<?php  
  
namespace App\Http\Controllers\Admin;  
  
use App\Enums\CommonEnum;  
use App\Exceptions\ApiException;  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Admin\AdminRequest;  
use App\Http\Resources\Admin\AdminResource;  
use App\Models\Admin;  
use App\Services\AdminService;  
  
class AdminController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(AdminService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function index(AdminRequest $request)  
 {  
 $paginator = $this->service->getPaginator();  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => AdminResource::collection($paginator),  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function store(AdminRequest $request)  
 {  
 $params = $request->only($this->service->postFields());  
 $this->service->createOrUpdateAdmin($params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function show($id)  
 {  
 return $this->rspSuccessData(new AdminResource(Admin::query()->findOrFail($id)));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function update(AdminRequest $request, $id)  
 {  
 $params = $request->only($this->service->postFields());  
 $params['id'] = $id;  
 $this->service->createOrUpdateAdmin($params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function destroy($id)  
 {  
 Admin::destroy($id);  
 return $this->rspSuccessData();  
 }  
  
 public function options(AdminRequest $request)  
 {  
 $res = $this->service->getOptions($request->keyword);  
 return $this->rspSuccessData($res);  
 }  
}

<?php  
  
namespace App\Http\Controllers\Admin;  
  
use App\Enums\CommonEnum;  
use App\Exceptions\ApiException;  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Admin\AuthRequest;  
use App\Http\Resources\Admin\AdminInfoResource;  
use App\Http\Resources\Admin\AdminResource;  
use App\Models\Admin;  
use App\Services\AdminService;  
use App\Services\AuthService;  
use App\Services\LoginRecordService;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Auth;  
  
class AuthController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(AdminService $service)  
 {  
 $this->service = $service;  
 }  
  
 public function login(Request $request)  
 {  
 $credentials = $request->only(['username', 'password']);  
 $admin = Admin::query()->where('username',$credentials['username'])->first();  
 if ($admin && $admin->status != CommonEnum::ACCOUNT\_STATUS\_ON){  
 throw new ApiException('账号已被禁用，请联系管理员');  
 }  
 if (Auth::attempt($credentials)) {  
 $admin = $this->service->getByUsername($credentials['username']);  
 Auth::login($admin);  
 app(LoginRecordService::class)->createLoginRecord($admin,[  
 'ip' => $request->ip(),  
 'ua' => $request->userAgent(),  
 ]);  
 $this->service->clear\_admin\_cache($this->user()->id); //删除用户下的所有缓存  
 return $this->rspSuccessData(new AdminResource($this->user()));  
 }  
 return $this->rspFailMsg('账号或密码有误');  
 }  
  
 public function info()  
 {  
 $admin = Auth::user();  
 return $this->rspSuccessData(new AdminInfoResource($admin));  
 }  
  
 public function logout()  
 {  
 Auth::logout();  
 return $this->rspSuccessData();  
 }  
  
 public function changePassword(AuthRequest $request)  
 {  
 $this->service->changePassword(Auth::user(),$request->only([  
 'old\_password','password'  
 ]));  
 return $this->rspSuccessData();  
 }  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/13  
 \* Time 3:44 下午  
 \*/  
  
namespace App\Http\Controllers\Common;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Common\AreaRequest;  
use App\Http\Resources\Common\AreaResource;  
use App\Models\Area;  
use App\Services\Common\AreaService;  
  
class AreaController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(AreaService $service)  
 {  
 $this->service = $service;  
 }  
  
 public function index()  
 {  
  
 }  
  
 public function show(Area $area)  
 {  
 return $this->rspSuccessData(new AreaResource($area));  
 }  
  
 public function store(AreaRequest $request)  
 {  
 $params = $request->only($this->service->postFields());  
 $area = $this->service->createOrUpdateArea($params);  
 return $this->rspSuccessData(new AreaResource($area));  
 }  
  
 public function update(AreaRequest $request,$id)  
 {  
 $params = array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'id' => $id  
 ]  
 );  
 $area = $this->service->createOrUpdateArea($params);  
 return $this->rspSuccessData(new AreaResource($area));  
 }  
  
 public function destroy($id)  
 {  
 //todo  
 }  
  
 public function options()  
 {  
 $tree = $this->service->getOptions();  
 return $this->rspSuccessData($tree);  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/11  
 \* Time 7:58 下午  
 \*/  
  
namespace App\Http\Controllers\Common;  
  
use App\Exceptions\ApiException;  
use App\Http\Controllers\Controller;  
use App\Services\Tool\OpenToolService;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Storage;  
  
class CommonController extends Controller  
{  
 public function upload\_single(Request $request)  
 {  
 $file = $request->file('file');  
 $path = '/common/uploads/';  
 $name = uniqid();  
 if ($request->input('is\_img',1)){  
 $name .= '.png';  
 }  
 $fullName = $path.$name;  
 $isSuccess = Storage::disk('cosv5')->putFileAs($path,$file,$name);  
 if (false === $isSuccess){  
 throw new ApiException('上传失败');  
 }  
 $url = OpenToolService::getObjectUrl($fullName);  
 return $this->rspSuccessData([  
 'path' => $fullName,  
 'url' => $url  
 ]);  
 }  
}

<?php  
  
namespace App\Http\Controllers\Common;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Common\DoctorRequest;  
use App\Http\Resources\Common\DoctorOptionResource;  
use App\Http\Resources\Common\DoctorResource;  
use App\Services\Common\DoctorService;  
use Illuminate\Http\Request;  
  
class DoctorController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(DoctorService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function index(DoctorRequest $request)  
 {  
 $paginator = $this->service->getPaginator($this->user()['id']);  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => DoctorResource::collection($paginator)  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function store(DoctorRequest $request)  
 {  
 $params = $request->only($this->service->postFields());  
 $doctor = $this->service->createOrUpdateDoctor($params);  
 return $this->rspSuccessData(new DoctorResource($doctor));  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function show($id)  
 {  
 $doctor = $this->service->getDoctorDetail($id);  
 return $this->rspSuccessData(new DoctorResource($doctor));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function update(DoctorRequest $request, $id)  
 {  
 $params = array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'id' => $id,  
 ]  
 );  
 $doctor = $this->service->createOrUpdateDoctor($params);  
 return $this->rspSuccessData(new DoctorResource($doctor));  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function destroy($id)  
 {  
 $this->service->deleteDoctor($id);  
 return $this->rspSuccessData();  
 }  
  
 public function options(DoctorRequest $request)  
 {  
 $doctors = $this->service->getOptions();  
 return $this->rspSuccessData(DoctorOptionResource::collection($doctors));  
 }  
}

<?php  
  
namespace App\Http\Controllers\Common;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Common\HospitalRequest;  
use App\Http\Resources\Common\HospitalResource;  
use App\Models\Hospital;  
use App\Services\Common\HospitalService;  
use Illuminate\Contracts\Foundation\Application;  
use Illuminate\Contracts\Routing\ResponseFactory;  
use Illuminate\Http\Request;  
use Illuminate\Http\Response;  
  
class HospitalController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(HospitalService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return Response  
 \*/  
 public function index(HospitalRequest $request)  
 {  
 $paginator = $this->service->getPaginator($this->user()['id']);  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => HospitalResource::collection($paginator)  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return Response  
 \*/  
 public function store(HospitalRequest $request)  
 {  
 $params = $request->only($this->service->postFields());  
 $hospital = $this->service->createOrUpdateHospital($params);  
 return $this->rspSuccessData(new HospitalResource($hospital));  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param \App\Models\Hospital $hospital  
 \* @return Response  
 \*/  
 public function show(Hospital $hospital)  
 {  
 return $this->rspSuccessData(new HospitalResource($hospital));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param string $id  
 \* @return Response  
 \*/  
 public function update(Request $request, $id)  
 {  
 $params = array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'id' => $id,  
 ]  
 );  
 $hospital = $this->service->createOrUpdateHospital($params);  
 return $this->rspSuccessData(new HospitalResource($hospital));  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param $id  
 \* @return Application|ResponseFactory|Response  
 \*/  
 public function destroy($id)  
 {  
 $this->service->deleteHospital($id);  
 return $this->rspSuccessData();  
 }  
  
 public function options(HospitalRequest $request)  
 {  
 $res = $this->service->getOptions($request->keyword);  
 return $this->rspSuccessData($res);  
 }  
  
 /\*\*  
 \* 所有机构  
 \* @param $id 区域id  
 \*/  
 public function list($id)  
 {  
 $list = $this->service->getHospitals('', $id, false, ['id', 'name']);  
 return $this->rspSuccessData($list);  
 }  
  
}

<?php  
  
namespace App\Http\Controllers\Common;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Common\StandardRequest;  
use App\Http\Resources\Common\StandardResource;  
use App\Services\Common\StandardService;  
use Illuminate\Http\Request;  
  
class StandardController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(StandardService $service)  
 {  
 $this->service = $service;  
 }  
  
 public function index()  
 {  
 $paginator = $this->service->getPaginator();  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => StandardResource::collection($paginator),  
 ]);  
 }  
  
 public function update(StandardRequest $request,$id)  
 {  
 $params = array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'id' => $id  
 ]  
 );  
 $standard = $this->service->updateStandard($params);  
 return $this->rspSuccessData(new StandardResource($standard));  
 }  
  
 public function categories()  
 {  
 $categories = $this->service->getCategoryOptions();  
 return $this->rspSuccessData($categories);  
 }  
}

<?php  
  
namespace App\Http\Controllers\Device;  
  
use App\Exceptions\ApiException;  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Device\DeviceHealthRequest;  
use App\Http\Resources\Device\DeviceHealthResource;  
use App\Services\Device\DeviceGeneralService;  
use Illuminate\Http\Request;  
use Illuminate\Http\Response;  
  
class DeviceGeneralController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(DeviceGeneralService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return Response  
 \*/  
 public function index(DeviceHealthRequest $request)  
 {  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param DeviceHealthRequest $request  
 \* @return Response  
 \* @throws ApiException  
 \*/  
 public function store(DeviceHealthRequest $request)  
 {  
 $params = array\_merge($request->only($this->service->postFields()),  
 [  
 'admin\_id' => $this->user()['id'],  
 ]  
 );  
 $deviceHealth = $this->service->createOrUpdateDeviceHealth($params);  
 return $this->rspSuccessData(new DeviceHealthResource($deviceHealth));  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return Response  
 \*/  
 public function show($id)  
 {  
 //  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return Response  
 \*/  
 public function update(Request $request, $id)  
 {  
 //  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return Response  
 \*/  
 public function destroy($id)  
 {  
 //  
 }  
}

<?php  
  
namespace App\Http\Controllers\Device;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Device\DeviceRecordRequest;  
use App\Http\Resources\Device\DeviceRecordListResource;  
use App\Http\Resources\Device\DeviceRecordResource;  
use App\Services\Device\DeviceRecordService;  
use Illuminate\Contracts\Foundation\Application;  
use Illuminate\Contracts\Routing\ResponseFactory;  
use Illuminate\Http\Request;  
use Illuminate\Http\Response;  
  
class DeviceRecordController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(DeviceRecordService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return Response  
 \*/  
 public function index(DeviceRecordRequest $request)  
 {  
 $documentId = $request->input('document\_id');  
 $paginator = $this->service->getDeviceRecordsByDocumentId($documentId);  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => DeviceRecordListResource::collection($paginator),  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return Response  
 \*/  
 public function store(DeviceRecordRequest $request)  
 {  
 $params = array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'admin\_id' => $this->user()['id'],  
 ]  
 );  
 $record = $this->service->createOrUpdateDeviceRecord($params);  
 return $this->rspSuccessData(new DeviceRecordResource($record));  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return Response  
 \*/  
 public function show($id)  
 {  
 $record = $this->service->getDeviceRecordDetail($id);  
 return $this->rspSuccessData(new DeviceRecordResource($record));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return Response  
 \*/  
 public function update(DeviceRecordRequest $request, $id)  
 {  
 $params = array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'admin\_id' => $this->user()['id'],  
 'id' => $id,  
 ]  
 );  
 $record = $this->service->updateDeviceRecord($params);  
 return $this->rspSuccessData(new DeviceRecordResource($record));  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return Response  
 \*/  
 public function destroy(DeviceRecordRequest $request,$id)  
 {  
 $this->service->deleteDeviceRecord($id);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* 获取公卫体检数据调入  
 \*  
 \* @param DeviceRecordRequest $request  
 \* @return Application|ResponseFactory|Response  
 \*/  
 public function examination\_info(DeviceRecordRequest $request)  
 {  
 $documentId = $request->input('document\_id');  
 $res = $this->service->getExaminationInfo($documentId);  
 return $this->rspSuccessData($res);  
 }  
}

<?php  
  
namespace App\Http\Controllers\Document;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Document\DocumentRequest;  
use App\Http\Resources\Document\DocumentDetailResource;  
use App\Http\Resources\Document\DocumentListResource;  
use App\Http\Resources\Document\DocumentResource;  
use App\Services\Document\DocumentService;  
use Illuminate\Http\Request;  
  
class DocumentController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(DocumentService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function index(DocumentRequest $request)  
 {  
 $paginator = $this->service->getPaginator($this->user()['id']);  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => DocumentListResource::collection($paginator)  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function store(DocumentRequest $request)  
 {  
 $document = $this->service->createOrUpdateDocument(array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'admin\_id' => $this->user()['id']  
 ]  
 ));  
 return $this->rspSuccessData(new DocumentDetailResource($document));  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function show(DocumentRequest $request,$id)  
 {  
 $document = $this->service->getDocumentDetail($id);  
 return $this->rspSuccessData(new DocumentDetailResource($document));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function update(DocumentRequest $request, $id)  
 {  
 $document = $this->service->updateDocument(array\_merge(  
 $request->only($this->service->postFields()),  
 [  
 'id' => $id,  
 'admin\_id' => $this->user()['id']  
 ]  
 ));  
 return $this->rspSuccessData(new DocumentDetailResource($document));  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function destroy($id)  
 {  
 $this->service->deleteDocument($id);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* 搜索建议  
 \*  
 \* @param DocumentRequest $request  
 \* @return mixed  
 \*/  
 public function search\_suggests(DocumentRequest $request)  
 {  
 $documents = $this->service->getSearchSuggests($request->keyword,$this->user()['id']);  
 return $this->rspSuccessData($documents);  
 }  
}

<?php  
  
namespace App\Http\Controllers\Examination;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Examination\ExaminationRequest;  
use App\Http\Resources\Examination\ExaminationDetailResource;  
use App\Http\Resources\Examination\ExaminationListResource;  
use App\Services\Examination\ExaminationService;  
  
class ExaminationController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(ExaminationService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function index(ExaminationRequest $request)  
 {  
 $documentId = $request->input('document\_id');  
 $paginator = $this->service->getPaginator($this->user()['id'],$documentId);  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => ExaminationListResource::collection($paginator)  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function store(ExaminationRequest $request)  
 {  
 $adminId = $this->user()['id'];  
 $params = $request->only($this->service->postFields());  
 $examination = $this->service->createExamination($adminId,$params);  
 return $this->rspSuccessData(new ExaminationListResource($examination));  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function show(ExaminationRequest $request,$id)  
 {  
 $examination = $this->service->getDetail($this->user()['id'],$id);  
 return $this->rspSuccessData(new ExaminationDetailResource($examination));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function update(ExaminationRequest $request, $id)  
 {  
 $params = $request->only($this->service->patchFields());  
 $examination = $this->service->updateExamination($this->user()['id'],$id,$params);  
 return $this->rspSuccessData(new ExaminationListResource($examination));  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function destroy(ExaminationRequest $request,$id)  
 {  
 $this->service->deleteExamination($this->user()['id'],$id);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* 下载报告  
 \* @param $id  
 \*/  
 public function report($id)  
 {  
 $this->service->getReport($id);  
 return $this->rspSuccessData();  
 }  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/16  
 \* Time 2:38 下午  
 \*/  
  
namespace App\Http\Controllers\Index;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Index\DashboardRequest;  
use App\Models\Area;  
use App\Services\AdminService;  
use App\Services\StatisticService;  
  
/\*\*  
 \* 大屏控制器  
 \*/  
class DashboardController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(StatisticService $service)  
 {  
 $this->service = $service;  
 }  
  
 public function statistic()  
 {  
 $res = [  
 'age' => $this->service->getDocumentStatisticByAge($this->user()['id']),  
 'sex' => $this->service->getDocumentStatisticBySex($this->user()['id']),  
 ];  
 return $this->rspSuccessData($res);  
 }  
  
 public function device\_dynamic(DashboardRequest $request)  
 {  
 $lastId = $request->input('last\_id',0);  
 $res = $this->service->getDeviceRecordDynamic($lastId,$this->user()['id']);  
 return $this->rspSuccessData($res);  
 }  
  
 public function abnormal\_statistic()  
 {  
 $admin = $this->user();  
 $res = [  
 'abnormal' => $this->service->getAbnormalStatistic($admin['id']),  
 'disease\_label' => $this->service->getDiseaseLabelStatistic($admin['id']),  
 'residence\_label' => $this->service->getResidenceLableStatistic($admin['id']),  
 ];  
 return $this->rspSuccessData($res);  
 }  
  
 public function center\_statistic(DashboardRequest $request)  
 {  
 $adminId = $this->user()['id'];  
 $count = $this->service->getCenterStatistic($adminId);  
 $graph = $this->service->getTopHospitalsStatistic($adminId);  
 $followUp = $this->service->getFollowUpStatistic($adminId);  
 $res = [  
 'count' => $count,  
 'graph' => $graph,  
 'follow\_up' => $followUp,  
 ];  
 return $this->rspSuccessData($res);  
 }  
  
 public function video\_list(DashboardRequest $request)  
 {  
 list($page,$pageSize,$extra,$streamType,$standardType) = [  
 $request->input('page',1),  
 $request->input('page\_size',10),  
 $request->input('extra',[]),  
 $request->input('streamtype',0),  
 $request->input('standardtype',3),  
 ];  
 $res = $this->service->getVideoList($this->user()['id'],$page,$pageSize,$extra,[  
 'streamtype' => $streamType,  
 'standardtype' => $standardType,  
 ]);  
 return $this->rspSuccessData($res);  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/12  
 \* Time 2:40 下午  
 \*/  
  
namespace App\Http\Controllers\Index;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Index\IndexRequest;  
use App\Services\StatisticService;  
use Illuminate\Contracts\Foundation\Application;  
use Illuminate\Contracts\Routing\ResponseFactory;  
use Illuminate\Http\Response;  
  
/\*\*  
 \* 工作台控制器  
 \*/  
class IndexController extends Controller  
{  
 /\*\*  
 \* 工作台档案统计  
 \*  
 \* @return Application|ResponseFactory|Response  
 \*/  
 public function document\_statistic()  
 {  
 $service = app(StatisticService::class);  
 $data = $service->getIndexStatistic($this->user()['id']);  
 return $this->rspSuccessData($data);  
 }  
  
