ billboard: {  
  
​ image: getFilingImg(filingStatus, connectStatus),  
  
​ width: 27,  
  
​ height: 60,  
  
​ disableDepthTestDistance: Number.POSITIVE\_INFINITY,  
  
​ },  
  
​ name: parkName,  
  
​ description: des,  
  
​ id: item?.id?.value,  
  
​ attributes: {  
  
​ ...item,  
  
​ },  
  
​ });  
  
​ mapViewer.entities.add(entity);  
  
​ });  
  
   
  
​ if (selected) {  
  
​ const coords = gcj02towgs84(list[0]?.longitude, list[0]?.latitude);  
  
​ const initPosition = window.Cesium.Cartesian3.fromDegrees(coords[0], coords[1], 3000);  
  
​ mapScene.camera.setView({  
  
​ destination: initPosition, //位置  
  
​ // orientation: initialOrientation,//方向  
  
​ endTransform: window.Cesium.Matrix4.IDENTITY,  
  
​ });  
  
​ setSelected(0);  
  
​ }  
  
​ }  
  
   
  
​ // eslint-disable-next-line react-hooks/exhaustive-deps  
  
 }, [list, window.Cesium]);  
  
   
  
 return (  
  
 <div className="yichang-map no-scale">  
  
 <div className="search-panel">  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={regionData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={regionTreeData}  
  
​ placeholder="查找区县市"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onRegionChange}  
  
​ />  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={filingData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={filingTreeData}  
  
​ placeholder="查找备案状态"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onfilingChange}  
  
​ />  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={typeData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={typeTreeData}  
  
​ placeholder="查找停车场类型"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onTypeChange}  
  
​ />  
  
​ <TreeSelect  
  
​ style={{ width: '200px' }}  
  
​ value={connectData}  
  
​ dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}  
  
​ treeData={connectTreeData}  
  
​ placeholder="联网状态"  
  
​ treeDefaultExpandAll  
  
​ allowClear  
  
​ onChange={onConnectChange}  
  
​ />  
  
 <div className="search-input">  
  
​ <SearchInput data={allList} onSelect={onSelect} onClear={onClear} />  
  
​ </div>  
  
​ </div>  
  
 <div id="cesiumContainer" ref={containerRef}></div>  
  
​ </div>  
  
 );  
  
};  
  
   
  
export default YichangMap;  
  
   
  
/\* eslint-disable no-unused-vars \*/  
  
import React, { useEffect, useRef, useState } from 'react';  
  
import moment from 'moment';  
  
import QRCode from 'qrcode.react';  
  
import { Request, Utils } from '@/utils';  
  
import api from '@/desktop/model';  
  
import { getDate, getPrintItems, getRelatedItems, getValue, isPC } from '../utils';  
  
import { Image, message } from 'antd';  
  
import { Storage } from '@/utils';  
  
import SubItem from './basic/subItem';  
  
import SubList from './basic/subList';  
  
   
  
const PrintPage = props => {  
  
 const { groupName, mode, list, sysList, subs, setting, extra, updateSubLayout, qrUrl, openNewQrCode } = props;  
  
   
  
 const [downLink, setDownLink] = useState('');  
  
 const [base64Url, setBase64Url] = useState('');  
  
   
  
 useEffect(() => {  
  
​ const search = qrUrl?.split?.('?')?.[1] || '';  
  
​ const base = qrUrl?.split?.('?')?.[0] || '';  
  
​ setBase64Url(`${base}?${window.btoa(window.encodeURI(search) + '\_print-qrcode')}`);  
  
 }, [qrUrl]);  
  
   
  
 useEffect(() => {  
  
​ const timer = setInterval(() => {  
  
​ const canvasImg = document.getElementById(qrUrl);  
  
​ const link = canvasImg?.toDataURL('image/png');  
  
​ if (link) {  
  
​ setDownLink(link);  
  
​ clearTimeout(timer);  
  
​ }  
  
​ }, 200);  
  
 }, [qrUrl, base64Url]);  
  
   
  
 return (  
  
 <div className="print-body">  
  
 <div className="print-body-header">  
  
​ {/\* 打印头部，包含企业logo，企业名称，表单名称 \*/}  
  
 <div className="print-title">  
  
​ {/\* <div className="group-logo">{groupLogo}</div> \*/}  
  
​ {extra?.groupName?.checked && <div className="group-title">{groupName || Storage.get('user')?.userGroupName || ''}</div>}  
  
​ {extra?.formName?.checked && <div className="form-title">{extra.formName.data}</div>}  
  
​ {extra?.qrcode?.checked &&  
  
​ (openNewQrCode && !isPC() ? (  
  
​ <div  
  
​ className="form-qrcode mobile-qrcode"  
  
​ onClick={() => window.open(`${window.\_pageURL}/print-sys-mobile-qrcode?base64Url=${base64Url}`, '\_self')}  
  
