```php
<?php

namespace App\Http\Controllers;

use App\Service\AppUpdate;
use App\Service\v1\ApiReturn;
use Illuminate\Http\Request;

class AppUpdateController extends Controller
{
    /**
     * ios更新
     * @param Request $request
     * @return Object
     */
    public function iosUpdate(Request $request)
    {
        $version = $request->get('version_code','');
        $user_id = $request->get('user_id',0);
        $device_num = $request->get('device_num','');
        return AppUpdate::iosUpdate($version,$user_id,$device_num);
    }
    /**
     * uni更新
     * @param Request $request
     * @return Object
     */
    public function uniUpdate(Request $request)
    {
        $version = $request->get('version','');
        $channel = $request->get('channel','');
        $user_id = $request->get('user_id',0);
        $device_num = $request->get('device_num','');
        return AppUpdate::uniUpdate($version,$channel,$user_id,$device_num);
    }

    //设置模式
    public function setModel(Request $request){
        $user_id = $request->get('user_id',0);
        $device_num = $request->get('device_num','');
        $status = $request->get('status');
        if(!isset($status)){
            return ApiReturn::error('请选择要设置的模式');
        }
        return AppUpdate::setModel($user_id,$device_num,$status);
    }
}

```

```php
<?php


namespace App\Http\Controllers;

```

```
use App\Models\v1\Article;
use App\Models\v1\ArticleCommentOne;
use App\Models\v1\ArticleCommentTwo;
use App\Service\Comment;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class CommentController extends Controller
{

    /**
     * 删除评论
     * @param Request $request
     * @return Object
     */
    public function deleteComment(Request $request)
    {
        $userId = $request->post('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $commentId = $request->post('comment_id', 0);
        if (empty($commentId)) {
            return ApiReturn::error('请选择要删除的评论');
        }
        return Comment::deleteComment($commentId, $userId);


    }

    /**
     * 删除评论回复
     * @param Request $request
     * @return Object
     */
    public function deleteReply(Request $request)
    {
        $userId = $request->post('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $commentReplyId = $request->post('comment_reply_id', 0);
        if (empty($commentReplyId)) {
```

```
            return ApiReturn::error('请选择要删除的评论');
        }
        return Comment::deleteReply($commentReplyId, $userId);
    }


}

```

```php
<?php

namespace App\Http\Controllers;

use App\Models\Protocol;
use App\Service\Comment;
use App\Service\Common;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\CommonFnc;
use Exception;
use Illuminate\Auth\Access\AuthorizationException;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Illuminate\Validation\ValidationException;

class CommonController extends Controller
{
    /**
     * 首页轮播图
     * @param Request $request
     * @return Object
     */
    public function getBannerList(Request $request): object
    {
        try {
            $redisKey = $request->get('type','banner-list');
            $version = $request->get('version_code_num');

            $list = Common::getNewBannerGameList($redisKey,$version);
            return ApiReturn::success($list);
        } catch (Exception $e) {
            return ApiReturn::error();
        }
    }

    /**
     * 平台协议
     * @param Request $request
     * @return Object
     */
    public function getProtocolInfo(Request $request): object
    {
        $id = $request->get('type', 1);
        if (!is_numeric($id) || $id < 1) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        try {
            $info = Protocol::query()
                ->where('id', $id)
```

```php
                ->where('status', 1)
                ->select('title', 'content', 'postscript', 'updated_at')
                ->first();
```

```
            return ApiReturn::success($info);
        } catch (\Exception $e) {
            return ApiReturn::error();
        }
    }
    //添加评论

    /**
     * 发布评论
     * @param Request $request
     * @return Object
     */
    public function addComment(Request $request): object
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $article_id = $request->post('article_id');
        if (empty($article_id)) {
            return ApiReturn::error('请选择要评论的文章');
        }
        $content = $request->post('content');
        if (empty($content)) {
            return ApiReturn::error('请输入要评论的内容');
        }
        $ip_location = $request->post('ip_location');
        return Comment::addComment($user_id, $article_id, $content, $ip_location);

    }

    /**
     * 发布评论回复
     * @param Request $request
     * @return Object
     */
    public function addReply(Request $request): object
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $article_id = $request->post('article_id');
        if (empty($article_id)) {
            return ApiReturn::error('请选择要评论的文章');
        }
        $reply_content = $request->post('reply_content');
```

```
        if (empty($reply_content)) {
            return ApiReturn::error('请输入要回复的内容');
        }
        $article_comment_one_id = $request->post('article_comment_one_id');
        if (empty($article_comment_one_id)) {
            return ApiReturn::error('请选择要回复的评论');
        }
        $by_evaluating_id = $request->post('by_evaluating_id');
        $ip_location = $request->post('ip_location');
        $type = $request->post('type');
        if (empty($type)) {
            return ApiReturn::error('请选择评论类型');
        }
        if(!in_array($type,[1,2])){
            return ApiReturn::error('评论类型错误');
        }
        $data = [
            'user_id' => $user_id,
            'article_id' => $article_id,
            'reply_content' => $reply_content,
            'article_comment_one_id' => $article_comment_one_id,
            'by_evaluating_id' => $by_evaluating_id,
            'ip_location' => $ip_location,
            'type' => $type,

        ];
        return Comment::addReply($data);
    }
    //PC游戏页面配置
    public function getConfig(): object
    {
        try {
            $lang = CommonFnc::lang();
            $list = Common::getConfig($lang);
            return ApiReturn::success($list);
        } catch (Exception $e) {
            return ApiReturn::error();
        }
    }
}