 /\*\*  
 \* 工作台档案概况  
 \*  
 \* @param IndexRequest $request  
 \* @return Application|ResponseFactory|Response  
 \*/  
 public function document\_graph(IndexRequest $request)  
 {  
 $service = app(StatisticService::class);  
 $data = $service->getDocumentGraphByType($request->input('type'),$this->user()['id']);  
 return $this->rspSuccessData($data);  
 }  
}

<?php  
  
namespace App\Http\Controllers\Lists;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Lists\ListDeviceAbnormalRequest;  
use App\Http\Resources\Lists\ListDeviceAbnormalFirstPageResource;  
use App\Http\Resources\Lists\ListDeviceAbnormalResource;  
use App\Services\Lists\ListDeviceAbnormalService;  
use Illuminate\Http\Response;  
  
/\*\*  
 \* 异常指标控制器  
 \*/  
class ListDeviceAbnormalController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(ListDeviceAbnormalService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return Response  
 \*/  
 public function index(ListDeviceAbnormalRequest $request)  
 {  
 $paginator = $this->service->getPaginator($this->user()['id']);  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => ListDeviceAbnormalResource::collection($paginator),  
 ]);  
 }  
  
 public function first\_page(ListDeviceAbnormalRequest $request)  
 {  
 $list = $this->service->getByNum(12,$this->user()['id']);  
 return $this->rspSuccessData(ListDeviceAbnormalFirstPageResource::collection($list));  
 }  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/12  
 \* Time 4:39 下午  
 \*/  
  
namespace App\Http\Controllers\Open;  
  
use App\Exceptions\ApiException;  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Open\DeviceRequest;  
use App\Http\Resources\Device\DeviceRecordResource;  
use App\Http\Resources\Open\GeneralRecordResource;  
use App\Services\Device\DeviceGeneralService;  
use App\Services\Device\DeviceRecordService;  
use App\Services\Tool\OpenToolService;  
use Illuminate\Contracts\Foundation\Application;  
use Illuminate\Contracts\Routing\ResponseFactory;  
use Illuminate\Http\Response;  
use Illuminate\Support\Facades\File;  
  
/\*\*  
 \* 设备开放接口，接收设备测量数据  
 \*/  
class DeviceController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(DeviceRecordService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* 体检机  
 \*  
 \* @param DeviceRequest $request  
 \* @return Application|ResponseFactory|Response  
 \* @throws ApiException  
 \*/  
 public function general(DeviceRequest $request)  
 {  
 $inputs = $request->only(app(DeviceGeneralService::class)->postFields());  
 $documentId = OpenToolService::parseDeviceBarCode($request->bar\_code);  
 $params = array\_merge($inputs,['document\_id' => $documentId]);  
 $res = $this->service->createFromDeviceGeneral($documentId,$params);  
 return $this->rspSuccessData(new DeviceRecordResource($res));  
 }  
  
 public function general\_record(DeviceRequest $request)  
 {  
 $paginator = $this->service->getDeviceGeneralPaginator();  
 return $this->rspSuccessData([  
 'total' => $paginator->total(),  
 'data' => GeneralRecordResource::collection($paginator),  
 ]);  
 }  
  
 /\*\*  
 \* 心电仪  
 \*  
 \* @param DeviceRequest $request  
 \* @return array  
 \* @throws ApiException  
 \*/  
 public function heart\_rate(DeviceRequest $request)  
 {  
 $this->service->createFromDeviceHeartRate($request->all());  
 return [  
 'code' => 0,  
 'message' => '处理成功',  
 ];  
 }  
  
 /\*\*  
 \* 尿液分析仪  
 \*  
 \* @param DeviceRequest $request  
 \* @return Application|ResponseFactory|Response  
 \* @throws ApiException  
 \*/  
 public function urine(DeviceRequest $request)  
 {  
 $this->service->createFromDeviceUrine($request->all());  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* 尿液分析仪上传记录查询  
 \*  
 \* @param DeviceRequest $request  
 \* @return Application|ResponseFactory|Response  
 \*/  
 public function urine\_query(DeviceRequest $request)  
 {  
 $res = $this->service->queryDeviceUrine($request->input('test\_record\_ids',[]));  
 return $this->rspSuccessData($res);  
 }  
  
 /\*\*  
 \* 康尚一体机  
 \*  
 \* @param DeviceRequest $request  
 \* @return Application|ResponseFactory|Response  
 \* @throws ApiException  
 \*/  
 public function ks(DeviceRequest $request)  
 {  
 $this->service->createFromDeviceKs($request->all());  
 return $this->rspSuccessData();  
 }  
}

<?php  
  
namespace App\Http\Controllers\Rbac;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Rbac\MenuRequest;  
use App\Http\Resources\Rbac\MenuResource;  
use App\Models\Rbac\MenuModel;  
use App\Services\Rbac\MenuService;  
  
class MenuController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(MenuService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function index(MenuRequest $request)  
 {  
 $role = $this->service->indexMenu();  
 return $this->rspSuccessData($role);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function store(MenuRequest $request)  
 {  
 $params = $request->only($this->service->postFields());  
 $this->service->createMenu($params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function show(MenuRequest $request, $id)  
 {  
 $menu = MenuModel::query()->findOrFail($id);  
 return $this->rspSuccessData(new MenuResource($menu));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function update(MenuRequest $request, $id)  
 {  
 $params = $request->only($this->service->patchFields());  
 $this->service->updateMenu($id, $params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function destroy(MenuRequest $request, $id)  
 {  
 MenuModel::destroy($id);  
 return $this->rspSuccessData();  
 }  
  
}

<?php  
  
namespace App\Http\Controllers\Rbac;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Rbac\PermissionRequest;  
use App\Http\Resources\Rbac\PermissionResource;  
use App\Models\Rbac\Permission;  
use App\Services\Rbac\PermissionService;  
  
class PermissionController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(PermissionService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function index(PermissionRequest $request)  
 {  
 $permission = $this->service->indexPermission();  
 return $this->rspSuccessData([  
 'data' => PermissionResource::collection($permission),  
 'total' => $permission->total()  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function store(PermissionRequest $request)  
 {  
 $params = $request->only($this->service->postFields());  
 $this->service->createPermission($params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function show(PermissionRequest $request, $id)  
 {  
 $permission = Permission::query()->findOrFail($id);  
 return $this->rspSuccessData(new PermissionResource($permission));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function update(PermissionRequest $request, $id)  
 {  
 $params = $request->only($this->service->postFields());  
 $this->service->updatePermission($id, $params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function destroy(PermissionRequest $request, $id)  
 {  
 Permission::destroy($id);  
 return $this->rspSuccessData();  
 }  
  
}

<?php  
  
namespace App\Http\Controllers\Rbac;  
  
use App\Http\Controllers\Controller;  
use App\Http\Requests\Rbac\RoleRequest;  
use App\Http\Resources\Rbac\RoleResource;  
use App\Models\Rbac\Role;  
use App\Services\Rbac\RoleService;  
  
class RoleController extends Controller  
{  
 protected $service;  
  
 public function \_\_construct(RoleService $service)  
 {  
 $this->service = $service;  
 }  
  
 /\*\*  
 \* Display a listing of the resource.  
 \*  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function index(RoleRequest $request)  
 {  
 $role = $this->service->indexRole();  
 return $this->rspSuccessData([  
 'data' => RoleResource::collection($role),  
 'total' => $role->total()  
 ]);  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function store(RoleRequest $request)  
 {  
 $params = $request->only($this->service->postFields());  
 $this->service->createRole($params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function show(RoleRequest $request, $id)  
 {  
 $role = Role::query()->findOrFail($id);  
 return $this->rspSuccessData(new RoleResource($role));  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*  
 \* @param \Illuminate\Http\Request $request  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function update(RoleRequest $request, $id)  
 {  
 $params = $request->only($this->service->patchFields());  
 $this->service->updateRole($id, $params);  
 return $this->rspSuccessData();  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \* @param int $id  
 \* @return \Illuminate\Http\Response  
 \*/  
 public function destroy(RoleRequest $request, $id)  
 {  
 $this->service->deleteRole($id);  
 return $this->rspSuccessData();  
 }  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/10  
 \* Time 7:36 下午  
 \*/  
  
namespace App\Services\Common;  
  
use App\Models\Area;  
use App\Services\RestService;  
  
class AreaService extends RestService  
{  
  
 public function getOptions($pid=0)  
 {  
 $all = Area::query()  
 ->select(['id','pid','name'])  
 ->orderByDesc('sort\_number')  
 ->get()  
 ->toArray();  
 return $this->toTree($all, $pid);  
 }  
  
 public function toTree(array $items,$pid = 0)  
 {  
 $res = [];  
 foreach ($items as $k => $item){  
 if ($item['pid'] == $pid){  
 $children = $this->toTree($items,$item['id']);  
 if (!empty($children)){  
 $item['children'] = $children;  
 }  
 unset($items[$k]);  
 $res[] = $item;  
 }  
 }  
 return $res;  
 }  
  
 public function postFields(): array  
 {  
 return [  
 'name','pid','sort\_number',  
 ];  
 }  
  
 public function createOrUpdateArea(array $params)  
 {  
 if (!empty($params['id'])){  
 $area = Area::query()->findOrFail($params['id']);  
 }else{  
 $area = new Area();  
 }  
 $area = $this->autoFill($area,$params);  
 $area->saveOrFail();  
 return $area;  
 }  
  
 public function getAreaIdsByArea(Area $area, &$areaIds = [])  
 {  
 $areaIds[] = $area->id;  
 foreach ($area->children as $child){  
 $this->getAreaIdsByArea($child, $areaIds);  
 }  
 return $areaIds;  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/11  
 \* Time 9:13 上午  
 \*/  
  
namespace App\Services\Common;  
  
use App\Enums\CommonEnum;  
use App\Exceptions\ApiException;  
use App\Http\Filters\DoctorFilter;  
use App\Models\Admin;  
use App\Models\Doctor;  
use App\Services\AdminService;  
use App\Services\RestService;  
use Illuminate\Database\Eloquent\Builder;  
use Illuminate\Support\Facades\DB;  
use Throwable;  
  
class DoctorService extends RestService  
{  
  
 /\*\*  
 \* 创建or更新医生信息  
 \*  
 \* @param array $params  
 \* @return mixed  
 \* @throws ApiException  
 \*/  
 public function createOrUpdateDoctor(array $params)  
 {  
 try {  
 return DB::transaction(function ()use ($params){  
 if (!empty($params['id'])){  
 $doctor = Doctor::findOrFail($params['id']);  
 }else{  
 $doctor = new Doctor();  
 }  
 $doctor = $this->autoFill($doctor,$params,['admin\_id']);  
 $doctor->saveOrFail();  
 //重新关联账号信息  
 if (!empty($params['admin\_id'])){  
 $admin = Admin::findOrFail($params['admin\_id']);  
 $admin->doctor\_id = $doctor->id;  
 $admin->saveOrFail();  
 }  
 return $doctor;  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function deleteDoctor(int $id)  
 {  
 try {  
 DB::transaction(function ()use ($id){  
 //todo 检查doctor关联关系，需级联删除或者拒绝请求  
 //todo 删除doctor  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function getPaginator($adminId)  
 {  
 $service = app(AdminService::class);  
 $areaIds = $service->getChildrenAreaIds($adminId,request('area\_id',0));  
 if (empty($areaIds)){  
 throw new ApiException('请先关联医生');  
 }  
 $filter = app(DoctorFilter::class);  
 return Doctor::query()  
 ->whereHas('hospital',function (Builder $builder)use ($areaIds){  
 $builder->whereIn('area\_id',$areaIds);  
 })  
 ->filter($filter)  
 ->with(['hospital','admin'])  
 ->orderByDesc('doctors.id')  
 ->paginate();  
 }  
  
 public function postFields(): array  
 {  
 return [  
 'name','hospital\_id','admin\_id','phone',  
 ];  
 }  
  
 public function getOptions()  
 {  
 $filter = app(DoctorFilter::class);  
 return Doctor::query()  
 ->with(['hospital'])  
 ->filter($filter)  
 ->where('status',CommonEnum::ACCOUNT\_STATUS\_ON)  
 ->get();  
 }  
  
 public function getDoctorDetail($id)  
 {  
 return Doctor::query()->findOrFail($id);  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/10  
 \* Time 5:57 下午  
 \*/  
  
namespace App\Services\Common;  
  
use App\Http\Filters\HospitalFilter;  
use App\Models\Admin;  
use App\Models\Area;  
use App\Models\Hospital;  
use App\Services\RestService;  
  
class HospitalService extends RestService  
{  
  
 public function getPaginator($adminId)  
 {  
 $admin = Admin::query()->findOrFail($adminId);  
 return $this->getHospitals($admin->hospital\_id, $admin->area\_id, true);  
 }  
  
 public function createOrUpdateHospital(array $params)  
 {  
 if (!empty($params['id'])) {  
 $hospital = Hospital::findOrFail($params['id']);  
 } else {  
 $hospital = new Hospital();  
 }  
 $hospital = $this->autoFill($hospital,$params);  
 $hospital->saveOrFail();  
 return $hospital;  
 }  
  
 public function postFields(): array  
 {  
 return [  
 'name','area\_id','contact\_name','contact\_phone','type',  
 'longitude','latitude','imgs',  
 ];  
 }  
  
 public function getOptions($keyword)  
 {  
 return Hospital::query()  
 ->leftJoin('areas','areas.id','=','hospitals.area\_id')  
 ->where('hospitals.name','like','%'.$keyword.'%')  
 ->select(['hospitals.id','hospitals.name','areas.name as area\_name'])  
 ->get();  
 }  
  
 public function deleteHospital($id)  
 {  
 //todo  
 }  
  
 /\*\*  
 \* @param $hospitalId 机构id  
 \* @param $areaId 区域id  
 \* @param bool $isPage 是否分页  
 \* @param array $cloumn 查询字段  
 \* @return \Illuminate\Database\Eloquent\Builder[]|\Illuminate\Database\Eloquent\Collection  
 \*/  
 public function getHospitals($hospitalId, $areaId, $isPage = false, $cloumn = [])  
 {  
 $query = Hospital::query()->filter(app(HospitalFilter::class));  
 !empty($cloumn) && $query->select($cloumn);  
  
 if (!empty($hospitalId)){  
 $query = $query->where('id', $hospitalId);  
 }else{  
 //取区域下的所有机构  
 $service = app(AreaService::class);  
 $areaIds = $service->getAreaIdsByArea(Area::query()->findOrFail($areaId));  
 $query = $query->whereIn('area\_id',$areaIds);  
 }  
 if ($isPage){  
 return $query->paginate();  
 }else{  
 return $query->get();  
 }  
 }  
  
  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/17  
 \* Time 9:50 上午  
 \*/  
  
namespace App\Services\Common;  
  
use App\Exceptions\ApiException;  
use App\Http\Filters\StandardFilter;  
use App\Models\Standard;  
use App\Services\RestService;  
  
class StandardService extends RestService  
{  
 public function findByKeyAndSex($key,$sex)  
 {  
 return Standard::query()  
 ->where('key',$key)  
 ->where('sex',$sex)  
 ->first();  
 }  
  
 public function getCategoryOptions()  
 {  
 return Standard::query()  
 ->select('category')  
 ->distinct()  
 ->get()  
 ->pluck('category');  
 }  
  
 public function getPaginator()  
 {  
 $filter = app(StandardFilter::class);  
 return Standard::query()  
 ->select(['key'])  
 ->with('children')  
 ->filter($filter)  
 ->groupBy('key')  
 ->paginate();  
 }  
  
 public function postFields(): array  
 {  
 return [  
 'name','category','unit','min','min\_desc','max','max\_desc',  
 ];  
 }  
  
 public function updateStandard(array $params)  
 {  
 if (!$params['id']){  
 throw new ApiException('id有误');  
 }  
 $standard = Standard::query()->findOrFail($params['id']);  
 $standard = $this->autoFill($standard,$params);  
 $standard->saveOrFail();  
 return $standard;  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/12  
 \* Time 4:52 下午  
 \*/  
  
namespace App\Services\Device;  
  
use App\Models\Device\DeviceAbnormal;  
  
class DeviceAbnormalService  
{  
 public function getIndexAbnormals()  
 {  
 return DeviceAbnormal::query()  
 ->select(['document\_id'])  
 ->with('document.abnormals')  
 ->groupBy('document\_id')  
 ->take(12)  
 ->get();  
 }  
  
 public function getLatest($documentId)  
 {  
 return DeviceAbnormal::query()  
 ->where('document\_id',$documentId)  
 ->orderByDesc('id')  
 ->first();  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/13  
 \* Time 9:49 上午  
 \*/  
  
namespace App\Services\Device;  
  
use App\Enums\CommonEnum;  
use App\Enums\StandardEnum;  
use App\Events\AbnormalChanged;  
use App\Exceptions\ApiException;  
use App\Models\Device\DeviceGeneral;  
use App\Models\Device\DeviceHeartB;  
use App\Models\Device\DeviceHeartRate;  
use App\Models\Device\DeviceKs;  
use App\Models\Device\DeviceRecord;  
use App\Models\Device\DeviceUrine;  
use App\Models\Device\DeviceUrineRoutine;  
use App\Models\Document\Document;  
use App\Services\Common\StandardService;  
use App\Services\RestService;  
use App\Services\Tool\OpenToolService;  
use Illuminate\Database\Eloquent\Builder;  
use Illuminate\Database\Eloquent\Collection;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Database\Eloquent\Relations\MorphMany;  
use Illuminate\Support\Facades\DB;  
use Illuminate\Support\Facades\Log;  
use Illuminate\Support\Facades\Storage;  
use Throwable;  
  
class DeviceRecordService extends RestService  
{  
 protected $standardService;  
  
 public function \_\_construct(StandardService $service)  
 {  
 $this->standardService = $service;  
 }  
  
 public function getDeviceRecordsByDocumentId($documentId)  
 {  
 return DeviceRecord::query()  
 ->where('document\_id',$documentId)  
 ->with([  
 'device\_general','device\_heart\_b','device\_liver\_kidney',  
 'device\_blood\_routine','device\_urine\_routine',  
 ])  
 ->orderByDesc('device\_records.id')  
 ->paginate();  
 }  
  
 /\*\*  
 \* 删除单项检查表  
 \*  
 \* @param int $id  
 \* @throws ApiException  
 \*/  
 public function deleteDeviceRecord(int $id)  
 {  
 try {  
 DB::transaction(function ()use ($id){  
 $record = DeviceRecord::query()->findOrFail($id);  
  
 if ($record->device\_general){  
 $record->device\_general->abnormals()->delete();  
 $record->device\_general()->delete();  
 }  
 if ($record->device\_heart\_b){  
 $record->device\_heart\_b->abnormals()->delete();  
 $record->device\_heart\_b()->delete();  
 }  
 if ($record->device\_liver\_kidney){  
 $record->device\_liver\_kidney->abnormals()->delete();  
 $record->device\_liver\_kidney()->delete();  
 }  
 if ($record->device\_blood\_routine){  
 $record->device\_blood\_routine->abnormals()->delete();  
 $record->device\_blood\_routine()->delete();  
 }  
 if ($record->device\_urine\_routine){  
 $record->device\_urine\_routine->abnormals()->delete();  
 $record->device\_urine\_routine()->delete();  
 }  
  
 event(new AbnormalChanged($record->document\_id));  
  