​ \>  
  
​ <QRCode id={qrUrl} value={base64Url} size={mode === 'mobile' ? 100 : 600} fgColor="#000000" />  
  
​ {extra?.qrName?.checked && <div className="qrcode-text">{extra?.qrName?.data || ''}</div>}  
  
​ </div>  
  
​ ) : (  
  
 <div className="form-qrcode">  
  
​ <QRCode id={qrUrl} value={base64Url} size={mode === 'mobile' && !isPC() ? 100 : 600} fgColor="#000000" />  
  
​ <Image width={125} src={downLink} placeholder={true} />  
  
​ {extra?.qrName?.checked && <div className="qrcode-text">{extra?.qrName?.data || ''}</div>}  
  
​ </div>  
  
​ ))}  
  
​ </div>  
  
​ {/\* 包含拥有者，创建者，创建时间，上次更新时间 \*/}  
  
 <div className="print-subTitle">  
  
​ {sysList  
  
​ ?.filter(item => item.checked)  
  
​ .map(item => {  
  
​ const { label, name, value } = item;  
  
​ return value || setting?.showEmpty ? (  
  
 <div key={name} className="item">  
  
​ <span className="label">{label}</span>  
  
​ <span className="value">{value}</span>  
  
​ </div>  
  
​ ) : (  
  
​ <></>  
  
​ );  
  
​ })}  
  
​ </div>  
  
​ </div>  
  
​ {/\* 打印的核心内容区域，布局和编辑表单一致 \*/}  
  
​ {list?.length > 0 && (  
  
 <div className="print-list">  
  
​ {list.map((arr, index) => {  
  
​ return (  
  
​ <div  
  
​ className={  
  
​ arr?.[0]?.type === 'section'  
  
​ ? 'row row-1'  
  
​ : `row row-${arr?.filter(item => (!setting?.showEmpty ? item.checked && item.v : item.checked))?.length || 0}`  
  
​ }  
  
​ key={index}  
  
​ style={{  
  
​ fontSize: setting?.fontSize || 16,  
  
​ borderBottom:  
  
​ arr?.filter(item => (!setting?.showEmpty ? item.checked && item.v : item.checked))?.length > 0  
  
​ ? '1px solid #8d8d8d'  
  
​ : '0px solid #8d8d8d',  
  
​ }}  
  
​ \>  
  
​ {arr?.length > 0 ? (  
  
​ arr.map(item => {  
  
​ const { name, label, v, type, checked, options, setting, picker } = item;  
  
​ return checked && (v || setting?.showEmpty || type === 'section') ? (  
  
​ <>  
  
​ {type === 'rate' && (  
  
 <div className="item" key={name}>  
  
​ <span className="label">{label}</span>  
  
​ <span className="value">{setting?.showCustomResult ? setting?.customResult?.[v] : v}</span>  
  
​ </div>  
  
​ )}  
  
​ {(type === 'attachment' || type === 'media') && (  
  
 <div className="item" key={name}>  
  
​ <span className="label">{label}</span>  
  
​ <span className="value">  
  
​ {v?.map?.(item => {  
  
​ return item.url?.split('/')?.reverse()?.[0] + ' ';  
  
​ })}  
  
​ </span>  
  
​ </div>  
  
​ )}  
  
​ {type === 'date' && (  
  
 <div className="item" key={name}>  
  
​ <span className="label">{label}</span>  
  
​ <span className="value">{getDate(v, picker)}</span>  
  
​ </div>  
  
​ )}  
  
​ {![  
  
​ 'attachment',  
  
​ 'media',  
  
​ 'cascade',  
  
​ 'other-form-field',  
  
​ 'radio',  
  
​ 'checkbox',  
  
​ 'section',  
  
​ 'marker',  
  
​ 'polygon',  
  
​ 'rate',  
  
​ 'date',  
  
​ ].includes(type) && (  
  
 <div className="item" key={name}>  
  
​ <span className="label">{label}</span>  
  
​ <span className="value">{v || ''}</span>  
  
​ </div>  
  
​ )}  
  
​ {(type === 'cascade' || type === 'other-form-field') && (  
  
 <div className="item" key={name}>  
  
​ <span className="label">{label}</span>  
  
​ <span className="value">{getValue(v)}</span>  
  
​ </div>  
  
​ )}  
  
​ {(type === 'radio' || type === 'checkbox') && (  
  
 <div className="item" key={name}>  
  
​ <span className="label">{label}</span>  
  
​ {setting?.showAllOptions ? (  
  
​ <span className="value value-opts">  
  
​ {options?.map(option => {  
  
​ return (  
  
​ <span className="option">  
  
​ <span  
  
​ className="box"  
  
​ style={{  
  
​ minWidth: setting?.fontSize || 16,  
  
​ height: setting?.fontSize || 16,  
  
​ lineHeight: `${setting?.fontSize || 16}px`,  
  
​ }}  
  
​ \>  
  
​ {(Array.isArray(v) ? v : v?.split(','))?.includes(option) ? '√' : ''}  
  
​ </span>  
  
​ <span className="value">{option}</span>  
  
​ </span>  
  
​ );  
  