 $record->delete();  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function getByIdAndDocumentId($id,$documentId)  
 {  
 return DeviceRecord::query()  
 ->where('document\_id',$documentId)  
 ->find($id);  
 }  
  
 public function postFields(): array  
 {  
 return [  
 'measured\_at','is\_completed','document\_id','admin\_id',  
 'general','heart\_b','liver\_kidney','blood\_routine','urine\_routine',  
 ];  
 }  
  
 /\*\*  
 \* 手动填写单项检查表  
 \*  
 \* @param array $params  
 \* @return mixed  
 \* @throws ApiException  
 \*/  
 public function createOrUpdateDeviceRecord(array $params)  
 {  
 try {  
 return DB::transaction(function ()use ($params){  
 //record  
 if (!empty($params['id'])){  
 $record = DeviceRecord::query()->findOrFail($params['id']);  
 }else{  
 $record = new DeviceRecord();  
 }  
 $params = $this->mergeDefaults([  
 'measured\_at' => now(),  
 ],$params);  
 $record = $this->autoFill($record,$params,[  
 'general','heart\_b','liver\_kidney','blood\_routine','urine\_routine'  
 ]);  
 $record->saveOrFail();  
 $document = $record->document;  
 //一般情况  
 if (!empty($params['general'])){  
 $this->handleDeviceGeneral($record,$params['general'],$document->sex);  
 }  
 //心电B超  
 if (!empty($params['heart\_b'])){  
 $this->handleDeviceHeartB($record,$params['heart\_b'],$document->sex);  
 }  
 //肝肾功能  
 if (!empty($params['liver\_kidney'])){  
 $this->handleDeviceLiverKidney($record,$params['liver\_kidney'],$document->sex);  
 }  
 //血常规  
 if (!empty($params['blood\_routine'])){  
 $this->handleDeviceBloodRoutine($record,$params['blood\_routine'],$document->sex);  
 }  
 //尿常规  
 if (!empty($params['urine\_routine'])){  
 $this->handleDeviceUrineRoutine($record,$params['urine\_routine'],$document->sex);  
 }  
 event(new AbnormalChanged($record->document\_id));  
 return $record;  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 /\*\*  
 \* 更新单项检查表  
 \*  
 \* @param array $params  
 \* @return mixed  
 \* @throws ApiException  
 \*/  
 public function updateDeviceRecord(array $params)  
 {  
 try {  
 return DB::transaction(function ()use ($params){  
 //record  
 $record = DeviceRecord::query()->findOrFail($params['id']);  
 $record = $this->autoFill($record,$params,[  
 'general','heart\_b','liver\_kidney','blood\_routine','urine\_routine'  
 ],false);  
 $record->updated\_at = now();  
 $record->saveOrFail();  
 $document = $record->document;  
 //一般情况  
 if (!empty($params['general'])){  
 $this->handleDeviceGeneral($record,$params['general'],$document->sex);  
 }  
 //心电B超  
 if (!empty($params['heart\_b'])){  
 $this->handleDeviceHeartB($record,$params['heart\_b'],$document->sex);  
 }  
 //肝肾功能  
 if (!empty($params['liver\_kidney'])){  
 $this->handleDeviceLiverKidney($record,$params['liver\_kidney'],$document->sex);  
 }  
 //血常规  
 if (!empty($params['blood\_routine'])){  
 $this->handleDeviceBloodRoutine($record,$params['blood\_routine'],$document->sex);  
 }  
 //尿常规  
 if (!empty($params['urine\_routine'])){  
 $this->handleDeviceUrineRoutine($record,$params['urine\_routine'],$document->sex);  
 }  
 event(new AbnormalChanged($record->document\_id));  
 return $record;  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 /\*\*  
 \* 处理 "一般情况" 数据  
 \*  
 \* @param DeviceRecord $record  
 \* @param array $params  
 \* @param $sex  
 \* @return Model|mixed  
 \* @throws Throwable  
 \*/  
 private function handleDeviceGeneral(DeviceRecord $record,array $params,$sex)  
 {  
 $general = $record->device\_general;  
 if (!$general){  
 //新增  
 $general = $record->device\_general()->create(array\_merge($params,[  
 'document\_id' => $record->document\_id,  
 'admin\_id' => $record->admin\_id,  
 ]));  
 !empty($params['height']) && $general->height = $params['height'];  
 !empty($params['weight']) && $general->weight = $params['weight'];  
 !empty($params['pulse']) && $general->pulse = $params['pulse'];  
 //bmi  
 if (!empty($params['height']) && !empty($params['weight'])){  
 $params['bmi'] = number\_format($params['weight']/ (($params['height'] / 100) \*\* 2),2);  
 if (strpos($params['bmi'],',') !== false){  
 throw new ApiException('bmi异常，请重新输入');  
 }  
 $general->bmi = $params['bmi'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BMI,$sex);  
 if ($standard){  
 if ($params['bmi'] < $standard->min){  
 $general->bmi\_judge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BMI,$standard->min\_desc);  
 }elseif ($params['bmi'] > $standard->max){  
 $general->bmi\_judge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BMI,$standard->max\_desc);  
 }else{  
 $general->bmi\_judge = CommonEnum::JUDGE\_NORMAL;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BMI,false);  
 }  
 }  
 }  
 //体温  
 if (!empty($params['temperature'])){  
 $general->temperature = $params['temperature'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_TEPERATURE,$sex);  
 if ($standard){  
 if ($params['temperature'] < $standard->min){  
 $general->temperature\_judge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_TEMPERATURE,$standard->min\_desc);  
 }elseif ($params['temperature'] > $standard->max){  
 $general->temperature\_judge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_TEMPERATURE,$standard->max\_desc);  
 }else{  
 $general->temperature\_judge = CommonEnum::JUDGE\_NORMAL;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_TEMPERATURE,false);  
 }  
 }  
 }  
 //血压  
 $bloodPressureJudge = CommonEnum::JUDGE\_NORMAL;  
 $hasBloodPressure = false;  
 if (!empty($params['blood\_pressure\_left1'])){  
 $hasBloodPressure = true;  
 $general->blood\_pressure\_left1 = $params['blood\_pressure\_left1'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_LEFT1,$sex);  
 if ($standard){  
 if ($params['blood\_pressure\_left1'] < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($params['blood\_pressure\_left1'] > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if (!empty($params['blood\_pressure\_left2'])){  
 $hasBloodPressure = true;  
 $general->blood\_pressure\_left2 = $params['blood\_pressure\_left2'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_LEFT2,$sex);  
 if ($standard){  
 if ($params['blood\_pressure\_left2'] < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($params['blood\_pressure\_left2'] > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if (!empty($params['blood\_pressure\_right1'])){  
 $hasBloodPressure = true;  
 $general->blood\_pressure\_right1 = $params['blood\_pressure\_right1'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_RIGHT1,$sex);  
 if ($standard){  
 if ($params['blood\_pressure\_right1'] < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($params['blood\_pressure\_right1'] > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if (!empty($params['blood\_pressure\_right2'])){  
 $hasBloodPressure = true;  
 $general->blood\_pressure\_right2 = $params['blood\_pressure\_right2'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_RIGHT2,$sex);  
 if ($standard){  
 if ($params['blood\_pressure\_right2'] < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($params['blood\_pressure\_right2'] > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if ($hasBloodPressure && $bloodPressureJudge === CommonEnum::JUDGE\_NORMAL){  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,false);  
 }  
 if ($hasBloodPressure){  
 $general->blood\_pressure\_judge = $bloodPressureJudge;  
 }  
 //血糖  
 if (!empty($params['blood\_sugar'])){  
 $general->blood\_sugar = $params['blood\_sugar'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_SUGAR,$sex);  
 if ($standard){  
 if ($params['blood\_sugar'] < $standard->min){  
 $general->blood\_sugar\_judge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_SUGAR,$standard->min\_desc);  
 }elseif ($params['blood\_sugar'] > $standard->max){  
 $general->blood\_sugar\_judge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_SUGAR,$standard->max\_desc);  
 }else{  
 $general->blood\_sugar\_judge = CommonEnum::JUDGE\_NORMAL;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_SUGAR,false);  
 }  
 }  
 }  
 $general->saveOrFail();  
 }else{  
 //更新  
 $isBmiNeedUpdate = false;  
 if (array\_key\_exists('height',$params) && $params['height'] != $general->height){  
 $isBmiNeedUpdate = true;  
 $general->height = $params['height'];  
 }  
 if (array\_key\_exists('weight',$params) && $params['weight'] != $general->weight){  
 $isBmiNeedUpdate = true;  
 $general->weight = $params['weight'];  
 }  
 array\_key\_exists('pulse',$params) && $general->pulse = $params['pulse'];  
 //bmi  
 if ($isBmiNeedUpdate){  
 $bmi = empty($general->height) ? 0 : number\_format($general->weight/ (($general->height / 100) \*\* 2),2);  
 if (strpos($bmi,',') !== false){  
 throw new ApiException('bmi异常，请重新输入');  
 }  
 $general->bmi = $bmi;  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BMI,$sex);  
 if ($standard){  
 if (empty($bmi)){  
 $general->bmi\_judge = null;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BMI,false);  
 }elseif($bmi < $standard->min){  
 $general->bmi\_judge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BMI,$standard->min\_desc);  
 }elseif($bmi > $standard->max){  
 $general->bmi\_judge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BMI,$standard->max\_desc);  
 }else{  
 $general->bmi\_judge = CommonEnum::JUDGE\_NORMAL;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BMI,false);  
 }  
 }  
 }  
 //体温  
 if (array\_key\_exists('temperature',$params) && $params['temperature'] != $general->temperature){  
 $general->temperature = $params['temperature'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_TEPERATURE,$sex);  
 if ($standard){  
 if (empty($params['temperature'])){  
 $general->temperature\_judge = null;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_TEMPERATURE,false);  
 }elseif($params['temperature'] < $standard->min){  
 $general->temperature\_judge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_TEMPERATURE,$standard->min\_desc);  
 }elseif($params['temperature'] > $standard->max){  
 $general->temperature\_judge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_TEMPERATURE,$standard->max\_desc);  
 }else{  
 $general->temperature\_judge = CommonEnum::JUDGE\_NORMAL;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_TEMPERATURE,false);  
 }  
 }  
 }  
 //血压  
 $bloodPressureJudge = $general->blood\_pressure\_judge;  
 $isBloodPressureNeedUpdate = false;  
 if (array\_key\_exists('blood\_pressure\_left1',$params) && $params['blood\_pressure\_left1'] != $general->blood\_pressure\_left1){  
 $isBloodPressureNeedUpdate = true;  
 $general->blood\_pressure\_left1 = $params['blood\_pressure\_left1'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_LEFT1,$sex);  
 if ($standard){  
 if (empty($general->blood\_pressure\_left1)){  
 }elseif ($general->blood\_pressure\_left1 < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($general->blood\_pressure\_left1 > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if (array\_key\_exists('blood\_pressure\_left2',$params) && $params['blood\_pressure\_left2'] != $general->blood\_pressure\_left2){  
 $isBloodPressureNeedUpdate = true;  
 $general->blood\_pressure\_left2 = $params['blood\_pressure\_left2'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_LEFT2,$sex);  
 if ($standard){  
 if (empty($general->blood\_pressure\_left2)){  
 }elseif ($general->blood\_pressure\_left2 < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($general->blood\_pressure\_left2 > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if (array\_key\_exists('blood\_pressure\_right1',$params) && $params['blood\_pressure\_right1'] != $general->blood\_pressure\_right1){  
 $isBloodPressureNeedUpdate = true;  
 $general->blood\_pressure\_right1 = $params['blood\_pressure\_right1'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_RIGHT1,$sex);  
 if ($standard){  
 if (empty($general->blood\_pressure\_right1)){  
 }elseif ($general->blood\_pressure\_right1 < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($general->blood\_pressure\_right1 > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if (array\_key\_exists('blood\_pressure\_right2',$params) && $params['blood\_pressure\_right2'] != $general->blood\_pressure\_right2){  
 $isBloodPressureNeedUpdate = true;  
 $general->blood\_pressure\_right2 = $params['blood\_pressure\_right2'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_PRESSURE\_RIGHT2,$sex);  
 if ($standard){  
 if (empty($general->blood\_pressure\_right2)){  
 }elseif ($general->blood\_pressure\_right2 < $standard->min){  
 $bloodPressureJudge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->min\_desc);  
 }elseif ($general->blood\_pressure\_right2 > $standard->max){  
 $bloodPressureJudge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE,$standard->max\_desc);  
 }  
 }  
 }  
 if ($isBloodPressureNeedUpdate){  
 if (empty($general->blood\_pressure\_left1) && empty($general->blood\_pressure\_left2) &&  
 empty($general->blood\_pressure\_right1) && empty($general->blood\_pressure\_right2)) {  
 $general->blood\_pressure\_judge = null;  
 $this->handleDeviceGeneralAbnormal($general, DeviceRecord::TYPE\_GENERAL\_BLOOD\_PRESSURE, false);  
 } else {  
 $general->blood\_pressure\_judge = $bloodPressureJudge;  
 }  
 }  
 //血糖  
 if (array\_key\_exists('blood\_sugar',$params) && $params['blood\_sugar'] != $general->blood\_sugar){  
 $general->blood\_sugar = $params['blood\_sugar'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_BLOOD\_SUGAR,$sex);  
 if ($standard){  
 if (empty($general->blood\_sugar)){  
 $general->blood\_sugar\_judge = null;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_SUGAR,false);  
 }elseif ($general->blood\_sugar < $standard->min){  
 $general->blood\_sugar\_judge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_SUGAR,$standard->min\_desc);  
 }elseif($general->blood\_sugar > $standard->max){  
 $general->blood\_sugar\_judge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_SUGAR,$standard->max\_desc);  
 }else{  
 $general->blood\_sugar\_judge = CommonEnum::JUDGE\_NORMAL;  
 $this->handleDeviceGeneralAbnormal($general,DeviceRecord::TYPE\_GENERAL\_BLOOD\_SUGAR,false);  
 }  
 }  
 }  
 $general->saveOrFail();  
 }  
 return $general;  
 }  
  
 /\*\*  
 \* 处理 "一般情况" 异常  
 \*  
 \* @param DeviceGeneral $deviceGeneral  
 \* @param $type  
 \* @param $title  
 \* @return Model|MorphMany|object|void  
 \* @throws Throwable  
 \*/  
 private function handleDeviceGeneralAbnormal(DeviceGeneral $deviceGeneral,$type,$title)  
 {  
 //异常 => 正常  
 if (false === $title){  
 $deviceGeneral->abnormals()  
 ->where('type',$type)  
 ->delete();  
 return;  
 }  
 $abnormal = $deviceGeneral->abnormals()  
 ->where('type',$type)  
 ->first();  
 //异常 => 异常  
 if ($abnormal){  
 $abnormal->title = $title;  
 $abnormal->saveOrFail();  
 return $abnormal;  
 }  
 //正常 => 异常  
 return $deviceGeneral->abnormals()->create([  
 'document\_id' => $deviceGeneral->document\_id,  
 'admin\_id' => $deviceGeneral->admin\_id,  
 'title' => $title,  
 'type' => $type,  
 ]);  
 }  
  
 /\*\*  
 \* 处理 "心电B超" 数据  
 \*  
 \* @param DeviceRecord $record  
 \* @param array $params  
 \* @param $sex  
 \* @return Model|mixed  
 \* @throws Throwable  
 \*/  
 private function handleDeviceHeartB(DeviceRecord $record,array $params,$sex)  
 {  
 $heartB = $record->device\_heart\_b;  
 if (!$heartB){  
 $heartB = $record->device\_heart\_b()->create([  
 'document\_id' => $record->document\_id,  
 'admin\_id' => $record->admin\_id,  
 ]);  
 }  
 //心率  
 if (array\_key\_exists('heart\_rate',$params) && $params['heart\_rate'] != $heartB->heart\_rate){  
 $heartB->heart\_rate = $params['heart\_rate'];  
 $standard = $this->standardService->findByKeyAndSex(StandardEnum::STD\_HEART\_RATE,$sex);  
 if ($standard){  
 if (is\_null($heartB->heart\_rate)){  
 $heartB->heart\_rate\_judge = null;  
 $this->handleDeviceHeartBAbnormal($heartB,false);  
 }elseif ($heartB->heart\_rate < $standard->min){  
 $heartB->heart\_rate\_judge = CommonEnum::JUDGE\_LOW;  
 $this->handleDeviceHeartBAbnormal($heartB,$standard->min\_desc);  
 }elseif ($heartB->heart\_rate > $standard->max){  
 $heartB->heart\_rate\_judge = CommonEnum::JUDGE\_HIGH;  
 $this->handleDeviceHeartBAbnormal($heartB,$standard->max\_desc);  
 }else{  
 $heartB->heart\_rate\_judge = CommonEnum::JUDGE\_NORMAL;  
 $this->handleDeviceHeartBAbnormal($heartB,false);  
 }  
 }  
 }  
 array\_key\_exists('heart\_rhythm',$params) && $heartB->heart\_rhythm = $params['heart\_rhythm'];  
 array\_key\_exists('murmur',$params) && $heartB->murmur = $params['murmur'];  
 array\_key\_exists('abdomen',$params) && $heartB->abdomen = $params['abdomen'];  
 array\_key\_exists('b\_name',$params) && $heartB->b\_name = $params['b\_name'];  
 array\_key\_exists('b\_judge',$params) && $heartB->b\_judge = $params['b\_judge'];  
 array\_key\_exists('cervical\_smear',$params) && $heartB->cervical\_smear = $params['cervical\_smear'];  
 array\_key\_exists('other',$params) && $heartB->other = $params['other'];  
 $heartB->saveOrFail();  
 return $heartB;  
 }  
  
 /\*\*  
 \* 处理 "心电B超" 异常  
 \*  
 \* @param DeviceHeartB $deviceHeartB  
 \* @param $title  
 \* @return Model|MorphMany|object|void  
 \* @throws Throwable  
 \*/  
 private function handleDeviceHeartBAbnormal(DeviceHeartB $deviceHeartB,$title)  
 {  
 $type = DeviceRecord::TYPE\_HAERT\_B\_HEART\_RATE;  
 if (false === $title){  
 $deviceHeartB->abnormals()  
 ->where('type',$type)  
 ->delete();  
 return;  
 }  
 $abnormal = $deviceHeartB->abnormals()  
 ->where('type',$type)  
 ->first();  
 if ($abnormal){  
 $abnormal->title = $title;  
 $abnormal->saveOrFail();  
 return $abnormal;  
 }  
 return $deviceHeartB->abnormals()->create([  
 'document\_id' => $deviceHeartB->document\_id,  
 'admin\_id' => $deviceHeartB->admin\_id,  
 'title' => $title,  
 'type' => $type,  
 ]);  
 }  
  