​ })}  
  
​ </span>  
  
​ ) : (  
  
​ <span className="value">{v?.join?.('，') || v}</span>  
  
​ )}  
  
​ </div>  
  
​ )}  
  
​ {type === 'section' && (  
  
 <div className="section">  
  
​ <span className="label" style={{ fontSize: setting?.fontSize \* 1.5 || 24 }}>  
  
​ {label}  
  
​ </span>  
  
​ </div>  
  
​ )}  
  
​ </>  
  
​ ) : (  
  
​ <></>  
  
​ );  
  
​ })  
  
​ ) : (  
  
​ <></>  
  
​ )}  
  
​ </div>  
  
​ );  
  
​ })}  
  
​ </div>  
  
​ )}  
  
​ {/\* 如果存在他表字段，展示他表字段 \*/}  
  
 <div className="print-subs">  
  
 <div className="subs-core">  
  
​ {(subs || []).map(sub => {  
  
​ if (sub?.length) {  
  
​ const { label, type, id, formUID, name } = sub?.[0]?.[0] || {};  
  
​ return (  
  
 <div className="record-list">  
  
​ {/\* {sub.map(item => {  
  
​ const data = (item || [])  
  
​ .map(c => {  
  
​ return { ...c.data, checked: c.checked };  
  
​ })  
  
​ .filter(item => item.checked);  
  
​ const { label, type, id, formUID } = item?.[0] || {};  
  
​ return (  
  
​ <SubItem  
  
​ title={label}  
  
​ data={data}  
  
​ type={type}  
  
​ formUID={formUID}  
  
​ recordID={id}  
  
​ mode={mode}  
  
​ setting={setting}  
  
​ updateSubType={updateSubLayout}  
  
​ />  
  
​ );  
  
​ })} \*/}  
  
​ <SubList  
  
​ title={label}  
  
​ arr={sub.map(item =>  
  
​ (item || [])  
  
​ .map(c => {  
  
​ return { ...c.data, checked: c.checked };  
  
​ })  
  
​ .filter(item => item.checked)  
  
​ )}  
  
​ type={type}  
  
​ formUID={formUID}  
  
​ recordID={id}  
  
​ name={name}  
  
​ mode={mode}  
  
​ setting={setting}  
  
​ updateSubType={updateSubLayout}  
  
​ />  
  
​ </div>  
  
​ );  
  
​ } else {  
  
​ return <></>;  
  
​ }  
  
​ })}  
  
​ </div>  
  
​ </div>  
  
​ {/\* 打印时间 \*/}  
  
 <div className="print-footer">  
  
​ {extra?.printTime?.checked && (  
  
 <div className="print-time">  
  
​ <span className="label">打印时间：</span>  
  
​ <span className="time">{moment().format('YYYY-MM-DD HH:mm:ss')}</span>  
  
​ </div>  
  
​ )}  
  
​ </div>  
  
​ </div>  
  
 );  
  
};  
  
   
  
export default PrintPage;  
  
   
  
import React, { PureComponent } from 'react';  
  
import { Selector, A } from '@/components';  
  
import api from '@/parkaccess-manager/model';  
  
// import { request } from '@/utils';  
  
// import { Button } from 'antd';  
  
import InfoModal from './components/infoModal';  
  
import ProgressModal from './components/progressModal';  
  
import SceneModal from './components/sceneModal';  
  
import RelateParkInfo from './components/relateParkInfo';  
  
import './components/index.less';  
  
   
  
class Access extends PureComponent {  
  
 constructor(props) {  
  
​ super(props);  
  
​ const editRight = props.page.hasRight('ACCESS\_PARK\_EDIT');  
  
​ const isYiChang = props.getConfig?.('LinkManager.Page.AccessPark.isYiChang');  
  
​ this.state = {  
  
​ showInfoModal: false,  
  
​ showProgressModal: false,  
  
​ showSceneModal: false,  
  
​ showRelateParkInfo: false,  
  
​ record: null,  
  
​ isYiChang: isYiChang,  
  
​ columns: [  
  
​ {  
  
​ title: '车场编号',  
  
​ dataIndex: 'parkCode',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '车场名称',  
  
​ dataIndex: 'parkName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '所在城市',  
  
​ dataIndex: 'cityName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '所在城区',  
  
​ dataIndex: 'districtName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '所在街道',  
  
​ dataIndex: 'streetName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '车场地址',  
  
​ dataIndex: 'address',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '停车场性质',  
  