 /\*\*  
 \* 处理 "肝肾功能" 数据  
 \*  
 \* @param DeviceRecord $record  
 \* @param array $params  
 \* @param $sex  
 \* @return Model|mixed  
 \* @throws Throwable  
 \*/  
 private function handleDeviceLiverKidney(DeviceRecord $record,array $params,$sex)  
 {  
 $kidney = $record->device\_liver\_kidney;  
 if (!$kidney){  
 $kidney = $record->device\_liver\_kidney()->create([  
 'document\_id' => $record->document\_id,  
 'admin\_id' => $record->admin\_id,  
 ]);  
 }  
 array\_key\_exists('liver\_1',$params) && $kidney->liver\_1 = $params['liver\_1'];  
 array\_key\_exists('liver\_2',$params) && $kidney->liver\_2 = $params['liver\_2'];  
 array\_key\_exists('liver\_3',$params) && $kidney->liver\_3 = $params['liver\_3'];  
 array\_key\_exists('liver\_4',$params) && $kidney->liver\_4 = $params['liver\_4'];  
 array\_key\_exists('liver\_5',$params) && $kidney->liver\_5 = $params['liver\_5'];  
 array\_key\_exists('kidney\_1',$params) && $kidney->kidney\_1 = $params['kidney\_1'];  
 array\_key\_exists('kidney\_2',$params) && $kidney->kidney\_2 = $params['kidney\_2'];  
 array\_key\_exists('kidney\_3',$params) && $kidney->kidney\_3 = $params['kidney\_3'];  
 array\_key\_exists('kidney\_4',$params) && $kidney->kidney\_4 = $params['kidney\_4'];  
 $kidney->saveOrFail();  
 return $kidney;  
 }  
  
 /\*\*  
 \* 处理 "血常规" 数据  
 \*  
 \* @param DeviceRecord $record  
 \* @param array $params  
 \* @param $sex  
 \* @return Model|mixed  
 \* @throws Throwable  
 \*/  
 private function handleDeviceBloodRoutine(DeviceRecord $record,array $params,$sex)  
 {  
 $routine = $record->device\_blood\_routine;  
 if (!$routine){  
 $routine = $record->device\_blood\_routine()->create([  
 'document\_id' => $record->document\_id,  
 'admin\_id' => $record->admin\_id,  
 ]);  
 }  
 array\_key\_exists('blood\_1',$params) && $routine->blood\_1 = $params['blood\_1'];  
 array\_key\_exists('blood\_2',$params) && $routine->blood\_2 = $params['blood\_2'];  
 array\_key\_exists('blood\_3',$params) && $routine->blood\_3 = $params['blood\_3'];  
 array\_key\_exists('blood\_4',$params) && $routine->blood\_4 = $params['blood\_4'];  
 array\_key\_exists('routine\_1',$params) && $routine->routine\_1 = $params['routine\_1'];  
 array\_key\_exists('routine\_2',$params) && $routine->routine\_2 = $params['routine\_2'];  
 array\_key\_exists('routine\_3',$params) && $routine->routine\_3 = $params['routine\_3'];  
 array\_key\_exists('other',$params) && $routine->other = $params['other'];  
 $routine->saveOrFail();  
 return $routine;  
 }  
  
 /\*\*  
 \* 处理 "尿常规" 数据  
 \*  
 \* @param DeviceRecord $record  
 \* @param array $params  
 \* @param $sex  
 \* @return Model|mixed  
 \* @throws Throwable  
 \*/  
 private function handleDeviceUrineRoutine(DeviceRecord $record,array $params,$sex)  
 {  
 $routine = $record->device\_urine\_routine;  
 if (!$routine){  
 $routine = $record->device\_urine\_routine()->create([  
 'document\_id' => $record->document\_id,  
 'admin\_id' => $record->admin\_id,  
 ]);  
 }  
 array\_key\_exists('urine\_1',$params) && $routine->urine\_1 = $params['urine\_1'];  
 array\_key\_exists('urine\_2',$params) && $routine->urine\_2 = $params['urine\_2'];  
 array\_key\_exists('urine\_3',$params) && $routine->urine\_3 = $params['urine\_3'];  
 array\_key\_exists('urine\_5',$params) && $routine->urine\_5 = $params['urine\_5'];  
 array\_key\_exists('urine\_6',$params) && $routine->urine\_6 = $params['urine\_6'];  
 array\_key\_exists('type\_b',$params) && $routine->type\_b = $params['type\_b'];  
 $routine->saveOrFail();  
 return $routine;  
 }  
  
 /\*\*  
 \* 获取单项检测详情  
 \*  
 \* @param int $id  
 \* @return Builder|Builder[]|Collection|Model|null  
 \*/  
 public function getDeviceRecordDetail(int $id)  
 {  
 return DeviceRecord::query()  
 ->with([  
 'device\_general','device\_heart\_b','device\_liver\_kidney',  
 'device\_blood\_routine','device\_urine\_routine',  
 ])  
 ->findOrFail($id);  
 }  
  
 /\*\*  
 \* 获取 $minutes 分钟内的数据  
 \*  
 \* @param $documentId  
 \* @param int $minutes  
 \* @return Builder|Model|object|null  
 \*/  
 private function getOneWithinMinutes($documentId,$minutes = 30)  
 {  
 return DeviceRecord::query()  
 ->where('document\_id',$documentId)  
 ->where('measured\_at','>=',now()->subMinutes($minutes))  
 ->first();  
 }  
  
 /\*\*  
 \* 从 体检机 创建记录  
 \*  
 \* @param $documentId  
 \* @param array $generals  
 \* @return mixed  
 \* @throws ApiException  
 \*/  
 public function createFromDeviceGeneral($documentId,array $generals)  
 {  
 try {  
 return DB::transaction(function ()use ($documentId,$generals){  
 $document = Document::query()->findOrFail($documentId);  
 $record = $this->getOneWithinMinutes($documentId);  
 if (!$record){  
 $record = $document->device\_records()->create([  
 'measured\_at' => now()  
 ]);  
 }  
 $this->handleDeviceGeneral($record,$generals,$document->sex);  
 event(new AbnormalChanged($documentId));  
 return DeviceRecord::query()->with('device\_general')->findOrFail($record->id);  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function createFromDeviceHeartRate(array $params)  
 {  
 $storage = Storage::disk('cosv5');  
 list($reportFilePath,$dataFilePath) = [  
 '/heart-rate/report/'.uniqid().'.jpg',  
 '/heart-rate/data/'.uniqid().'.xml'  
 ];  
 if (false === $storage->put($reportFilePath,base64\_decode($params['reportFile']))){  
 throw new ApiException('心电图片文件上传失败');  
 }  
 if (false === $storage->put($dataFilePath,base64\_decode($params['dataFile']))){  
 throw new ApiException('心电数据文件上传失败');  
 }  
 unset($params['reportFile'],$params['dataFile']);  
 try {  
 $document = Document::query()->where('barcode',$params['patientID'])->firstOrFail();  
 $record = $document->device\_records()->create([  
 'measured\_at' => now(),  
 ]);  
 $params['heart\_rate'] = $params['heartRate'];  
 $heartB = $this->handleDeviceHeartB($record,$params,$document->sex);  
 $heartB->device\_heart\_rate()->create([  
 'document\_id' => $document->id,  
 'heart\_rate' => $params['heartRate'],  
 'report\_file\_path' => $reportFilePath,  
 'data\_file\_path' => $dataFilePath,  
 'extra' => json\_encode($params),  
 ]);  
 $heartB->heart\_rate\_img = $reportFilePath;  
 $heartB->heart\_rate\_result = $params['reportResult'];  
 $heartB->saveOrFail();  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 /\*\*  
 \* 尿液分析仪导入数据  
 \*  
 \* @param array $params  
 \* @throws ApiException  
 \*/  
 public function createFromDeviceUrine(array $params)  
 {  
 try {  
 if (DeviceUrine::query()->where('test\_record\_id',$params['ID'])->exists()){  
 throw new \Exception('该记录已上传过了');  
 }  
 $document = Document::query()->where('barcode',$params['Barcode'] ?? $params['PatientName'])->first();  
 if (!$document){  
 throw new \Exception('该记录未找到对应档案:'.$params['ID']);  
 }  
 $record = $document->device\_records()->create([  
 'measured\_at' => $params['TestDate'],  
 ]);  
 $detail = [];  
 foreach ($params['TestRecordDetails'] as $item){  
 if ($item['TestItemCode'] === 'GLU'){  
 $detail['urine\_1'] = $item['TestResult'];  
 }  
 if ($item['TestItemCode'] === 'KET'){  
 $detail['urine\_2'] = $item['TestResult'];  
 }  
 if ($item['TestItemCode'] === 'BLO'){  
 $detail['urine\_3'] = $item['TestResult'];  
 }  
 if ($item['TestItemCode'] === 'ALB'){  
 $detail['urine\_5'] = $item['TestResult'];  
 }  
 }  
 $urineRoutine = $this->handleDeviceUrineRoutine($record,$detail,$document->sex);  
 $urineRoutine->device\_urine()->create([  
 'document\_id' => $document->id,  
 'test\_record\_id' => $params['ID'],  
 'result' => $params['TestRecordDetails'],  
 ]);  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 /\*\*  
 \* 康尚一体机导入数据  
 \*  
 \* @param array $params  
 \* @throws ApiException  
 \*/  
 public function createFromDeviceKs(array $params)  
 {  
 try {  
 DB::transaction(function ()use ($params){  
 $rsId = config('scs.device.ks.rsId');  
 if (!isset($params['key']) || $params['key'] !== md5($rsId.'konsungyitijijsondata')){  
 throw new ApiException('key验证失败');  
 }  
 $document = Document::query()->where('card\_number',$params['personInfo']['idNumber'])->firstOrFail();  
 $deviceKs = DeviceKs::query()->where('data\_id',$params['dataId'])->first();  
 if (!$deviceKs){  
 $deviceKs = $document->device\_ks()->create([  
 'data\_id' => $params['dataId'],  
 'device\_code' => $params['deviceCode'],  
 'id\_card' => $params['personInfo']['idNumber'],  
 ]);  
 }  
 $record = $deviceKs->device\_record;  
 if (!$record){  
 $record = $deviceKs->device\_record()->create([  
 'measured\_at' => $params['checkDate'],  
 'document\_id' => $document->id,  
 ]);  
 $deviceKs->device\_record\_id = $record->id;  
 }  
 $data = $this->formatKsCheckData($params['checkData']);  
 if (!empty($data['general'])){  
 $general = $this->handleDeviceGeneral($record,$data['general'],$document->sex);  
 $deviceKs->device\_general\_id = $general->id;  
 }  
 if (!empty($data['heart\_b'])){  
 $heartB = $this->handleDeviceHeartB($record,$data['heart\_b'],$document->sex);  
 $deviceKs->device\_heart\_b\_id = $heartB->id;  
 //保存心电图  
 $query = [  
 'dataId' => substr($data['heart\_b']['file\_path'],strpos($data['heart\_b']['file\_path'],'=')+1),  
 'gs' => '6 2 1R',  
 'dl' => '',  
 'zy' => '10',  
 'bxzs' => '25',  
 'imageType' => '1',  
 ];  
 $url = 'http://'.config('scs.device.ks.ip').':'.config('scs.device.ks.port')  
 .'/gpgp-web/outpatients/downHeartImage?'.http\_build\_query($query);  
 $path = '/device/ks/heart\_rate/';  
 $name = uniqid().'.jpg';  
 $isSuccess = Storage::disk('cosv5')->put($path.$name,file\_get\_contents($url));  
 if (false === $isSuccess){  
 throw new ApiException('上传失败');  
 }  
 $heartB->heart\_rate\_img = $path.$name;  
 $heartB->heart\_rate\_result = $data['heart\_b']['check\_result'];  
 $heartB->saveOrFail();  
 }  
 if (!empty($data['liver\_kidney'])){  
 $liverKidney = $this->handleDeviceLiverKidney($record,$data['liver\_kidney'],$document->sex);  
 $deviceKs->device\_liver\_kidney\_id = $liverKidney->id;  
 }  
 if (!empty($data['blood\_routine'])){  
 $bloodRoutine = $this->handleDeviceBloodRoutine($record,$data['blood\_routine'],$document->sex);  
 $deviceKs->device\_blood\_routine\_id = $bloodRoutine->id;  
 }  
 if (!empty($data['urine\_routine'])){  
 $urineRoutine = $this->handleDeviceUrineRoutine($record,$data['urine\_routine'],$document->sex);  
 $deviceKs->device\_urine\_routine\_id = $urineRoutine->id;  
 }  
 $params['checkData'] = array\_values($data['origin']);  
 $deviceKs->detail = $params;  
 $deviceKs->saveOrFail();  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 private function formatKsCheckData(array $checkData)  
 {  
 $general = $heartB = $liverKidney = $bloodRoutine = $urineRoutine = [];  
 foreach ($checkData as $i => $item){  
 switch ($item['checkKindCode']){  
 case '0'://血压  
 if ($item['checkTypeCode'] === 'sbp'){  
 $general['blood\_pressure\_left2'] = $item['value'];  
 $general['blood\_pressure\_right2'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'dbp'){  
 $general['blood\_pressure\_left1'] = $item['value'];  
 $general['blood\_pressure\_right1'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'blood\_pr'){  
 $general['pulse'] = $item['value'];  
 }  
 break;  
 case '1'://体温  
 if ($item['checkTypeCode'] === 'temperature'){  
 $general['temperature'] = $item['value'];  
 }  
 break;  
 case '2'://血氧  
 break;  
 case '3'://血糖  
 if ($item['checkTypeCode'] === 'sugar'){  
 $general['blood\_sugar'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'sugar\_pbg'){  
 $general['blood\_sugar'] = $item['value'];  
 }  
 break;  
 case '4'://尿常规  
 if ($item['checkTypeCode'] === 'urine\_glu'){  
 $urineRoutine['urine\_1'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'urine\_ket'){  
 $urineRoutine['urine\_2'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'urine\_bld'){  
 $urineRoutine['urine\_3'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'urine\_ma'){  
 $urineRoutine['urine\_5'] = $item['value'];  
 }  
 break;  
 case '5'://血脂  
 if ($item['checkTypeCode'] === 'fat\_flipidschol'){  
 $bloodRoutine['blood\_1'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'fat\_flipidstrig'){  
 $bloodRoutine['blood\_2'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'fat\_flipidsldl'){  
 $bloodRoutine['blood\_3'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'fat\_flipidshdl') {  
 $bloodRoutine['blood\_4'] = $item['value'];  
 }  
 break;  
 case '6'://心电  
 if ($item['checkTypeCode'] === 'ecg\_hr'){  
 $heartB['heart\_rate'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'checkResult'){  
 $heartB['check\_result'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'file\_path'){  
 $heartB['file\_path'] = $item['value'];  
 }else{  
 unset($checkData[$i]);  
 }  
 break;  
 case '7'://炎症  
 break;  
 case '8'://心肌  
 break;  
 case '9'://尿酸  
 break;  
 case '10'://白细胞  
 if ($item['checkTypeCode'] === 'wbc'){  
 $bloodRoutine['routine\_2'] = $item['value'];  
 }  
 break;  
 case '11'://血红蛋白  
 if ($item['checkTypeCode'] === 'assxhdb\_man'){  
 $bloodRoutine['routine\_1'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'assxhdb\_woman'){  
 $bloodRoutine['routine\_1'] = $item['value'];  
 }  
 break;  
 case '12'://糖化血红蛋白  
 if ($item['checkTypeCode'] === 'ghb\_ngsp'){  
 $urineRoutine['routine\_6'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'ghb\_ifcc'){  
 $urineRoutine['routine\_6'] = $item['value'];  
 }elseif ($item['checkTypeCode'] === 'ghb\_eag'){  
 $urineRoutine['routine\_6'] = $item['value'];  
 }  
 break;  
 case '13'://身高  
 if ($item['checkTypeCode'] === 'height'){  
 $general['height'] = $item['value'];  
 }  
 break;  
 case '14'://体重  
 if ($item['checkTypeCode'] === 'weight'){  
 $general['weight'] = $item['value'];  
 }  
 break;  
 case '15'://总胆固醇  
 break;  
 case '16'://肺功能  
 break;  
 case '17'://听诊器  
 break;  
 }  
 }  
 return [  
 'general' => $general,  
 'heart\_b' => $heartB,  
 'liver\_kidney' => $liverKidney,  
 'blood\_routine' => $bloodRoutine,  
 'urine\_routine' => $urineRoutine,  
 'origin' => $checkData,  
 ];  
 }  
  
 public function patchFields(): array  
 {  
 return [  
 'measured\_at','is\_completed',  
 'general','heart\_b','liver\_kidney','blood\_routine','urine\_routine',  
 ];  
 }  
  
 public function getExaminationInfo($documentId)  
 {  
 $document = Document::query()->findOrFail($documentId);  
 $general = $document->device\_generals()->orderByDesc('id')->first();  
 $heartB = $document->device\_heart\_bs()->orderByDesc('id')->first();  
 $liverKidney = $document->device\_liver\_kidneys()->orderByDesc('id')->first();  
 $bloodRoutine = $document->device\_blood\_routines()->orderByDesc('id')->first();  
 $urineRoutine = $document->device\_urine\_routines()->orderByDesc('id')->first();  
 $res = [];  
 if ($general){  
 $res['general'] = [  
 'temperature' => $general->temperature,  
 'pulse' => $general->pulse,  
 'height' => $general->height,  
 'weight' => $general->weight,  
 'bmi' => $general->bmi,  
 'blood\_pressure\_left1' => $general->blood\_pressure\_left1,  
 'blood\_pressure\_left2' => $general->blood\_pressure\_left2,  
 'blood\_pressure\_right1' => $general->blood\_pressure\_right1,  
 'blood\_pressure\_right2' => $general->blood\_pressure\_right2,  
 ];  
 }  
 if ($heartB){  
 $res['body'] = [  
 'heart\_rate' => $heartB->heart\_rate,  
 'heart\_rate\_judge' => $heartB->heart\_rhythm,  
 'heart\_rate\_murmur' => $heartB->murmur,  
 'heart\_rate\_img' => $heartB->heart\_rate\_img,  
 'heart\_rate\_img\_full' => OpenToolService::getObjectUrl($heartB->heart\_rate\_img),  
 ];  
 }  
 if ($liverKidney){  
 $res['liver\_kidney'] = [  
 'liver\_1' => $liverKidney->liver\_1,  
 'liver\_2' => $liverKidney->liver\_2,  
 'liver\_3' => $liverKidney->liver\_3,  
 'liver\_4' => $liverKidney->liver\_4,  
 'liver\_5' => $liverKidney->liver\_5,  
 'kidney\_1' => $liverKidney->kidney\_1,  
 'kidney\_2' => $liverKidney->kidney\_2,  
 'kidney\_3' => $liverKidney->kidney\_3,  
 'kidney\_4' => $liverKidney->kidney\_4,  
 ];  
 }  
 if ($bloodRoutine){  
 $res['blood\_routine'] = [  
 'blood\_1' => $bloodRoutine->blood\_1,  
 'blood\_2' => $bloodRoutine->blood\_2,  
 'blood\_3' => $bloodRoutine->blood\_3,  
 'blood\_4' => $bloodRoutine->blood\_4,  
 'routine\_1' => $bloodRoutine->routine\_1,  
 'routine\_2' => $bloodRoutine->routine\_2,  
 'routine\_3' => $bloodRoutine->routine\_3,  
 ];  
 }  
 if ($urineRoutine){  
 $res['urine\_routine'] = [  
 'urine\_1' => $urineRoutine->urine\_1,  
 'urine\_2' => $urineRoutine->urine\_2,  
 'urine\_3' => $urineRoutine->urine\_3,  
 'urine\_5' => $urineRoutine->urine\_5,  
 'urine\_6' => $urineRoutine->urine\_6,  
 'type\_b' => $urineRoutine->type\_b,  
 ];  
 }  
 return $res;  
 }  
  