​ dataIndex: 'parkProperty',  
  
​ source: {  
  
​ api: api.dictionary.selectoption,  
  
​ param: { dicName: 'PARK\_PROPERTY' },  
  
​ field: 'options',  
  
​ map: item => ({ value: parseInt(item.value, 10), text: item.text }),  
  
​ },  
  
​ width: 140,  
  
​ },  
  
​ {  
  
​ title: '停车位置类型',  
  
​ dataIndex: 'parkLocation',  
  
​ width: 160,  
  
​ source: {  
  
​ api: api.dictionary.selectoption,  
  
​ param: { dicName: 'PARK\_LOCATION' },  
  
​ field: 'options',  
  
​ map: item => ({ value: parseInt(item.value, 10), text: item.text }),  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '总泊位数',  
  
​ dataIndex: 'totalBerthNum',  
  
​ filter: {  
  
​ type: 'number',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '开放泊位数',  
  
​ dataIndex: 'totalCount',  
  
​ filter: {  
  
​ type: 'number',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '经营单位',  
  
​ dataIndex: 'userGroupName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '开发单位',  
  
​ dataIndex: 'developmentUnitUserGroupName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '接入状态',  
  
​ dataIndex: 'projectStatus',  
  
​ filter: {  
  
​ items: [  
  
​ // { value: 0, text: '新登记', type: 'Tag', color: 'geekblue' },  
  
​ { value: 1, text: '联调中', type: 'Tag', color: 'blue' },  
  
​ { value: 2, text: '待上线', type: 'Tag', color: 'magenta' },  
  
​ { value: 3, text: '已上线', type: 'Tag', color: 'green' },  
  
​ { value: 4, text: '已下线', type: 'Tag', color: 'red' },  
  
​ { value: 5, text: '审核中', type: 'Tag', color: 'volcano' },  
  
​ ],  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '接入场景',  
  
​ key: 'action',  
  
​ fixed: 'right',  
  
​ render: (text, record) => (  
  
​ <span>  
  
​ <A onClick={() => this.setState({ record, showSceneModal: true })}>查看</A>  
  
​ </span>  
  
​ ),  
  
​ },  
  
​ {  
  
​ title: '普查关联',  
  
​ key: 'action',  
  
​ fixed: 'right',  
  
​ // dataIndex: 'status',  
  
​ // filter: {  
  
​ // items: [  
  
​ // { value: 0, text: '未关联' },  
  
​ // { value: 1, text: '已关联' },  
  
​ // ],  
  
​ // },  
  
​ render: (text, record) => (  
  
​ <span>  
  
​ {  
  
​ record.status !== 1 ?  
  
​ <A onClick={() => this.handleToRelated(record)}>去关联</A> :  
  
​ <A onClick={() => this.setState({ record, showRelateParkInfo: true })} style={{color: '#1890ff'}}>已关联</A>  
  
​ }  
  
​ </span>  
  
​ ),  
  
​ },  
  
​ {  
  
​ title: '操作',  
  
​ key: 'action',  
  
​ width: 100,  
  
​ fixed: 'right',  
  
​ render: (text, record) => (  
  
​ <span>  
  
​ {/\* {(record.projectStatus === 0 || record.projectStatus === 1) && editRight && <A size="small" type="primary" onClick={() => this.setState({ showInfoModal: true, record })}>编辑</A>} \*/}  
  
​ {<A size="small" type="primary"  
  
​ onClick={() => this.props.openTab({  
  
​ rightID: `park${record.parkID}`,  
  
​ path: 'ParkAccessManagerParkInfo',  
  
​ displayName: `编辑项目-${record.parkName}`,  
  
​ }, {  
  
​ parkID: record.parkID,  
  
​ record: record,  
  
​ })}>{record.projectStatus === 5 ? '审核' : editRight ? '编辑' : ''}</A>  
  
​ }  
  
​ {/\* <Button size="small" type="primary" onClick={() => this.setState({ record, showProgressModal: true })}>查看进度</Button> \*/}  
  
​ </span>  
  
​ ),  
  
​ },  
  
​ ],  
  
​ columnsYiChang: [  
  
​ {  
  
​ title: '车场编号',  
  
​ dataIndex: 'parkCode',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '车场名称',  
  
​ dataIndex: 'parkName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ width: 220,  
  
​ },  
  
​ {  
  
​ title: '所在城市',  
  
​ dataIndex: 'cityName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '所在区县',  
  
​ dataIndex: 'districtName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '总泊位数',  
  
​ dataIndex: 'totalCount',  
  
​ filter: {  
  
​ type: 'number',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '剩余泊位数',  
  
​ dataIndex: 'intrinsictotal',  
  
​ filter: {  
  
​ type: 'number',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '经营单位',  
  
​ dataIndex: 'userGroupName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ width: 200,  
  
​ },  
  
​ {  
  
​ title: '接入时间',  
  
​ dataIndex: 'createTime',  
  
​ filter: {  
  
​ type: 'date',  
  
​ },  
  
​ width: 180,  
  
​ },  
  
​ {  
  
​ title: '普查关联',  
  
​ key: 'action',  
  
​ fixed: 'right',  
  
​ render: (text, record) => (  
  
​ <span>  
  
​ {  
  
​ record.status !== 1 ?  
  