 public function queryDeviceUrine(array $testRecordIds)  
 {  
 return DeviceUrine::query()  
 ->select(['test\_record\_id'])  
 ->whereIn('test\_record\_id',$testRecordIds)  
 ->get()  
 ->pluck('test\_record\_id')  
 ->toArray();  
 }  
  
 public function getDeviceGeneralPaginator()  
 {  
 return DeviceGeneral::query()  
 ->with('document')  
 ->orderByDesc('id')  
 ->paginate();  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/11  
 \* Time 3:29 下午  
 \*/  
  
namespace App\Services\Document;  
  
use App\Enums\CommonEnum;  
use App\Exceptions\ApiException;  
use App\Http\Filters\DocumentFilter;  
use App\Models\Document\Document;  
use App\Services\AdminService;  
use App\Services\RestService;  
use Illuminate\Database\Eloquent\Builder;  
use Illuminate\Support\Facades\DB;  
use Throwable;  
  
class DocumentService extends RestService  
{  
 public function createOrUpdateDocument(array $params)  
 {  
 try {  
 return DB::transaction(function ()use ($params){  
 if (!empty($params['id'])){  
 $document = Document::query()->findOrFail($params['id']);  
 if (empty($document->barcode)){  
 $document->barcode = $this->generateUniqueBarcode();  
 }  
 unset($params['admin\_id']);  
 $document = $this->autoFill($document,$params,[  
 'public\_labels','residence\_labels','disease\_labels'  
 ],true);  
 }else{  
 $document = new Document([  
 'barcode' => $this->generateUniqueBarcode()  
 ]);  
 $document = $this->autoFill($document,$params,[  
 'public\_labels','residence\_labels','disease\_labels'  
 ]);  
 }  
 $document->saveOrFail();  
 //关联标签  
 !empty($params['public\_labels']) && $params['public\_labels'] = array\_unique($params['public\_labels']);  
 if (empty($params['public\_labels'])){  
 $document->public\_labels()->delete();  
 }  
 if (!empty($params['public\_labels']) &&  
 $params['public\_labels'] != $document->public\_labels()->pluck('value')->toArray()) {  
 $document->public\_labels()->delete();  
 $labels = array\_map(function ($item) {  
 if (array\_key\_exists($item, CommonEnum::PUBLIC\_LABEL\_MAP)){  
 return ['value' => $item];  
 }  
 }, $params['public\_labels']);  
 $document->public\_labels()->createMany($labels);  
 }  
 !empty($params['residence\_labels']) && $params['residence\_labels'] = array\_unique($params['residence\_labels']);  
 if (empty($params['residence\_labels'])){  
 $document->residence\_labels()->delete();  
 }  
 if (!empty($params['residence\_labels']) &&  
 $params['residence\_labels'] != $document->residence\_labels()->pluck('value')->toArray()) {  
 $document->residence\_labels()->delete();  
 $labels = array\_map(function ($item) {  
 if (array\_key\_exists($item, CommonEnum::RESIDENCE\_LABEL\_MAP)){  
 return ['value' => $item];  
 }  
 }, $params['residence\_labels']);  
 $document->residence\_labels()->createMany($labels);  
 }  
 !empty($params['disease\_labels']) && $params['disease\_labels'] = array\_unique($params['disease\_labels']);  
 if (empty($params['disease\_labels'])){  
 $document->disease\_labels()->delete();  
 }  
 if (!empty($params['disease\_labels']) &&  
 $params['disease\_labels'] != $document->disease\_labels()->pluck('value')->toArray()){  
 $document->disease\_labels()->delete();  
 $labels = array\_map(function ($item){  
 if (array\_key\_exists($item, CommonEnum::DISEASE\_LABEL\_MAP)){  
 return ['value' => $item];  
 }  
 },$params['disease\_labels']);  
 $document->disease\_labels()->createMany($labels);  
 }  
 return Document::query()->findOrFail($document->id);  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function postFields():array  
 {  
 return [  
 'number','name','sex','birthday','address',  
 'residence\_address','township','village\_committee','old\_number',  
 'create\_organization','create\_person','create\_date','person\_id','admin\_id',  
 'public\_labels','residence\_labels','disease\_labels','area\_id','phone',  
 'id\_card\_img','card\_type','card\_number',  
 ];  
 }  
  
 public function deleteDocument(int $id)  
 {  
 //todo 检测级联关系删除  
 }  
  
 public function getPaginator($adminId)  
 {  
 $filter = app(DocumentFilter::class);  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
 return Document::query()  
 ->whereIn('admin\_id',$adminIds)  
 ->with(['public\_labels', 'person'])  
 ->filter($filter)  
 ->paginate();  
 }  
  
 /\*\*  
 \* 档案 联想搜索建议  
 \*  
 \* @param $keyword  
 \* @return array|\Illuminate\Database\Eloquent\Collection  
 \*/  
 public function getSearchSuggests($keyword,$adminId)  
 {  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
 if ($keyword === '' || is\_null($keyword)){  
 return [];  
 }  
 $condition = '%'.$keyword.'%';  
 return Document::query()  
 ->whereIn('admin\_id',$adminIds)  
 ->where(function (Builder $builder)use ($condition){  
 $builder->where('name','like',$condition)  
 ->orWhere('card\_number','like',$condition);  
 })  
 ->orderByDesc('id')  
 ->select(['id','name','card\_type','card\_number'])  
 ->get();  
 }  
  
 public function getDocumentDetail($id)  
 {  
 return Document::query()  
 ->with(['public\_labels','residence\_labels','disease\_labels'])  
 ->findOrFail($id);  
 }  
  
 /\*\*  
 \* 生产唯一条码  
 \*  
 \* @param int $length  
 \* @return string  
 \* @throws \Exception  
 \*/  
 public function generateUniqueBarcode($length = 6)  
 {  
 $chars = [  
 0,1,2,3,4,5,6,7,8,9,'A','B','C','D','E','F','G','H','I','J','K','L','M','N',  
 'O','P','Q','R','S','T','U','V','W','X','Y','Z',  
 ];  
 $count = count($chars);  
 $str = '';  
 for ($i = 0;$i < $length;$i++){  
 $str .= $chars[random\_int(0,$count-1)];  
 }  
 $exists = Document::query()->where('barcode',$str)->exists();  
 if ($exists){  
 return $this->generateUniqueBarcode($length);  
 }  
 return $str;  
 }  
  
 /\*\*  
 \* 更新档案  
 \*  
 \* @param array $params  
 \* @return mixed  
 \* @throws ApiException  
 \*/  
 public function updateDocument(array $params)  
 {  
 try {  
 return DB::transaction(function ()use ($params){  
 $document = Document::query()->findOrFail($params['id']);  
 if (empty($document->barcode)) {  
 $document->barcode = $this->generateUniqueBarcode();  
 }  
 $document = $this->autoFill($document, $params, [  
 'public\_labels', 'residence\_labels', 'disease\_labels'  
 ], false);  
 $document->saveOrFail();  
 //关联标签  
 if (isset($params['public\_labels'])){  
 $params['public\_labels'] = array\_unique($params['public\_labels']);  
 if (empty($params['public\_labels'])){  
 $document->public\_labels()->delete();  
 }elseif ($params['public\_labels'] != $document->public\_labels()->pluck('value')->toArray()) {  
 $document->public\_labels()->delete();  
 $labels = array\_map(function ($item) {  
 if (array\_key\_exists($item, CommonEnum::PUBLIC\_LABEL\_MAP)){  
 return ['value' => $item];  
 }  
 }, $params['public\_labels']);  
 $document->public\_labels()->createMany($labels);  
 }  
 }  
 if (isset($params['residence\_labels'])){  
 $params['residence\_labels'] = array\_unique($params['residence\_labels']);  
 if (empty($params['residence\_labels'])){  
 $document->residence\_labels()->delete();  
 }elseif ($params['residence\_labels'] != $document->residence\_labels()->pluck('value')->toArray()){  
 $document->residence\_labels()->delete();  
 $labels = array\_map(function ($item) {  
 if (array\_key\_exists($item, CommonEnum::RESIDENCE\_LABEL\_MAP)){  
 return ['value' => $item];  
 }  
 }, $params['residence\_labels']);  
 $document->residence\_labels()->createMany($labels);  
 }  
 }  
 if (isset($params['disease\_labels'])){  
 $params['disease\_labels'] = array\_unique($params['disease\_labels']);  
 if (empty($params['disease\_labels'])){  
 $document->disease\_labels()->delete();  
 }elseif ($params['disease\_labels'] != $document->disease\_labels()->pluck('value')->toArray()){  
 $document->disease\_labels()->delete();  
 $labels = array\_map(function ($item){  
 if (array\_key\_exists($item, CommonEnum::DISEASE\_LABEL\_MAP)){  
 return ['value' => $item];  
 }  
 },$params['disease\_labels']);  
 $document->disease\_labels()->createMany($labels);  
 }  
 }  
 return Document::query()->findOrFail($document->id);  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function patchFields()  
 {  
 return [  
 'number','name','sex','birthday','address',  
 'residence\_address','township','village\_committee','old\_number',  
 'create\_organization','create\_person','create\_date','person\_id',  
 'public\_labels','residence\_labels','disease\_labels','area\_id','phone',  
 'id\_card\_img','card\_type','card\_number',  
 ];  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/12/9  
 \* Time 10:16 上午  
 \*/  
  
namespace App\Services\Examination;  
  
use App\Exceptions\ApiException;  
use App\Models\Document\Document;  
use App\Models\Examination\Examination;  
use App\Models\Examination\ExaminationHospitalization;  
use App\Services\AdminService;  
use App\Services\RestService;  
use App\Services\TemplateReport\TemplateService;  
use App\Services\Tool\OpenToolService;  
use Illuminate\Support\Carbon;  
use Illuminate\Support\Facades\DB;  
use PhpOffice\PhpWord\Settings;  
use Throwable;  
  
class ExaminationService extends RestService  
{  
 public function createExamination($adminId,array $params)  
 {  
 try {  
 return DB::transaction(function ()use ($params,$adminId){  
 $document = Document::query()->findOrFail($params['document\_id']);  
 $this->checkPermission($adminId,$document->id);  
 $examination = $document->examinations()->create([  
 'examination\_date' => now(),  
 ]);  
 //症状/一般状况  
 if (!empty($params['general'])){  
 $this->handleExaminationGeneral($examination,$params['general']);  
 }  
 //生活方式  
 if (!empty($params['lifestyle'])){  
 $this->handleExaminationLifestyle($examination,$params['lifestyle']);  
 }  
 //脏器功能  
 if (!empty($params['organ\_function'])){  
 $this->handleExaminationOrganFunction($examination,$params['organ\_function']);  
 }  
 //查体  
 if (!empty($params['body'])){  
 $this->handleExaminationBody($examination,$params['body']);  
 }  
 //辅助检查  
 if (!empty($params['auxiliary'])){  
 $this->handleExaminationAuxiliary($examination,$params['auxiliary']);  
 }  
 //中医体质辨识  
 if (!empty($params['physique\_recognition'])){  
 $this->handleExaminationPhysiqueRecognition($examination,$params['physique\_recognition']);  
 }  
 //现存健康问题  
 if (!empty($params['health\_problem'])){  
 $this->handleExaminationHealthProblem($examination,$params['health\_problem']);  
 }  
 //住院病床史  
 if (!empty($params['hospitalizations'])){  
 $this->handleExaminationHospitalizations($examination,$params['hospitalizations']);  
 }  
 //用药情况  
 if (!empty($params['medications'])){  
 $this->handleExaminationMedications($examination,$params['medications']);  
 }  
 //疫苗接种  
 if (!empty($params['vaccinations'])){  
 $this->handleExaminationVaccinations($examination,$params['vaccinations']);  
 }  
 //评价与指导  
 if (!empty($params['judge'])){  
 $this->handleExaminationJudge($examination,$params['judge']);  
 }  
 //计算检查项数  
 $this->handleExaminationNum($examination);  
 return $examination;  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function updateExamination($adminId,$id,array $params)  
 {  
 try {  
 return DB::transaction(function ()use ($adminId,$id,$params){  
 $examination = Examination::query()->findOrFail($id);  
 $this->checkPermission($adminId,$examination->document\_id);  
 $examination->updated\_at = now();  
 $examination->saveOrFail();  
 //一般状况  
 if (!empty($params['general'])){  
 $this->handleExaminationGeneral($examination,$params['general']);  
 }  
 //生活方式  
 if (!empty($params['lifestyle'])){  
 $this->handleExaminationLifestyle($examination,$params['lifestyle']);  
 }  
 //脏器功能  
 if (!empty($params['organ\_function'])){  
 $this->handleExaminationOrganFunction($examination,$params['organ\_function']);  
 }  
 //查体  
 if (!empty($params['body'])){  
 $this->handleExaminationBody($examination,$params['body']);  
 }  
 //辅助检查  
 if (!empty($params['auxiliary'])){  
 $this->handleExaminationAuxiliary($examination,$params['auxiliary']);  
 }  
 //中医体质辨识  
 if (!empty($params['physique\_recognition'])){  
 $this->handleExaminationPhysiqueRecognition($examination,$params['physique\_recognition']);  
 }  
 //现存健康问题  
 if (!empty($params['health\_problem'])){  
 $this->handleExaminationHealthProblem($examination,$params['health\_problem']);  
 }  
 //住院病床史  
 if (array\_key\_exists('hospitalizations',$params) && is\_array($params['hospitalizations'])){  
 $this->handleExaminationHospitalizations($examination,$params['hospitalizations']);  
 }  
 //用药情况  
 if (array\_key\_exists('medications',$params) && is\_array($params['medications'])){  
 $this->handleExaminationMedications($examination,$params['medications']);  
 }  
 //疫苗接种  
 if (!empty($params['vaccinations'])){  
 $this->handleExaminationVaccinations($examination,$params['vaccinations']);  
 }  
 //评价与指导  
 if (!empty($params['judge'])){  
 $this->handleExaminationJudge($examination,$params['judge']);  
 }  
 //计算检查项数  
 $this->handleExaminationNum($examination);  
 return $examination;  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 private function handleExaminationGeneral(Examination $examination,array $params)  
 {  
 $service = app(ExaminationGeneralService::class);  
 $general = $examination->examination\_general;  
 if (!$general){  
 //新建  
 $general = $examination->examination\_general()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $general = $service->autoFill($general,$params);  
 }else{  
 //更新  
 $general = $service->autoFill($general,$params,[],false);  
 }  
 if (empty($general->bmi) && $general->height && $general->weight){  
 $bmi = compute\_bmi($general->height,$general->weight);  
 $general->bmi = $bmi;  
 }  
 $general->saveOrFail();  
 return $examination->examination\_general;  
 }  
  
 private function handleExaminationLifestyle(Examination $examination, array $params)  
 {  
 $service = app(ExaminationLifestyleService::class);  
 $lifestyle = $examination->examination\_lifestyle;  
 if (!$lifestyle){  
 //新建  
 $lifestyle = $examination->examination\_lifestyle()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $lifestyle = $service->autoFill($lifestyle,$params);  
 }else{  
 //更新  
 $lifestyle = $service->autoFill($lifestyle,$params,[],false);  
 }  
 $lifestyle->saveOrFail();  
 return $examination->examination\_lifestyle;  
 }  
  
 private function handleExaminationOrganFunction(Examination $examination,array $params)  
 {  
 $service = app(ExaminationOrganFunctionService::class);  
 $organ = $examination->examination\_organ\_function;  
 if (!$organ){  
 $organ = $examination->examination\_organ\_function()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $organ = $service->autoFill($organ,$params);  
 }else{  
 $organ = $service->autoFill($organ,$params,[],false);  
 }  
 $organ->saveOrFail();  
 return $examination->examination\_organ\_function;  
 }  
  
 private function handleExaminationBody(Examination $examination,array $params)  
 {  
 $service = app(ExaminationBodyService::class);  
 $body = $examination->examination\_body;  
 if (!$body){  
 $body = $examination->examination\_body()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $body = $service->autoFill($body,$params);  
 }else{  
 $body = $service->autoFill($body,$params,[],false);  
 }  
 $body->saveOrFail();  
 return $examination->examination\_body;  
 }  
  
 private function handleExaminationAuxiliary(Examination $examination,array $params)  
 {  
 $service = app(ExaminationAuxiliaryService::class);  
 $auxiliary = $examination->examination\_auxiliary;  
 if (!$auxiliary){  
 $auxiliary = $examination->examination\_auxiliary()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $auxiliary = $service->autoFill($auxiliary,$params);  
 }else{  
 $auxiliary = $service->autoFill($auxiliary,$params,[],false);  
 }  
 $auxiliary->saveOrFail();  
 return $examination->examination\_auxiliary;  
 }  
  
 private function handleExaminationPhysiqueRecognition(Examination $examination,array $params)  
 {  
 $service = app(ExaminationPhysiqueRecognitionService::class);  
 $recognition = $examination->examination\_physique\_recognition;  
 if (!$recognition){  
 $recognition = $examination->examination\_physique\_recognition()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $recognition = $service->autoFill($recognition,$params);  
 }else{  
 $recognition = $service->autoFill($recognition,$params,[],false);  
 }  
 $recognition->saveOrFail();  
 return $examination->examination\_physique\_recognition;  
 }  
  
 private function handleExaminationHealthProblem(Examination $examination,array $params)  
 {  
 $service = app(ExaminationHealthProblemService::class);  
 $healthProblem = $examination->examination\_health\_problem;  
 if (!$healthProblem){  
 $healthProblem = $examination->examination\_health\_problem()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $healthProblem = $service->autoFill($healthProblem,$params);  
 }else{  
 $healthProblem = $service->autoFill($healthProblem,$params,[],false);  
 }  
 $healthProblem->saveOrFail();  
 return $examination->examination\_health\_problem;  
 }  
  
 private function handleExaminationHospitalizations(Examination $examination,array $params)  
 {  
 $service = app(ExaminationHospitalizationService::class);  
 $examination->examination\_hospitalizations()->delete();  
 foreach ($params as $item){  
 if (empty($item['start\_at']) && empty($item['end\_at']) && empty($item['reason']) &&  
 empty($item['hospital\_name']) && empty($item['medical\_record\_number'])){  
 continue;  
 }  
 $hospitalization = $examination->examination\_hospitalizations()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $hospitalization = $service->autoFill($hospitalization,$item);  
 $hospitalization->saveOrFail();  
 }  
 return $examination->examination\_hospitalizations;  
 }  
  
 public function handleExaminationMedications(Examination $examination,array $params)  
 {  
 $service = app(ExaminationMedicationService::class);  
 $examination->examination\_medications()->delete();  
 foreach ($params as $item){  
 if (empty($item['dosage']) && empty($item['dosage\_2']) && empty($item['drug\_name']) &&  
 empty($item['medication\_compliance']) && empty($item['medication\_time']) && empty($item['usage'])){  
 continue;  
 }  
 $medication = $examination->examination\_medications()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $medication = $service->autoFill($medication,$item);  
 $medication->saveOrFail();  
 }  
 return $examination->examination\_medications;  
 }  
  
 public function handleExaminationVaccinations(Examination $examination,array $params)  
 {  
 $service = app(ExaminationVaccinationService::class);  
 $examination->examination\_vaccinations()->delete();  
 foreach ($params as $item){  
 if (empty($item['name']) && empty($item['hospital\_name']) && empty($item['vaccination\_date'])){  
 continue;  
 }  
 $vaccination = $examination->examination\_vaccinations()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $vaccination = $service->autoFill($vaccination,$item);  
 $vaccination->saveOrFail();  
 }  
 return $examination->examination\_vaccinations;  
 }  
  
 public function handleExaminationJudge(Examination $examination,array $params)  
 {  
 $service = app(ExaminationJudgeService::class);  
 $judge = $examination->examination\_judge;  
 if (!$judge){  
 $judge = $examination->examination\_judge()->create([  
 'document\_id' => $examination->document\_id,  
 ]);  
 $judge = $service->autoFill($judge,$params);  
 }else{  
 $judge = $service->autoFill($judge,$params,[],false);  
 }  
 $judge->saveOrFail();  
 return $examination->examination\_judge;  
 }  
  
 public function handleExaminationNum(Examination $examination)  
 {  
 $checkedNum = 0;  
 $document = $examination->document;  
 $general = $examination->examination\_general()->first();  
 if (Carbon::parse($document->birthday)->diffInYears(now()) >= 65){  
 $examination->total\_num = 126;  
 if ($general){  
 !is\_null($general->self\_health\_status) && $checkedNum++;  
 !is\_null($general->self\_care) && $checkedNum++;  
 (!is\_null($general->cognitive\_function) ||  
 !is\_null($general->cognitive\_function\_score)) && $checkedNum++;  
 (!is\_null($general->emotional\_state) ||  
 !is\_null($general->emotional\_state\_score)) && $checkedNum++;  
 }  
 }else{  
 $examination->total\_num = 122;  
 }  
 //一般状况  
 if ($general){  
 !is\_null($general->symptom) && $checkedNum++;  
 !is\_null($general->temperature) && $checkedNum++;  
 !is\_null($general->respiratory\_rate) && $checkedNum++;  
 !is\_null($general->heart\_rate) && $checkedNum++;  
 !is\_null($general->height) && $checkedNum++;  
 !is\_null($general->weight) && $checkedNum++;  
 !is\_null($general->waistline) && $checkedNum++;  
 !is\_null($general->bmi) && $checkedNum++;  
 (!is\_null($general->blood\_pressure\_left1) ||  
 !is\_null($general->blood\_pressure\_left2)) && $checkedNum++;  
 (!is\_null($general->blood\_pressure\_right1) ||  
 !is\_null($general->blood\_pressure\_right2)) && $checkedNum++;  
 }  
 //生活方式  
 $lifestyle = $examination->examination\_lifestyle()->first();  
 if ($lifestyle){  
 !is\_null($lifestyle->physical\_exercise\_frequency) && $checkedNum++;  
 !is\_null($lifestyle->smoking\_status) && $checkedNum++;  
 !is\_null($lifestyle->drinking\_frequency) && $checkedNum++;  
 !is\_null($lifestyle->occupational\_disease) && $checkedNum++;  
 }  
 //脏器功能  
 $organFunction = $examination->examination\_organ\_function()->first();  
 if ($organFunction){  
 !is\_null($organFunction->oral\_cavity\_lips) && $checkedNum++;  
 !is\_null($organFunction->oral\_cavity\_dentition) && $checkedNum++;  
 !is\_null($organFunction->oral\_cavity\_pharynx) && $checkedNum++;  
 (!is\_null($organFunction->vision\_left) ||  
 !is\_null($organFunction->correct\_vision\_left)) && $checkedNum++;  
 (!is\_null($organFunction->vision\_right) ||  
 !is\_null($organFunction->correct\_vision\_right)) && $checkedNum++;  
 !is\_null($organFunction->hearing) && $checkedNum++;  
 !is\_null($organFunction->motor\_function) && $checkedNum++;  
 }  
 //查体  
 $body = $examination->examination\_body()->first();  
 if ($body){  
 !is\_null($body->fundus) && $checkedNum++;  
 !is\_null($body->skin) && $checkedNum++;  
 !is\_null($body->sclera) && $checkedNum++;  
 !is\_null($body->lymph\_nodes) && $checkedNum++;  
 !is\_null($body->barrel\_chest) && $checkedNum++;  
 !is\_null($body->breath\_sounds) && $checkedNum++;  
 !is\_null($body->rae) && $checkedNum++;  
 !is\_null($body->heart\_rate) && $checkedNum++;  
 !is\_null($body->heart\_rate\_judge) && $checkedNum++;  
 !is\_null($body->heart\_rate\_murmur) && $checkedNum++;  
 !is\_null($body->abdomen\_tenderness) && $checkedNum++;  
 !is\_null($body->abdomen\_mass) && $checkedNum++;  
 !is\_null($body->abdomen\_hepatomegaly) && $checkedNum++;  
 !is\_null($body->abdomen\_splenomegaly) && $checkedNum++;  
 !is\_null($body->abdomen\_mobile\_voiced) && $checkedNum++;  
 !is\_null($body->lower\_extremity\_edema) && $checkedNum++;  
 !is\_null($body->arterial\_pulsation) && $checkedNum++;  
 !is\_null($body->anus) && $checkedNum++;  
 !is\_null($body->breast) && $checkedNum++;  
 !is\_null($body->vulva) && $checkedNum++;  
 !is\_null($body->vaginal) && $checkedNum++;  
 !is\_null($body->cervix) && $checkedNum++;  
 !is\_null($body->palace\_body) && $checkedNum++;  
 !is\_null($body->appendix) && $checkedNum++;  
 }  
 //辅助检查  
 $auxiliary = $examination->examination\_auxiliary()->first();  
 if ($auxiliary){  
 !is\_null($auxiliary->hemoglobin) && $checkedNum++;  
 !is\_null($auxiliary->leukocyte) && $checkedNum++;  
 !is\_null($auxiliary->platelets) && $checkedNum++;  
 !is\_null($auxiliary->urine\_protein) && $checkedNum++;  
 !is\_null($auxiliary->urine\_sugar) && $checkedNum++;  
 !is\_null($auxiliary->urine\_ketone\_bodies) && $checkedNum++;  
 !is\_null($auxiliary->urine\_occult\_blood) && $checkedNum++;  
 !is\_null($auxiliary->electrocardiogram) && $checkedNum++;  
 !is\_null($auxiliary->heart\_rate\_judge) && $checkedNum++;  
 !is\_null($auxiliary->fasting\_blood\_glucose) && $checkedNum++;  
 !is\_null($auxiliary->urine\_microalbumin) && $checkedNum++;  
 !is\_null($auxiliary->fecal\_occult\_blood) && $checkedNum++;  
 !is\_null($auxiliary->glycated\_hemoglobin) && $checkedNum++;  
 !is\_null($auxiliary->hepatitis\_b\_antigen) && $checkedNum++;  
 !is\_null($auxiliary->serum\_alanine\_aminotransferase) && $checkedNum++;  
 !is\_null($auxiliary->serum\_aspartate\_aminotransferase) && $checkedNum++;  
 !is\_null($auxiliary->albumin) && $checkedNum++;  
 !is\_null($auxiliary->total\_bilirubin) && $checkedNum++;  
 !is\_null($auxiliary->conjugated\_bilirubin) && $checkedNum++;  
 !is\_null($auxiliary->serum\_creatinine) && $checkedNum++;  
 !is\_null($auxiliary->blood\_urea) && $checkedNum++;  
 !is\_null($auxiliary->blood\_potassium\_concentration) && $checkedNum++;  
 !is\_null($auxiliary->serum\_sodium\_concentration) && $checkedNum++;  
 !is\_null($auxiliary->total\_cholesterol) && $checkedNum++;  
 !is\_null($auxiliary->triglycerides) && $checkedNum++;  
 !is\_null($auxiliary->serum\_low\_density\_lipoprotein\_cholesterol) && $checkedNum++;  
 !is\_null($auxiliary->serum\_high\_density\_lipoprotein\_cholesterol) && $checkedNum++;  
 !is\_null($auxiliary->abdominal\_ultrasound) && $checkedNum++;  
 !is\_null($auxiliary->extra\_ultrasound) && $checkedNum++;  
 !is\_null($auxiliary->cervical\_smear) && $checkedNum++;  
 !is\_null($auxiliary->chest\_x\_ray) && $checkedNum++;  
 }  
 //中医体质  
 $recognition = $examination->examination\_physique\_recognition()->first();  
 if ($recognition){  
 $answers = trim($recognition->answers,'[]');  
 $checked = 0;//33道题  
 foreach (explode(',',$answers) as $v){  
 if ($v !== "" && $v !== "null"){  
 $checked++;  
 }  
 }  
 $checkedNum += $checked;  
 }  
 //现存主要健康问题  
 $healthProblem = $examination->examination\_health\_problem()->first();  
 if ($healthProblem){  
 !is\_null($healthProblem->cerebrovascular\_disease) && $checkedNum++;  
 !is\_null($healthProblem->kidney\_disease) && $checkedNum++;  
 !is\_null($healthProblem->heart\_disease) && $checkedNum++;  
 !is\_null($healthProblem->vascular\_disease) && $checkedNum++;  
 !is\_null($healthProblem->eye\_disease) && $checkedNum++;  
 !is\_null($healthProblem->nervous\_system\_disease) && $checkedNum++;  
 !is\_null($healthProblem->other\_system\_disease) && $checkedNum++;  
 }  
 //住院用药情况  
 if ($examination->examination\_hospitalizations()->where('type',ExaminationHospitalization::TYPE\_1)->count() > 0){  
 $checkedNum++;  
 }  
 if ($examination->examination\_hospitalizations()->where('type',ExaminationHospitalization::TYPE\_2)->count() > 0){  
 $checkedNum++;  
 }  
 if ($examination->examination\_medications()->count() > 0){  
 $checkedNum++;  
 }  
 //接种史评价指导  
 if ($examination->examination\_vaccinations()->count() > 0){  
 $checkedNum++;  
 }  
 $judge = $examination->examination\_judge()->first();  
 if($judge){  
 (!is\_null($judge->is\_normal) ||  
 !is\_null($judge->abnormal1)||  
 !is\_null($judge->abnormal2)||  
 !is\_null($judge->abnormal3)||  
 !is\_null($judge->abnormal4)||  
 !is\_null($judge->abnormal5)||  
 !is\_null($judge->abnormal6)) && $checkedNum++;  
 (!is\_null($judge->suggests) ||  
 !is\_null($judge->risk\_factor\_control)||  
 !is\_null($judge->target\_weight)||  
 !is\_null($judge->suggest\_vaccination)||  
 !is\_null($judge->extra)) && $checkedNum++;  
 }  
 $examination->checked\_num = $checkedNum;  
 $examination->saveOrFail();  
 return $examination;  
 }  
  
 public function postFields(): array  
 {  
 return [  
 'document\_id','general','lifestyle','organ\_function','body',  
 'auxiliary','physique\_recognition','health\_problem','hospitalizations','medications',  
 "vaccinations","judge",  
 ];  
 }  
  
 public function getPaginator($adminId,$documentId)  
 {  
 $this->checkPermission($adminId,$documentId);  
 return Examination::query()  
 ->where('document\_id',$documentId)  
 ->orderByDesc('id')  
 ->paginate();  
 }  
  
 public function getDetail($adminId, $id)  
 {  
 $examination = Examination::query()->findOrFail($id);  
 $this->checkPermission($adminId,$examination->document\_id);  
 return $examination;  
 }  
  
 public function checkPermission($adminId,$documentId)  
 {  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
 $document = Document::query()->findOrFail($documentId);  
 if (!in\_array($document->admin\_id,$adminIds)){  
 throw new ApiException('暂无权限');  
 }  
 }  
  
 public function deleteExamination($adminId,$id)  
 {  
 try {  
 DB::transaction(function ()use ($adminId,$id){  
 $examination = Examination::query()->findOrFail($id);  
 $this->checkPermission($adminId,$examination->document\_id);  
 $examination->examination\_general()->delete();  
 //todo 删除其他关联表数据  
  
 $examination->delete();  
 });  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function getReport($id)  
 {  
 $examination = Examination::query()->findOrFail($id);  
 $checkDate = date('Y-m-d', strtotime($examination->created\_at)); //查体日期  
 $reportService = app(ExaminationReportService::class);  
 $setValue = [  
 //封面  
 'HOSPITAL' => $examination->document->person->hospital->name ?? '', //机构  
 'AGE' => now()->diffInYears($examination->document->birthday), //年龄  
 'TIJIAN\_NO' => $examination->document->number, //体检编号  
 'TIJIAN\_DATE' => $checkDate, //查体日期  
 'USER\_NAME' => $examination->document->name, //姓 名  
 'SEX' => $examination->document->sex == 1 ? '男' : '女', //性 别  
 'PHONE' => $examination->document->phone, //联系电话  
 'DOCTOR' => $examination->document->person->real\_name ?? '', //责任医生  
  
 //一般情况  
 'sg' => $examination->examination\_general->height, //身高  
 'tz' => $examination->examination\_general->weight, //体重  
 'bmi' => $examination->examination\_general->bmi, //BMI  
 'rgy' => '', //血压（收缩压）  
 'rdy' => '', //血压（舒张压）  
 'ml' => $examination->examination\_general->heart\_rate, //脉率  
  
 //血常规检验报告单  
 'WBC' => $examination->examination\_auxiliary->leukocyte, 'WBC\_T' => '', 'WBC\_P' => '', //WBC血白细胞数目  
 'LYM' => '', 'LYM\_T' => '', 'LYM\_P' => '', //Lymph#淋巴细胞总数  
 'MID' => '', 'MID\_T' => '', 'MID\_P' => '', //Mid#中间细胞总数  
 'NEU' => '', 'NEU\_T' => '', 'NEU\_P' => '', //Gran#中性粒细胞  
 'LYM\_' => '', 'LYM\_\_T' => '', 'LYM\_\_P' => '', //Lymph%淋巴细胞比率  
 'MID\_' => '', 'MID\_\_T' => '', 'MID\_\_P' => '', //Mid%中间细胞比率  
 'NEU\_' => '', 'NEU\_\_T' => '', 'NEU\_\_P' => '', //Gran%中性粒细胞比率  
 'HGB' => $examination->examination\_auxiliary->hemoglobin, 'HGB\_T' => '', 'HGB\_P' => '', //HGB血红蛋白  
 'RBC' => '', 'RBC\_T' => '', 'RBC\_P' => '', //RBC红细胞总数  
 'HCT' => '', 'HCT\_T' => '', 'HCT\_P' => '', //HCT红细胞压积  
 'MCV' => '', 'MCV\_T' => '', 'MCV\_P' => '', //MCV红细胞平均体积  
 'MCH' => '', 'MCH\_T' => '', 'MCH\_P' => '', //MCH平均血红蛋白量  
 'MCHC' => '', 'MCHC\_T' => '', 'MCHC\_P' => '', //MCHC平均血红蛋白浓度  
 'RDW\_CV' => '', 'RDW\_CV\_T' => '', 'RDW\_CV\_P' => '', //RDW-CV红细胞分布宽度变异系数  
 'RDW\_SD' => '', 'RDW\_SD\_T' => '', 'RDW\_SD\_P' => '', //RDW\_SD红细胞分布宽度标准差  
 'PLT' => $examination->examination\_auxiliary->platelets, 'PLT\_T' => '', 'PLT\_P' => '', //PLT血小板数目  
 'MPV' => '', 'MPV\_T' => '', 'MPV\_P' => '', //MPV血小板平均体积  
 'PDW' => '', 'PDW\_T' => '', 'PDW\_P' => '', //PDW血小板分布宽度  
 'PCT' => '', 'PCT\_T' => '', 'PCT\_P' => '', //PCT血小板压积  
 'xqdoctor' => '', //校验医师  
 'xqlookdoctor' => '', //审核医师  
 'tjdate' => $checkDate, //检查日期  
  
 //生化检验报告单  
 'ALT' => $examination->examination\_auxiliary->serum\_alanine\_aminotransferase, 'ALT\_T' => '', 'ALT\_P' => '', //谷丙转氨酶ALT  
 'AST' => $examination->examination\_auxiliary->serum\_aspartate\_aminotransferase, 'AST\_T' => '', 'AST\_P' => '', //谷草转氨酶AST  
 'TBIL' => $examination->examination\_auxiliary->total\_bilirubin, 'TBIL\_T' => '', 'TBIL\_P' => '', //总胆红素TB  
 'FPGL' => '', 'FPGL\_T' => '', 'FPGL\_P' => '', //葡萄糖GLU  
 'CHOL' => $examination->examination\_auxiliary->total\_cholesterol, 'CHOL\_T' => '', 'CHOL\_P' => '', //胆固醇TC  
 'TG' => $examination->examination\_auxiliary->triglycerides, 'TG\_T' => '', 'TG\_P' => '', //甘油三酯TG  
 'HDL' => $examination->examination\_auxiliary->serum\_high\_density\_lipoprotein\_cholesterol, 'HDL\_T' => '', 'HDL\_P' => '', //高密度脂蛋白胆固醇HDL\_C  
 'LDL' => $examination->examination\_auxiliary->serum\_low\_density\_lipoprotein\_cholesterol, 'LDL\_T' => '', 'LDL\_P' => '', //低密度脂蛋白胆固醇LDL\_C  
 'BUN' => $examination->examination\_auxiliary->blood\_urea, 'BUN\_T' => '', 'BUN\_P' => '', //尿素UREA  
 'CREA' => $examination->examination\_auxiliary->serum\_creatinine, 'CREA\_T' => '', 'CREA\_P' => '', //肌酐Cr  
  