​ <A onClick={() => this.handleToRelated(record)}>去关联</A> :  
  
​ <A onClick={() => this.setState({ record, showRelateParkInfo: true })} style={{color: '#1890ff'}}>已关联</A>  
  
​ }  
  
​ </span>  
  
​ ),  
  
​ },  
  
​ ],  
  
​ };  
  
 }  
  
   
  
 async handleToRelated(record) {  
  
​ const page = {  
  
​ rightID: `accessinfo-${record.parkCode}`,  
  
​ displayName: `${record.parkName}-关联普查车场`,  
  
​ path: 'PageLayout',  
  
​ pageID: 'ParkAccessAccessInfoTab'  
  
​ };  
  
​ if (this.state.isYiChang) {  
  
​ page.path = 'RelatePark';  
  
​ delete page.pageID;  
  
​ }  
  
​ const params = {  
  
​ record: record,  
  
​ isYiChang: this.state.isYiChang,  
  
​ };  
  
​ this.props?.openTab && this.props.openTab(page, { ...params });  
  
 }  
  
   
  
 modalClose = () => {  
  
​ this.setState({ showInfoModal: false, showProgressModal: false, showSceneModal: false, showRelateParkInfo: false, record: null });  
  
 }  
  
   
  
 reload = () => {  
  
​ this.refs.table && this.refs.table.reload();  
  
 }  
  
   
  
 render() {  
  
​ const { showInfoModal, showProgressModal, showSceneModal, showRelateParkInfo, record, columns, columnsYiChang, isYiChang } = this.state;  
  
​ const table = {  
  
​ className: "table-time-colmuns",  
  
​ rowKey: record => `${record.parkID}-${record.parkCode}`,  
  
​ };  
  
​ return (  
  
 <div>  
  
​ <Selector  
  
​ {...this.props}  
  
​ ref='table'  
  
​ columns={isYiChang ? columnsYiChang : columns}  
  
​ table={table}  
  
​ api={isYiChang ? api.park.getyichangparklist : api.park.getparklist}  
  
​ param={{ systemType: 1 }}  
  
​ refresh={!isYiChang}  
  
​ />  
  
​ {  
  
​ showInfoModal &&  
  
​ <InfoModal  
  
​ visible={showInfoModal}  
  
​ modalClose={this.modalClose}  
  
​ reload={this.reload}  
  
​ record={record}  
  
​ token={this.props.user?.token}  
  
​ {...this.props}  
  
​ />  
  
​ }  
  
​ {  
  
​ showProgressModal &&  
  
​ <ProgressModal  
  
​ visible={showProgressModal}  
  
​ modalClose={this.modalClose}  
  
​ reload={this.reload}  
  
​ />  
  
​ }  
  
​ {  
  
​ showSceneModal &&  
  
​ <SceneModal  
  
​ visible={showSceneModal}  
  
​ modalClose={this.modalClose}  
  
​ reload={this.reload}  
  
​ record={record}  
  
​ />  
  
​ }  
  
​ {  
  
​ showRelateParkInfo &&  
  
​ <RelateParkInfo  
  
​ visible={showRelateParkInfo}  
  
​ modalClose={this.modalClose}  
  
​ reload={this.reload}  
  
​ record={record}  
  
​ isYiChang={isYiChang}  
  
​ />  
  
​ }  
  
​ </div>  
  
​ );  
  
 }  
  
}  
  
   
  
export default Access;  
  
   
  
import React, { PureComponent } from 'react';  
  
import { Selector, A } from '@/components';  
  
import api, { ENUM\_SYSTEMHOOK } from '@/parkaccess-manager/model';  
  
import { request } from '@/utils';  
  
import RelateParkInfo from './components/relateParkInfo';  
  
import { message, Popconfirm } from 'antd';  
  
import './components/index.less';  
  
   
  
class RelatePark extends PureComponent {  
  
 constructor(props) {  
  
​ super(props);  
  
​ this.state = {  
  
​ showInfoModal: false,  
  
​ record: null,  
  
​ propRecord: props?.page?.params?.record || {},  
  
​ columns: [  
  
​ {  
  
​ title: '普查车场ID',  
  
​ dataIndex: 'uid',  
  
​ },  
  
​ {  
  
​ title: '车场名称',  
  
​ dataIndex: 'parkName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '经营单位',  
  
​ dataIndex: 'userGroupName',  
  
​ },  
  
​ {  
  
​ title: '所在城区',  
  
​ dataIndex: 'districtName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '所在街道',  
  
​ dataIndex: 'streetName',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '所在地址',  
  
​ dataIndex: 'address',  
  
​ filter: {  
  
​ type: 'input',  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '车场性质',  
  
​ dataIndex: 'parkProperty',  
  
​ width: 160,  
  
​ source: {  
  
​ api: api.dictionary.selectoption,  
  
​ field: 'options',  
  
​ param: { dicName: 'PARK\_PROPERTY' },  
  
​ },  
  
​ filter: {  
  
​ type: 'none',  
  
​ },  
  
​ otherValue: ' ',  
  
​ },  
  
​ {  
  
​ title: '业态',  
  
​ dataIndex: 'tradeID',  
  
​ source: {  
  
​ api: api.dictionary.selectoption,  
  
​ field: 'options',  
  
​ param: { dicName: 'TRADE\_TYPE' },  
  
​ },  
  
​ filter: {  
  
​ type: 'none',  
  
​ },  
  
​ otherValue: ' ',  
  
​ },  
  
​ {  
  
​ title: '停车位置',  
  
​ dataIndex: 'parkLocation',  
  
​ width: 160,  
  
​ source: {  
  
​ api: api.dictionary.selectoption,  
  
​ field: 'options',  
  
​ param: { dicName: 'PARK\_LOCATION' },  
  
​ },  
  
​ filter: {  
  
​ type: 'none',  
  
​ },  
  
​ otherValue: ' ',  
  
​ },  
  
​ {  
  
​ title: '关联状态',  
  
​ dataIndex: 'status',  
  
​ filter: {  
  
​ items: [  
  
​ { value: 0, text: '未关联', type: 'Tag', color: 'blue' },  
  
​ { value: 1, text: '已关联', type: 'Tag', color: 'green' },  
  
​ ],  
  
​ },  
  
​ },  
  
​ {  
  
​ title: '操作',  
  
​ key: 'action',  
  
​ width: 130,  
  
​ fixed: 'right',  
  
​ render: (text, record) =>  
  
​ record.status === 1 ? (  
  
​ <span>  
  
​ <Popconfirm  
  
​ placement="topRight"  
  
​ title={`确定取消该车场的关联车场吗?`}  
  
​ onConfirm={() => this.cancelRelate(record)}  
  
​ okText="确定"  
  
​ cancelText="取消"  
  
​ \>  
  
​ <A style={{ marginRight: '10px' }}>取消关联</A>  
  
​ </Popconfirm>  
  
​ <A onClick={() => this.setState({ showInfoModal: true, record })}>详情</A>  
  
​ </span>  
  
​ ) : (  
  
​ <span>  
  
​ <A onClick={() => this.addRelate(record)} style={{ marginRight: '10px' }}>  
  
​ 关联  
  
​ </A>  
  
​ <A onClick={() => this.setState({ showInfoModal: true, record })}>详情</A>  
  
​ </span>  
  
​ ),  
  
​ },  
  
​ ],  
  
​ };  
  
 }  
  
   
  
 async cancelRelate(record) {  
  
​ const result = await request(api.park.deleteinparkid, { outParkID: record.uid });  
  
​ if (result.success) {  
  
​ message.success('取消关联成功');  
  
​ this.reload();  
  
​ this.props.systemHook({ type: ENUM\_SYSTEMHOOK.REFRESH\_PAGE, data: 'NAVIGATION\_ACCESS\_MANAGERMENT\_BACKGROUND\_ACCESSINFO' });  
  
​ } else {  
  
​ message.error(result.message || '取消关联失败');  
  
​ }  
  
 }  
  
   
  
 async addRelate(record) {  
  
​ const result = await request(api.park.addinparkid, {  
  
​ parkID: this.state.propRecord.parkID,  
  
​ outParkID: record.uid,  
  
​ longitude: record.longitude,  
  
​ latitude: record.latitude,  
  
​ });  
  
​ if (result.success) {  
  
​ message.success('关联成功');  
  
​ // this.reload();  
  
​ this.props.systemHook({ type: ENUM\_SYSTEMHOOK.REFRESH\_PAGE, data: 'NAVIGATION\_ACCESS\_MANAGERMENT\_BACKGROUND\_ACCESSINFO' });  
  
​ this.props.systemHook({ type: ENUM\_SYSTEMHOOK.CLOSE\_PAGE, data: 'current' });  
  
​ } else {  
  
​ message.error(result.message || '关联失败');  
  
​ }  
  
 }  
  
   
  
 modalClose = () => {  
  
​ this.setState({ showInfoModal: false, record: null });  
  
 };  
  
   
  
 reload = () => {  
  
​ this.refs.table && this.refs.table.reload();  
  
 };  
  
   
  
 render() {  
  
​ const { showInfoModal, record } = this.state;  
  
​ const table = {  
  
​ className: 'table-time-colmuns',  
  
​ };  
  
​ return (  
  
 <div className="group-access-info-relate-park">  
  
​ {/\* <div className="title" style={{height: '44px', lineHeight: '44px'}}>  
  
​ <Row>  
  
​ <Col span={6}>  
  
​ <label>{`车场名称：${propRecord.parkName || ''}`}</label>  
  
​ </Col>  
  
​ <Col span={12}>  
  
​ <label>{`车场地址：${propRecord.aaaaaa || ''}`}</label>  
  
​ </Col>  
  
​ </Row>  
  
​ </div> \*/}  
  
​ <Selector  
  
​ // {...this.props}  
  
​ ref="table"  
  