 //尿常规检验报告单  
 'URO' => '', 'URO\_T' => '', //尿胆原UBG  
 'GLU' => $examination->examination\_auxiliary->urine\_sugar, 'GLU\_T' => '', //葡萄糖GLU  
 'KET' => $examination->examination\_auxiliary->urine\_ketone\_bodies, 'KET\_T' => '', //酮体KET  
 'BIL' => '', 'BIL\_T' => '', //胆红素BIL  
 'PRO' => $examination->examination\_auxiliary->urine\_protein, 'PRO\_T' => '', //蛋白质PRO  
 'NIT' => '', 'NIT\_T' => '', //亚硝酸盐NIT  
 'PH' => '', 'PH\_T' => '', //PH值  
 'BLD' => $examination->examination\_auxiliary->urine\_occult\_blood, 'BLD\_T' => '', //尿潜血  
 'SG' => '', 'SG\_T' => '', //比重  
 'LEU' => '', 'LEU\_T' => '', //白细胞LEU  
 'ASC' => '', 'ASC\_T' => '', //抗坏血酸  
  
 //超声检测报告单  
 'imgbcpic1' => '', 'imgbcpic2' => '',  
 'bcsj' => '', //超声所见  
 'bcts' => '', //诊断意见  
 'bcdoctor' => '', //检查医生  
  
 //心电图检测报告单  
 'imgxdpic' => $examination->examination\_auxiliary->heart\_rate\_img ?  
 [  
 'data' => OpenToolService::getObjectUrl($examination->examination\_auxiliary->heart\_rate\_img),  
 'type' => 2,  
 'style' => [  
 'width' => 600,  
 'height' => 600  
 ]  
 ] : '',  
 'xdyc' => $examination->examination\_auxiliary->heart\_rate\_judge, //心电结论  
 'bcdoctor' => $examination->document->person->real\_name ?? '', //检查医生  
  
 //异常结果及健康指导  
 'ycpj' => $reportService->examinationJudge($examination)  
 ];  
  
 foreach ($setValue as $k => $v){  
 if (is\_array($v)){  
 $data[$k] = $v;  
 }else{  
 $data[$k] = ['data' => $v, 'type' => 1];  
 }  
 }  
  
 //体质辨识结果  
 $complexBlock = $reportService->examinationPhysiqueRecognition($examination);  
  
 $service = new TemplateService();  
 $templatePath = app\_path('Services/TemplateReport/template/健康体检报告源文件.docx');  
 $service->generateReport($templatePath, ['SetValue' => $data, 'ComplexBlock' => ['中医药健康管理' => $complexBlock]]);  
 }  
  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/15  
 \* Time 11:40 上午  
 \*/  
  
namespace App\Services\Lists;  
  
use App\Http\Filters\ListDeviceAbnormalFilter;  
use App\Models\Lists\ListDeviceAbnormal;  
use App\Services\AdminService;  
use App\Services\Device\DeviceAbnormalService;  
use Illuminate\Database\Eloquent\Builder;  
use Throwable;  
  
class ListDeviceAbnormalService  
{  
 /\*\*  
 \* 创建 or 更新档案最新异常  
 \*  
 \* @param $documentId  
 \* @throws Throwable  
 \*/  
 public function createOrUpdateDocumentAbnormal($documentId)  
 {  
 $service = app(DeviceAbnormalService::class);  
 $latest = $service->getLatest($documentId);  
 $first = $this->findByDocumentId($documentId);  
 if ($first && !$latest){  
 $first->delete();  
 return;  
 }  
 if (!$first){  
 $first = new ListDeviceAbnormal([  
 'document\_id' => $latest->document\_id,  
 'admin\_id' => $latest->admin\_id,  
 ]);  
 }  
 $first->title = $latest->title;  
 $first->saveOrFail();  
 }  
  
 public function findByDocumentId($documentId)  
 {  
 return ListDeviceAbnormal::query()  
 ->where('document\_id',$documentId)  
 ->orderByDesc('id')  
 ->first();  
 }  
  
 public function getPaginator($adminId)  
 {  
 $filter = app(ListDeviceAbnormalFilter::class);  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
 return ListDeviceAbnormal::query()  
 ->whereHas('document',function (Builder $builder)use ($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->filter($filter)  
 ->with(['document.abnormals','document.public\_labels'])  
 ->orderByDesc('updated\_at')  
 ->paginate();  
 }  
  
 public function getByNum($num,$adminId)  
 {  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
 return ListDeviceAbnormal::query()  
 ->whereHas('document',function (Builder $builder)use ($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->with(['document.public\_labels'])  
 ->orderByDesc('updated\_at')  
 ->take($num)  
 ->get();  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/12/17  
 \* Time 3:48 下午  
 \*/  
  
namespace App\Services\Rbac;  
  
use App\Exceptions\ApiException;  
use App\Models\Rbac\MenuModel;  
use App\Services\RestService;  
  
class MenuService extends RestService  
{  
  
 public function postFields(): array  
 {  
 return [  
 'pid', 'title', 'uri', 'sort'  
 ];  
 }  
  
 public function createMenu(array $params)  
 {  
 $menu = MenuModel::query()->firstOrCreate(['title' => $params['title'], 'pid' => $params['pid']], $params);  
 if (!$menu->wasRecentlyCreated){  
 throw new ApiException('同级菜单名称已存在');  
 }  
 }  
  
 public function updateMenu($id, array $params)  
 {  
 $menu = MenuModel::query()->findOrFail($id);  
 if (!empty($params['title']) || !empty($params['pid'])){  
 $title = !empty($params['title']) ? $params['title'] : $menu['title'];  
 $pid = !empty($params['pid']) ? $params['pid'] : $menu['pid'];  
 MenuModel::query()->where('title', $title)  
 ->where('pid', $pid)  
 ->where('id', '<>', $id)  
 ->doesntExistOr(function(){  
 throw new ApiException('同级菜单名称已存在');  
 });  
 }  
 $menu = $this->autoFill($menu, $params);  
 $menu->saveOrFail();  
 }  
  
 public function indexMenu()  
 {  
 $all = MenuModel::query()->get()->toArray();  
 return $this->toTree($all,null);  
 }  
  
 public function toTree(array $items,$pid = 0)  
 {  
 $res = [];  
 foreach ($items as $k => $item){  
 if ($item['pid'] == $pid){  
 $children = $this->toTree($items,$item['id']);  
 if (!empty($children)){  
 $item['children'] = $children;  
 }  
 unset($items[$k]);  
 $res[] = $item;  
 }  
 }  
 return $res;  
 }  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/12/17  
 \* Time 3:48 下午  
 \*/  
  
namespace App\Services\Rbac;  
  
use App\Exceptions\ApiException;  
use App\Http\Filters\PermissionFilter;  
use App\Models\Rbac\Permission;  
use App\Services\RestService;  
  
class PermissionService extends RestService  
{  
  
 public function postFields(): array  
 {  
 return [  
 'name','http\_method','http\_path',  
 ];  
 }  
  
 public function indexPermission()  
 {  
 $role = Permission::query()  
 ->filter(app(PermissionFilter::class))  
 ->paginate();  
 return $role;  
 }  
  
 public function createPermission(array $params)  
 {  
 $permission = Permission::query()->firstOrCreate(['name' => $params['name']], $params);  
 if (!$permission->wasRecentlyCreated){  
 throw new ApiException('权限名称已存在');  
 }  
 }  
  
 public function updatePermission($id, array $params)  
 {  
 $permission = Permission::query()->findOrFail($id);  
 if (!empty($params['name'])){  
 Permission::query()->where('name', $params['name'])  
 ->where('id', '<>', $id)  
 ->doesntExistOr(function(){  
 throw new ApiException('权限名称已存在');  
 });  
 }  
 $permission = $this->autoFill($permission, $params);  
 $permission->saveOrFail();  
 }  
  
 public function getPermissionTree()  
 {  
 $all = Permission::query()->get()->toArray();  
 return $this->toTree($all,null);  
 }  
  
 public function toTree(array $items,$pid = 0)  
 {  
 $res = [];  
 foreach ($items as $k => $item){  
 if ($item['pid'] == $pid){  
 $children = $this->toTree($items,$item['id']);  
 if (!empty($children)){  
 $item['children'] = $children;  
 }  
 unset($items[$k]);  
 $res[] = $item;  
 }  
 }  
 return $res;  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/10  
 \* Time 5:38 下午  
 \*/  
  
namespace App\Services;  
  
use App\Enums\CommonEnum;  
use App\Exceptions\ApiException;  
use App\Http\Filters\AdminFilter;  
use App\Models\Admin;  
use App\Models\Area;  
use App\Services\Common\AreaService;  
use Illuminate\Database\Eloquent\Builder;  
use Illuminate\Support\Facades\Cache;  
use Illuminate\Support\Facades\Hash;  
use Illuminate\Support\Facades\Redis;  
use Throwable;  
  
class AdminService extends RestService  
{  
 protected $adminIds = [];  
  
 protected $areaIds = [];  
  
 public function postFields(): array  
 {  
 return [  
 'username','password','real\_name','phone', 'id\_number', 'area\_id', 'hospital\_id', 'status'  
 ];  
 }  
  
 public function patchFields(): array  
 {  
 return [  
 'username','password','avatar','status','doctor\_id','role',  
 ];  
 }  
  
 public function getByUsername($username)  
 {  
 return Admin::query()  
 ->where('username', $username)  
 ->firstOrFail();  
 }  
  
 public function createOrUpdateAdmin(array $params)  
 {  
 if (!empty($params['id'])) {  
 $admin = Admin::findOrFail($params['id']);  
 if (!empty($params['username']) || !empty($params['phone']) || !empty($params['id\_number'])) {  
 Admin::query()->where('id', '<>', $params['id'])  
 ->where(function(Builder $builder) use ($params){  
 //唯一校验  
 !empty($params['username']) && $builder->where('username', $params['username']);  
 !empty($params['phone']) && $builder->orWhere('phone', $params['phone']);  
 !empty($params['id\_number']) && $builder->orWhere('id\_number', $params['id\_number']);  
 })->doesntExistOr(function(){  
 throw new ApiException('该用户信息已存在');  
 });  
 }  
 //clear admin key  
 $this->clear\_admin\_cache($admin->id);  
 } else {  
 $admin = new Admin();  
 }  
 if (!empty($params['password'])){  
 $params['password'] = Hash::make($params['password']);  
 }  
  
 $admin = $this->autoFill($admin,$params);  
 $admin->saveOrFail();  
  
 //同步角色  
 if (!empty($roleIds = request('role\_ids'))){  
 $admin->roles()->sync($roleIds);  
 }  
 return $admin;  
 }  
  
 public function deleteAdmin(int $id)  
 {  
 try {  
 //todo 检查admin级联资源是否需要删除  
  
 }catch (Throwable $exception){  
 throw new ApiException($exception);  
 }  
 }  
  
 public function getPaginator()  
 {  
 $filter = app(AdminFilter::class);  
 return Admin::query()  
 ->filter($filter)  
 ->with(['roles:id,name', 'hospital:id,name'])  
 ->orderByDesc('admins.id')  
 ->paginate();  
 }  
  
 public function getOptions($keyword)  
 {  
 return Admin::query()  
 ->where(function (Builder $builder)use ($keyword){  
 $condition = '%'.$keyword.'%';  
 $builder->where('username','like',$condition);  
 })  
 ->where('status',CommonEnum::ACCOUNT\_STATUS\_ON)  
 ->select(['id','username','status'])  
 ->get();  
 }  
  
 /\*\*  
 \* 获取管理员的所有下级  
 \*  
 \* @param $adminId  
 \* @param int $areaId  
 \* @return array  
 \*/  
 public function getChildrenAdminIds($adminId,$areaId = 0)  
 {  
 if (empty($areaId)){  
 $admin = Admin::query()->findOrFail($adminId);  
 $areaId = $admin->area\_id;  
 }  
 $key = 'child\_admin\_ids\_'.$adminId.'\_'.$areaId;  
 if ($data = Cache::driver('redis')->get($key)){  
 return json\_decode($data,true);  
 }  
 //账号可查看管辖区域本级和下级所有数据  
 $this->getAdminsByArea(Area::query()->findOrFail($areaId));  
 Cache::driver('redis')->set($key,json\_encode($this->adminIds), 3600);  
 return $this->adminIds;  
 }  
  
 public function getAdminsByArea(Area $area)  
 {  
 $service = app(AreaService::class);  
 $areaId = $service->getAreaIdsByArea($area);  
 $this->adminIds = Admin::query()->whereIn('area\_id', $areaId)->pluck('id')->toArray();  
 }  
  
 /\*\*  
 \* 获取admin可管辖的地区  
 \*  
 \* @param $adminId  
 \* @return array  
 \*/  
 public function getChildrenAreaIds($adminId,$areaId = 0)  
 {  
 $admin = Admin::query()->findOrFail($adminId);  
 if (empty($admin->doctor\_id)){  
 return [];  
 }  
 $area = $admin->doctor->hospital->area;  
 if (!empty($areaId)){  
 $this->getAreaIdsByArea(Area::query()->findOrFail($areaId));  
 }else{  
 $this->getAreaIdsByArea($area);  
 }  
 return $this->areaIds;  
 }  
  
 public function getAreaIdsByArea(Area $area)  
 {  
 $this->areaIds[] = $area->id;  
 foreach ($area->children as $child){  
 $this->getAreaIdsByArea($child);  
 }  
 }  
  
 public function changePassword(Admin $admin, array $params)  
 {  
 if (!Hash::check($params['old\_password'],$admin->password)){  
 throw new ApiException('旧密码有误');  
 }  
 $admin->password = Hash::make($params['password']);  
 $admin->saveOrFail();  
 }  
  
 /\*\*  
 \* 删除用户下的所有缓存  
 \* @param $adminId  
 \*/  
 public function clear\_admin\_cache($adminId)  
 {  
 $key = 'child\_admin\_ids\_' . $adminId . '\_\*';  
 //切换配置  
 $redis = app('redis')->connection('cache');  
 $keys = $redis->keys(config('cache.prefix') . ':' . $key);  
 //前缀  
 $prefix = config('database.redis.options.prefix');;  
 foreach ($keys as $\_key){  
 $redis->del(str\_replace($prefix, '', $\_key));  
 }  
 }  
  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/12/24  
 \* Time 10:15 上午  
 \*/  
  
namespace App\Services;  
  
use App\Models\Admin;  
use App\Models\LoginRecord;  
use Illuminate\Support\Facades\Log;  
  
class LoginRecordService  
{  
 public function createLoginRecord(Admin $admin,$extra = [])  
 {  
 try {  
 $record = new LoginRecord([  
 'admin\_id' => $admin->id,  
 'username' => $admin->username,  
 'ip' => $extra['ip'] ?? null,  
 'ua' => $extra['ua'] ?? null,  
 ]);  
 $record->saveOrFail();  
 }catch (\Throwable $exception){  
 Log::channel('scs')->error('登录记录失败：'.$exception->getMessage());  
 }  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/12  
 \* Time 3:15 下午  
 \*/  
  
namespace App\Services;  
  
use Illuminate\Database\Eloquent\Model;  
  
abstract class RestService  
{  
 abstract public function postFields():array;  
  
 public function patchFields(){  
 return $this->postFields();  
 }  
  
 /\*\*  
 \* 自动填充属性  
 \*  
 \* @param Model $object  
 \* @param array $params  
 \* @param array $except  
 \* @param bool $isPost  
 \* @return Model  
 \*/  
 public function autoFill(Model $object, array $params, array $except = [], bool $isPost = true)  
 {  
 foreach ($params as $k => $v){  
 if (in\_array($k,$except)){  
 continue;  
 }  
 if ($isPost && !in\_array($k,$this->postFields())){  
 continue;  
 }  
 if (!$isPost && !in\_array($k,$this->patchFields())){  
 continue;  
 }  
 $object->$k = $v;  
 }  
 return $object;  
 }  
  
 public function mergeDefaults(array $defaults,array $params)  
 {  
 return array\_merge($defaults,$params);  
 }  
  
 public function filterParams(array $params)  
 {  
 $res = [];  
 foreach ($params as $k => $v){  
 if (is\_array($v)){  
 foreach ($v as $i => $j){  
 if ($j !== '' && !is\_null($j)){  
 $res[$k][$i] = $j;  
 }  
 }  
 }else{  
 if ($v !== '' && !is\_null($v)){  
 $res[$k] = $v;  
 }  
 }  
 }  
 return $res;  
 }  
}

<?php  
/\*\*  
 \* Created by  
 \* Author purelight  
 \* Date 2021/11/15  
 \* Time 4:13 下午  
 \*/  
  
namespace App\Services;  
  
use App\Enums\CommonEnum;  
use App\Models\Device\DeviceAbnormal;  
use App\Models\Device\DeviceGeneral;  
use App\Models\Device\DeviceHeartB;  
use App\Models\Device\DeviceRecord;  
use App\Models\Device\DeviceUrineRoutine;  
use App\Models\Document\Document;  
use App\Models\Document\DocumentDiseaseLabel;  
use App\Models\Document\DocumentPublicLabel;  
use App\Models\Document\DocumentResidenceLabel;  
use App\Models\Lists\ListDeviceAbnormal;  
use App\Models\TmpFollowUp;  
use App\Services\Tool\VideoService;  
use Illuminate\Database\Eloquent\Builder;  
use Illuminate\Support\Facades\DB;  
  
class StatisticService  
{  
 public function getIndexStatistic($adminId)  
 {  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
// print\_r($adminIds);  
 return [  
 'document\_total' => Document::query()->whereIn('admin\_id',$adminIds)->count(),  
 'active\_document\_total' => 0,  
 'examination\_document\_total' => 0,  
 'abnormal\_document\_total' => Document::query()->whereIn('admin\_id',$adminIds)->whereHas('list\_device\_abnormal')->count(),  
 'public\_labels\_statistic' => $this->getPublicLabelStatistic($adminIds),  
 ];  
 }  
  
 public function getPublicLabelStatistic($adminIds)  
 {  
 $statistic = DocumentPublicLabel::query()  
 ->whereHas('document',function (Builder $builder)use ($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->select(DB::raw('value,count(\*) as value\_total'))  
 ->groupBy('value')  
 ->get()  
 ->pluck('value\_total','value')  
 ->toArray();  
 $res = [];  
 foreach (CommonEnum::PUBLIC\_LABEL\_MAP as $k => $v){  
 if (!in\_array($k,[  
 CommonEnum::PUBLIC\_LABEL\_COMMON,  
 CommonEnum::PUBLIC\_LABEL\_ELDER,  
 CommonEnum::PUBLIC\_LABEL\_HYPERTENSION,  
 CommonEnum::PUBLIC\_LABEL\_DIABETES,  
 CommonEnum::PUBLIC\_LABEL\_PREGNANT  
 ])){  
 continue;  
 }  
 $res[] = [  
 'key' => $k,  
 'label' => CommonEnum::PUBLIC\_LABEL\_MAP[$k],  
 'value' => $statistic[$k] ?? 0  
 ];  
 }  
 return $res;  
 }  
  
 public function getDocumentGraphByType($type,$adminId)  
 {  
 switch ($type){  
 case CommonEnum::GRAPH\_TYPE\_WEEK:  
 return $this->getDocumentGraphByDayNums(7,$adminId);  
 case CommonEnum::GRAPH\_TYPE\_MONTH:  
 return $this->getDocumentGraphByDayNums(30,$adminId);  
 case CommonEnum::GRAPH\_TYPE\_YEAR:  
 return $this->getDocumentGraphByYear($adminId);  
 }  
 }  
  