​ columns={this.state.columns}  
  
​ table={table}  
  
​ api={api.park.getoutparklist}  
  
​ />  
  
​ {showInfoModal && (  
  
​ <RelateParkInfo  
  
​ {...this.props}  
  
​ visible={showInfoModal}  
  
​ modalClose={this.modalClose}  
  
​ reload={this.reload}  
  
​ record={record}  
  
​ isYiChang={this.props.page?.params?.isYiChang || false}  
  
​ />  
  
​ )}  
  
​ </div>  
  
​ );  
  
 }  
  
}  
  
   
  
export default RelatePark;  
  
   
  
import React, { useMemo } from 'react';  
  
import { Card, Descriptions, Image } from 'antd';  
  
import { Selector } from '@/components';  
  
import api from '@/parkaccess-manager/model';  
  
import './index.less';  
  
   
  
export default props => {  
  
 const record = useMemo(() => props.page?.params?.record || {}, [props.page?.params?.record]);  
  
 const useRealData = props.getConfig?.('LinkManager.Page.AccessPark.useRealData'); // 是否使用真数据，不配置默认使用假数据  
  
 const fakeDataParkCode = props.getConfig?.('LinkManager.Page.AccessPark.fakeDataParkCode'); // 使用假数据的车场  
  
 const isFakeData = useMemo(() => record.parkCode === (fakeDataParkCode || '71711301689') && !useRealData, [record, fakeDataParkCode, useRealData]); // 是否使用假数据  
  
 const columnsAccess = useMemo(  
  
​ () => [  
  
​ {  
  
​ title: '车场名称',  
  
​ dataIndex: 'parkName',  
  
​ width: 220,  
  
​ render: () => record.parkName,  
  
​ },  
  
​ {  
  
​ title: '车牌',  
  
​ dataIndex: 'plateNo',  
  
​ },  
  
​ {  
  
​ title: '进场时间',  
  
​ dataIndex: 'inTime',  
  
​ },  
  
​ {  
  
​ title: '出场时间',  
  
​ dataIndex: 'outTime',  
  
​ },  
  
​ {  
  
​ title: '进场图片',  
  
​ dataIndex: 'in\_picture\_url',  
  
​ key: 'action',  
  
​ render: value => <Image src={value} style={{ height: 32 }} />,  
  
​ },  
  
​ {  
  
​ title: '出场图片',  
  
​ dataIndex: 'out\_picture\_url',  
  
​ key: 'action',  
  
​ render: value => <Image src={value} style={{ height: 32 }} />,  
  
​ },  
  
​ ],  
  
​ [record]  
  
 );  
  
   
  
 const columnsBill = useMemo(  
  
​ () => [  
  
​ {  
  
​ title: '车场名称',  
  
​ dataIndex: 'parkName',  
  
​ width: 220,  
  
​ render: () => record.parkName,  
  
​ },  
  
​ {  
  
​ title: '车牌',  
  
​ dataIndex: 'plateNo',  
  
​ },  
  
​ {  
  
​ title: '账单流水号',  
  
​ dataIndex: 'billNo',  
  
​ },  
  
​ {  
  
​ title: '账单时间',  
  
​ dataIndex: 'billTime',  
  
​ },  
  
​ {  
  
​ title: '账单金额',  
  
​ dataIndex: 'money',  
  
​ },  
  
​ ],  
  
​ [record]  
  
 );  
  
 console.log(record);  
  
 return (  
  
 <div className="parkaccess-manager-yichang-access-info" style={{ padding: 20 }}>  
  
​ <Card title="车场信息" style={{ marginBottom: 20, borderRadius: 12 }} headStyle={{ fontWeight: 'bold' }}>  
  
​ <Descriptions column={2}>  
  
​ <Descriptions.Item label="停车场编号">{record.parkCode}</Descriptions.Item>  
  
​ <Descriptions.Item label="车场名称">{record.parkName}</Descriptions.Item>  
  
​ <Descriptions.Item label="所在城市">{record.cityName}</Descriptions.Item>  
  
​ <Descriptions.Item label="所在区县">{record.districtName}</Descriptions.Item>  
  
​ <Descriptions.Item label="总泊位数">{record.totalCount}</Descriptions.Item>  
  
​ <Descriptions.Item label="剩余泊位">{record.intrinsictotal}</Descriptions.Item>  
  
​ <Descriptions.Item label="经营单位">{record.userGroupName}</Descriptions.Item>  
  
​ <Descriptions.Item label="计费规则">{isFakeData ? '3元/小时' : record.rateInfo}</Descriptions.Item>  
  
​ </Descriptions>  
  
​ </Card>  
  
​ <Card title="进出记录" style={{ marginBottom: 20, borderRadius: 12 }} headStyle={{ fontWeight: 'bold' }}>  
  
​ <Selector  
  
​ columns={columnsAccess}  
  
​ api={  
  
​ isFakeData  
  
​ ? {  
  
​ \_result: {  
  
​ success: true,  
  
​ data: {  
  
​ data: [  
  
​ { plateNo: '鄂B6U90C', inTime: '2022-09-15 09:03:42', outTime: '2022-09-15 18:23:45' },  
  
​ { plateNo: '鄂B9BV32', inTime: '2022-09-14 19:13:04', outTime: '2022-09-15 17:43:15' },  
  
​ { plateNo: '鄂CU90B3', inTime: '2022-09-14 11:24:41', outTime: '2022-09-15 17:21:04' },  
  
​ { plateNo: '鄂A8H762', inTime: '2022-09-13 08:53:14', outTime: '2022-09-15 17:13:42' },  
  
​ { plateNo: '鄂BN67G8', inTime: '2022-09-15 14:21:35', outTime: '2022-09-15 16:53:22' },  
  
​ ],  
  
​ },  
  
​ },  
  
​ }  
  
​ : api.park.getyichangrecordlist  
  
​ }  
  
​ listField="data"  
  
​ param={{ parkID: record.parkID }}  
  
​ autoHeight={false}  
  
​ enabledExport={false}  
  
​ refresh={false}  
  
​ pagination={false}  
  
​ />  
  
​ </Card>  
  
​ <Card title="账单记录" style={{ marginBottom: 20, borderRadius: 12 }} headStyle={{ fontWeight: 'bold' }}>  
  
​ <Selector  
  
​ columns={columnsBill}  
  
​ api={  
  
​ isFakeData  
  
​ ? {  
  
​ \_result: {  
  
​ success: true,  
  
​ data: {  
  
​ list: [  
  
​ { plateNo: '鄂B6U90C', billNo: '4200001587202209129489322653', billTime: '2022-09-15 18:23:45', money: 12 },  
  
​ { plateNo: '鄂B9BV32', billNo: '4200001587202209129488792323', billTime: '2022-09-15 17:43:15', money: 36 },  
  
​ { plateNo: '鄂CU90B3', billNo: '4200001587202209129485928322', billTime: '2022-09-15 17:21:04', money: 21 },  
  
​ { plateNo: '鄂A8H762', billNo: '4200001587202209129484237238', billTime: '2022-09-15 17:13:42', money: 54 },  
  
​ { plateNo: '鄂BN67G8', billNo: '4200001587202209129482334552', billTime: '2022-09-15 16:53:22', money: 6 },  
  
​ ],  
  
​ },  
  
​ },  
  
​ }  
  
​ : { \_result: { success: true, data: [] } }  
  
​ }  
  
​ autoHeight={false}  
  
​ enabledExport={false}  
  
​ refresh={false}  
  
​ pagination={false}  
  
​ />  
  
​ </Card>  
  
​ </div>  
  
 );  
  
};  
  
   
  
.parkaccess-manager-yichang-access-overview {  
  
 padding: 20px 20px 0;  
  
   
  
 .overview-chart-title {  
  
​ padding-right: 1px;  
  
​ padding-top: 15px;  
  
​ height: 50px;  
  
​ user-select: none;  
  
​ .icon {  
  
​ display: inline-block;  
  
​ top: 16px;  
  
​ width: 3px;  
  
​ height: 25px;  
  
​ border-top-right-radius: 3px; // 右上角圆角  
  
​ border-bottom-right-radius: 3px; // 右下角圆角  
  
​ background-color: #4094ff;  
  
​ }  
  
​ .title-name {  
  
​ position: absolute;  
  
​ margin-left: 18px;  
  
​ font-size: 18px;  
  
​ font-weight: bold;  
  
​ }  
  
 }  
  
   
  
 .overview-chart-wrap {  
  
​ border-radius: 8px;  
  
​ box-shadow: 0 2px 6px 2px rgb(84 48 132 / 10%);  
  
​ cursor: grab;  
  
​ height: 200px;  
  
​ margin-bottom: 24px;  
  
   
  
​ .number-chart-number-wrap {  
  
​ border-radius: 8px;  
  
​ width: 100%;  
  
​ height: calc(100% - 40px);  
  
​ position: relative;  
  
​ display: flex;  
  
​ align-items: center;  
  
​ justify-content: center;  
  
​ .number {  
  
​ user-select: none;  
  
​ .value {  
  
​ font-weight: bold;  
  
​ font-size: 48px;  
  
​ }  
  
​ .unit {  
  
​ margin-left: 16px;  
  
​ font-size: 24px;  
  
​ }  
  
​ }  
  
​ canvas {  
  
​ border-bottom-right-radius: 8px;  
  
​ border-bottom-left-radius: 8px;  
  
​ }  
  
​ }  
  
 }  
  
}  
  
   
  
.parkaccess-manager-yichang-access-info {  
  
 .selector {  
  
​ .tag-group {  
  
​ min-height: 0;  
  
​ height: 0;  
  
​ }  
  
 }  
  
}