 public function getDocumentGraphByYear($adminId)  
 {  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
 //构建months数组  
 $now = now();  
 $months = [$now->format('Y-m')];  
 while (count($months) < 12){  
 $now = $now->subMonth();  
 array\_unshift($months,$now->format('Y-m'));  
 }  
 //增量数据  
 $documentTotals = Document::query()  
 ->select(DB::raw('concat(year(created\_at),"-",month(created\_at)) as month,count(\*) as total'))  
 ->whereIn(DB::raw('concat(year(created\_at),"-",month(created\_at))'),$months)  
 ->whereIn('admin\_id',$adminIds)  
 ->groupBy('month')  
 ->get();  
 //基准数据  
 $baseTotal = Document::query()  
 ->where(DB::raw('concat(year(created\_at),"-",month(created\_at))'),$months[0])  
 ->whereIn('admin\_id',$adminIds)  
 ->count();  
 $res = [];  
 foreach ($months as $i => $month){  
 $currentTotal = 0;  
 foreach ($documentTotals as $item){  
 if ($item['month'] === $month){  
 $currentTotal = $item['total'];  
 }  
 }  
 $res[] = [  
 'date' => $month,  
 'current\_total' => $currentTotal,  
 'total' => $i === 0 ? $baseTotal : $res[$i-1]['total']+$currentTotal,  
 ];  
 }  
 return $res;  
 }  
  
 protected function getDocumentGraphByDayNums($nums,$adminId)  
 {  
 $adminService = app(AdminService::class);  
 $adminIds = $adminService->getChildrenAdminIds($adminId,request('area\_id',0));  
 //构建dates数组  
 $now = now();  
 $dates = [$now->toDateString()];  
 while (count($dates) < $nums){  
 $now = $now->subDay();  
 array\_unshift($dates,$now->toDateString());  
 }  
 //增量数据  
 $documentTotals = Document::query()  
 ->select(DB::raw('date(created\_at) as date,count(\*) as total'))  
 ->whereIn(DB::raw('date(created\_at)'),$dates)  
 ->whereIn('admin\_id',$adminIds)  
 ->groupBy('date')  
 ->get();  
 //基准数据  
 $baseTotal = Document::query()  
 ->where('created\_at','<=',$dates[0].' 23:59:59')  
 ->whereIn('admin\_id',$adminIds)  
 ->count();  
 $res = [];  
 foreach ($dates as $i => $date){  
 $currentTotal = 0;  
 foreach ($documentTotals as $item){  
 if ($item['date'] === $date){  
 $currentTotal = $item['total'];  
 }  
 }  
 $res[] = [  
 'date' => $date,  
 'current\_total' => $currentTotal,  
 'total' => $i === 0 ? $baseTotal : $res[$i-1]['total']+$currentTotal,  
 ];  
 }  
 return $res;  
 }  
  
 public function getDocumentStatisticByAge($adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 $expr = 'case when (birthday >= '."'".now()->subYears(20)->toDateString()."'".') then '."'".CommonEnum::AGE\_P1."'".  
 ' when (birthday >= '."'".now()->subYears(30)->toDateString()."'".') then '."'".CommonEnum::AGE\_P2."'".  
 ' when (birthday >= '."'".now()->subYears(40)->toDateString()."'".') then '."'".CommonEnum::AGE\_P3."'".  
 ' when (birthday >= '."'".now()->subYears(50)->toDateString()."'".') then '."'".CommonEnum::AGE\_P4."'".  
 ' when (birthday >= '."'".now()->subYears(60)->toDateString()."'".') then '."'".CommonEnum::AGE\_P5."'".  
 ' when (birthday >= '."'".now()->subYears(70)->toDateString()."'".') then '."'".CommonEnum::AGE\_P6."'".  
 ' when (birthday >= '."'".now()->subYears(80)->toDateString()."'".') then '."'".CommonEnum::AGE\_P7."'".  
 ' when (birthday >= '."'".now()->subYears(90)->toDateString()."'".') then '."'".CommonEnum::AGE\_P8."'".  
 ' else '."'".CommonEnum::AGE\_P9."'".' end';  
 $data = Document::query()  
 ->selectRaw($expr.' as age,count(\*) as total')  
 ->whereIn('admin\_id',$adminIds)  
 ->groupByRaw($expr)  
 ->get()  
 ->toArray();  
 $res = [];  
 $count = Document::query()->whereIn('admin\_id',$adminIds)->count();  
 foreach (CommonEnum::AGE\_MAP as $v){  
 $total = 0;  
 foreach ($data as $item){  
 if ($item['age'] === $v){  
 $total = $item['total'];  
 }  
 }  
 $res[] = [  
 'age' => $v,  
 'total' => $total,  
 'percentage' => $count == 0 ? 0 : bcmul(bcdiv($total,$count,2),100,2). '%'  
 ];  
 }  
 return $res;  
 }  
  
 public function getDocumentStatisticBySex($adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 $documents = Document::query()  
 ->whereHas('list\_device\_abnormal',function (Builder $builder){  
 $builder->whereNotNull('title');  
 })  
 ->whereIn('admin\_id',$adminIds)  
 ->select(DB::raw('sex,count(\*) as total'))  
 ->groupBy('sex')  
 ->get()  
 ->toArray();  
 $res = [  
 [  
 'sex' => CommonEnum::SEX\_MALE,  
 'total' => 0,  
 ],  
 [  
 'sex' => CommonEnum::SEX\_FEMALE,  
 'total' => 0,  
 ]  
 ];  
 foreach ($res as $i => $item){  
 foreach ($documents as $document){  
 if ($document['sex'] == $item['sex']){  
 $res[$i]['total'] = $document['total'];  
 }  
 }  
 }  
 return $res;  
 }  
  
 public function getDeviceRecordDynamic($lastId,$adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 $res = [];  
 $records = DeviceRecord::query()  
 ->where('id','>',$lastId)  
 ->whereHas('document',function (Builder $builder)use ($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->with([  
 'device\_general','device\_heart\_b','device\_liver\_kidney',  
 'device\_blood\_routine','device\_urine\_routine','document',  
 ])  
 ->orderByDesc('device\_records.id')  
 ->take(10)  
 ->get();  
 foreach ($records as $record){  
 $name = mb\_substr($record->document->name,0,1).  
 str\_repeat('\*',mb\_strlen($record->document->name)-1);  
 $time = $record->created\_at->format('m月d日 H:i');  
 if ($record->device\_general){  
 if ($record->device\_general->bmi\_judge){  
 $res[] = [  
 'id' => $record->id,  
 'name' => $name,  
 'time' => $time,  
 'project' => '身高体重',  
 'judge' => $record->device\_general->bmi\_judge == CommonEnum::JUDGE\_NORMAL ? '正常' : '异常'  
 ];  
 }  
 if ($record->device\_general->temperature\_judge){  
 $res[] = [  
 'id' => $record->id,  
 'name' => $name,  
 'time' => $time,  
 'project' => '体温',  
 'judge' => $record->device\_general->temperature\_judge == CommonEnum::JUDGE\_NORMAL ? '正常' : '异常'  
 ];  
 }  
 if ($record->device\_general->blood\_pressure\_judge){  
 $res[] = [  
 'id' => $record->id,  
 'name' => $name,  
 'time' => $time,  
 'project' => '血压',  
 'judge' => $record->device\_general->blood\_pressure\_judge == CommonEnum::JUDGE\_NORMAL ? '正常' : '异常'  
 ];  
 }  
 if ($record->device\_general->blood\_sugar\_judge){  
 $res[] = [  
 'id' => $record->id,  
 'name' => $name,  
 'time' => $time,  
 'project' => '血糖',  
 'judge' => $record->device\_general->blood\_sugar\_judge == CommonEnum::JUDGE\_NORMAL ? '正常' : '异常'  
 ];  
 }  
 }  
 if ($record->device\_heart\_b){  
 if ($record->device\_heart\_b->heart\_rate\_judge){  
 $res[] = [  
 'id' => $record->id,  
 'name' => $name,  
 'time' => $time,  
 'project' => '心率',  
 'judge' => $record->device\_heart\_b->heart\_rate\_judge == CommonEnum::JUDGE\_NORMAL ? '正常' : '异常'  
 ];  
 }  
 }  
 }  
 return $res;  
 }  
  
 public function getAbnormalStatistic($adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 $res = [];  
 //Bmi  
 $res[] = [  
 'key' => 'BMI',  
 'normal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('bmi\_judge')  
 ->where('bmi\_judge', CommonEnum::JUDGE\_NORMAL)  
 ->count(),  
 'abnormal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('bmi\_judge')  
 ->whereIn('bmi\_judge', [CommonEnum::JUDGE\_LOW, CommonEnum::JUDGE\_HIGH])  
 ->count(),  
 ];  
 //体温  
 /\* $res[] = [  
 'key' => '体温',  
 'normal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('temperature\_judge')  
 ->where('temperature\_judge',CommonEnum::JUDGE\_NORMAL)  
 ->count(),  
 'abnormal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('temperature\_judge')  
 ->whereIn('temperature\_judge',[CommonEnum::JUDGE\_LOW, CommonEnum::JUDGE\_HIGH])  
 ->count(),  
 ];\*/  
 //血压  
 $res[] = [  
 'key' => '血压',  
 'normal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('blood\_pressure\_judge')  
 ->where('blood\_pressure\_judge',CommonEnum::JUDGE\_NORMAL)  
 ->count(),  
 'abnormal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('blood\_pressure\_judge')  
 ->whereIn('blood\_pressure\_judge',[CommonEnum::JUDGE\_LOW, CommonEnum::JUDGE\_HIGH])  
 ->count(),  
 ];  
 //心率  
 $res[] = [  
 'key' => '心率',  
 'normal' => DeviceHeartB::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('heart\_rate\_judge')  
 ->where('heart\_rate\_judge',CommonEnum::JUDGE\_NORMAL)  
 ->count(),  
 'abnormal' => DeviceHeartB::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('heart\_rate\_judge')  
 ->whereIn('heart\_rate\_judge',[CommonEnum::JUDGE\_LOW, CommonEnum::JUDGE\_HIGH])  
 ->count(),  
 ];  
 //血糖  
 $res[] = [  
 'key' => '血糖',  
 'normal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('blood\_sugar\_judge')  
 ->where('blood\_sugar\_judge',CommonEnum::JUDGE\_NORMAL)  
 ->count(),  
 'abnormal' => DeviceGeneral::query()  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->whereNotNull('blood\_sugar\_judge')  
 ->whereIn('blood\_sugar\_judge',[CommonEnum::JUDGE\_LOW, CommonEnum::JUDGE\_HIGH])  
 ->count(),  
 ];  
 return $res;  
 }  
  
 public function getDiseaseLabelStatistic($adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 return [  
 [  
 'key' => CommonEnum::DISEASE\_LABEL\_MAP[CommonEnum::DISEASE\_LABEL\_HEAD],  
 'total\_1' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_HEAD)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '>=', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 'total\_2' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_HEAD)  
 ->whereHas('document', function (Builder $builder) use($adminIds){  
 $builder->where('birthday', '<', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 ],  
 [  
 'key' => CommonEnum::DISEASE\_LABEL\_MAP[CommonEnum::DISEASE\_LABEL\_HEART],  
 'total\_1' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_HEART)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '>=', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 'total\_2' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_HEART)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '<', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 ],  
 [  
 'key' => CommonEnum::DISEASE\_LABEL\_MAP[CommonEnum::DISEASE\_LABEL\_LUNG],  
 'total\_1' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_LUNG)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '>=', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 'total\_2' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_LUNG)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '<', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 ],  
 [  
 'key' => CommonEnum::DISEASE\_LABEL\_MAP[CommonEnum::DISEASE\_LABEL\_RHEUMATISM],  
 'total\_1' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_RHEUMATISM)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '>=', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 'total\_2' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_RHEUMATISM)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '<', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 ],  
 [  
 'key' => CommonEnum::DISEASE\_LABEL\_MAP[CommonEnum::DISEASE\_LABEL\_BONE],  
 'total\_1' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_BONE)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '>=', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 'total\_2' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_BONE)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '<', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 ],  
 [  
 'key' => CommonEnum::DISEASE\_LABEL\_MAP[CommonEnum::DISEASE\_LABEL\_TRACHEA],  
 'total\_1' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_TRACHEA)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '>=', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 'total\_2' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_TRACHEA)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '<', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 ],  
 [  
 'key' => CommonEnum::DISEASE\_LABEL\_MAP[CommonEnum::DISEASE\_LABEL\_OTHER],  
 'total\_1' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_OTHER)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '>=', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 'total\_2' => DocumentDiseaseLabel::query()  
 ->where('value', CommonEnum::DISEASE\_LABEL\_OTHER)  
 ->whereHas('document', function (Builder $builder)use($adminIds) {  
 $builder->where('birthday', '<', now()->subYears(65)->toDateString())  
 ->whereIn('admin\_id',$adminIds);  
 })  
 ->count(),  
 ],  
 ];  
 }  
  
 public function getResidenceLableStatistic($adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 $res = [];  
 $counts = DocumentResidenceLabel::query()  
 ->select(DB::raw('value,count(\*) as total'))  
 ->whereHas('document',function (Builder $builder)use($adminIds){  
 $builder->whereIn('admin\_id',$adminIds);  
 })  
 ->groupBy('value')  
 ->get()  
 ->pluck('total','value')  
 ->toArray();  
 foreach (array\_keys(CommonEnum::RESIDENCE\_LABEL\_MAP) as $key) {  
 $res[] = [  
 'key' => CommonEnum::RESIDENCE\_LABEL\_MAP[$key],  
 'total' => $counts[$key] ?? 0,  
 ];  
 }  
 return $res;  
 }  
  
 public function getCenterStatistic($adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 return [  
 'total' => [  
 'record\_total' => Document::query()  
 ->whereIn('admin\_id',$adminIds)  
 ->count(),  
 'abnormal\_total' => Document::query()  
 ->whereIn('admin\_id', $adminIds)  
 ->whereHas('device\_abnormals')  
 ->count(),  
 ],  
 'today' => [  
 'record\_total' => DeviceRecord::query()  
 ->whereHas('document', function (Builder $builder) use ($adminIds) {  
 $builder->whereIn('admin\_id', $adminIds);  
 })  
 ->where('created\_at','>=',now()->startOfDay())  
 ->where('created\_at','<=',now()->endOfDay())  
 ->count(),  
 'abnormal\_total' => DeviceAbnormal::query()  
 ->whereHas('document', function (Builder $builder) use ($adminIds) {  
 $builder->whereIn('admin\_id', $adminIds);  
 })  
 ->where('created\_at','>=',now()->startOfDay())  
 ->where('created\_at','<=',now()->endOfDay())  
 ->count(),  
 ],  
 ];  
 }  
  
 public function getTopHospitalsStatistic($adminId,$nums = 4)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 $hospitals = DeviceRecord::query()  
 ->leftJoin('documents','documents.id','=','device\_records.document\_id')  
 ->whereIn('documents.admin\_id',$adminIds)  
 ->leftJoin('admins','admins.id','=','documents.admin\_id')  
 ->leftJoin('hospitals', 'admins.area\_id', '=', 'hospitals.area\_id')  
 ->select(DB::raw('hospitals.id as hospital\_id,hospitals.name as hospital\_name,count(\*) as total, admins.area\_id'))  
 ->groupBy('hospitals.id','hospitals.name')  
 ->orderByDesc('total')  
 ->take($nums)  
 ->get()  
 ->toArray();  
 $res = [];  
 $months = $this->generateMonths(12);  
 foreach ($hospitals as $hospital){  
 $tmp = [  
 'name' => $hospital['hospital\_name'],  
 ];  
 $records = DeviceRecord::query()  
 ->leftJoin('documents','documents.id','=','device\_records.document\_id')  
 ->leftJoin('admins','admins.id','=','documents.admin\_id')  
 ->whereIn('documents.admin\_id',$adminIds)  
 ->where('admins.area\_id', $hospital['area\_id'])  
 ->select(DB::raw('date\_format(device\_records.measured\_at,\'%Y-%m\') as month,count(\*) as total'))  
 ->groupBy('month')  
 ->get()  
 ->pluck('total','month')  
 ->toArray();  
 $data = [];  
 foreach ($months as $month){  
 $data[] = [  
 'date' => $month,  
 'total' => $records[$month] ?? 0,  
 ];  
 }  
 $tmp['data'] = $data;  
 $res[] = $tmp;  
 }  
 return $res;  
 }  
  
 public function getFollowUpStatistic($adminId)  
 {  
 $service = app(AdminService::class);  
 $adminIds = $service->getChildrenAdminIds($adminId,request('area\_id',0));  
 return [  
 'year' => TmpFollowUp::query()  
 ->leftJoin('documents','documents.id','=','tmp\_follow\_ups.document\_id')  
 ->whereIn('admin\_id', $adminIds)  
 ->whereNull('documents.deleted\_at')  
 ->where('follow\_up\_at', '>=', now()->startOfYear())  
 ->where('follow\_up\_at', '<=', now()->endOfYear())  
 ->count(),  
 'last\_month' => TmpFollowUp::query()  
 ->leftJoin('documents','documents.id','=','tmp\_follow\_ups.document\_id')  
 ->whereIn('admin\_id', $adminIds)  
 ->whereNull('documents.deleted\_at')  
 ->where('follow\_up\_at', '>=', now()->subMonths(2)->startOfMonth())  
 ->where('follow\_up\_at', '<=', now()->subMonths(2)->endOfMonth())  
 ->count(),  
 'current\_month' => TmpFollowUp::query()  
 ->leftJoin('documents','documents.id','=','tmp\_follow\_ups.document\_id')  
 ->whereIn('admin\_id', $adminIds)  
 ->whereNull('documents.deleted\_at')  
 ->where('follow\_up\_at', '>=', now()->subMonth()->startOfMonth())  
 ->where('follow\_up\_at', '<=', now()->subMonth()->endOfMonth())  
 ->count(),  
 ];  
 }  
  
 private function generateDays($nums)  
 {  
 $now = now();  
 $res = [];  
 while (count($res) < $nums){  
 $res[] = $now->toDateString();  
 $now = $now->subDay();  
 }  
 return $res;  
 }  
  
 public function generateMonths($nums)  
 {  
 $now = now();  
 $res = [];  
 while (count($res) < $nums){  
 if (!empty($res) && $now->year . '-0' . $now->month === $res[count($res)-1]){  
 $now = $now->subDays(3);  
 }  
 $res[] = $now->year .'-' . ($now->month < 10 ? '0'.$now->month : $now->month);  
 $now->subMonth();  
 }  
 return $res;  
 }  
  
 public function getVideoList($adminId,$page,$pageSize,$extra,$filter)  
 {  
 $videoService = app(VideoService::class);  
 return $videoService->getVideoList($page,$pageSize,$extra,$filter);  
 }  
}