```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
    //星际起源页面
    public function AstralOrigin(){
        return view('AstralOrigin');
    }
}

```

```php
<?php

namespace App\Http\Controllers;

use App\Mail\CodeSender;
use App\Service\Email;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\CommonFnc;
use App\Service\v1\RedisKey;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Facades\Redis;
use Illuminate\Support\Facades\Validator;

class EmailController extends Controller
{
    //发送登录验证码
    public function sendLoginCode(Request $request)
    {
        $lang = $request->header('Lang', 'en-CN');
        $lang = CommonFnc::getLang($lang);
        $email = $request->post('email');
        $validator = Validator::make(['email' => $email], [
            'email' => 'required|email',
        ], [
            'email.required' => '请输入邮箱',
            'email.email' => '邮箱格式不正确'
        ]);
        if ($validator->fails()) {
            // 获取验证失败的消息
            $errors = $validator->errors();
            return ApiReturn::error($errors->first());
        }
        $verifyCode = CommonFnc::randomNumeric();
        if (Redis::exists(RedisKey::EMAIL_LOGIN_CODE . $email)) {
            return ApiReturn::error('请勿重复发送');
        }
        try {
            Redis::setex(RedisKey::EMAIL_LOGIN_CODE . $email, 600, $verifyCode);
            Mail::to($email)->send(new CodeSender($verifyCode, $lang));
            return ApiReturn::success('发送成功');
        }catch (\Exception $e){
            return ApiReturn::error('发送失败'.$e->getMessage());

        }

    }
    //邮箱登录
    public function login(Request $request)
```

```php
    {
        $email = $request->post('email');
        $code = $request->post('code');
        $platform = $request->post('platform', 0);
        $isApp = $request->post('is_app', 1);
        $validator = Validator::make(['email' => $email,'code' => $code], [
            'email' => 'required|email',
            'code' =>'required|numeric|digits:6'
        ], [
            'email.required' => '请输入邮箱',
            'email.email' => '邮箱格式不正确',
            'code.required' => '请输入验证码',
            'code.numeric' => '验证码格式不正确',
            'code.digits' => '验证码格式不正确'
        ]);
        if ($validator->fails()) {
            // 获取验证失败的消息
            $errors = $validator->errors();
            return ApiReturn::error($errors->first());
        }
        if (!Redis::exists(RedisKey::EMAIL_LOGIN_CODE . $email)) {
            //验证码过期重新发送验证码
            return ApiReturn::rdata(ApiCode::VERIFY_CODE_NIL);
        }
        //校验验证码是否正确
        if ($code != Redis::get(RedisKey::EMAIL_LOGIN_CODE . $email)) {
            return ApiReturn::rdata(ApiCode::VERIFY_CODE_ERROR);
        }
        $inviteCode = (string)$request->post('invite_code','');
        return Email::login($email,$platform,$isApp,$inviteCode);


    }
}

```

```
<?php

namespace App\Http\Controllers;

use App\Models\v1\Label;
use App\Service\v1\ApiReturn;
use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;

class LabelController extends Controller
{

    /**
     * 后台发布标签列表
     * @return JsonResponse|Object
     */
    public function getPublishLabel()
    {
        $list =  Label::query()
            ->where('status', '=', 1)
            ->where('type', '=', 1)
            ->where('is_hot', 1)
            ->orWhere('recommend_type', 1)
            ->select('id as label_id','label_name')
            ->get()
            ->toArray();
        return ApiReturn::success(['data'=>$list]);
    }

    /**
     * 后台评价标签列表
     * @return JsonResponse|Object
     */
    public function getEvaluateLabel()
    {
        $list =  Label::query()
            ->where('user_id',0)
            ->where('status','=',1)
            ->where('type','=',2)
            ->select('id','label_name as text')
            ->get()
            ->toArray();
        return ApiReturn::success($list);
    }
    //添加帖子/评价标签
    public function addArticleLabel(Request $request)
    {
        try {
            $labelName = $request->post('label_name');
```

```
            if (empty($labelName)) {
                return ApiReturn::error('请输入标签名称');
            }
            //标签类型
            $type = $request->post('type');
            if (empty($type)) {
                return ApiReturn::error('请选择标签类型');
            }
            if(!in_array($type,[1,2])){
                return ApiReturn::error('请选择正确标签类型');
            }
            $userId = $request->post('user_id');
            return Label::addArticleLabel($userId,$labelName,$type);
        }catch (\Exception $e) {
            return ApiReturn::error($e->getMessage());
        }
    }
}

```

```
<?php

namespace App\Http\Controllers;

use App\Service\Manager;
use App\Service\ReportAndBlock;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use Illuminate\Http\Request;

class ManagerController extends Controller
{
    public function menu(Request $request)
    {
        $userId = $request->get('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return Manager::menu($userId);
    }

    //黑名单列表
    public function blackList(Request $request)
    {
        $userId = $request->get('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $page = $request->get('page', 1);
        $limit = $request->get('limit', 10);
        return Manager::blackList($userId, $page, $limit);
    }

    //权限列表
    public function permissionList(Request $request)
    {
        $userId = $request->get('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return Manager::permissionList($userId);

    }

    //拉黑用户
    public function addUserBlock(Request $request): object
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
```

```
        }
        $block_user_id = $request->post('block_user_id');
        if (empty($block_user_id)) {
            return ApiReturn::error('请选择要拉黑的用户');
        }
        if ($block_user_id == $user_id) {
            return ApiReturn::error('您不能拉黑自己');
        }
        $remark = $request->post('remark');
        return Manager::addUserBlock($user_id, $block_user_id, $remark);
    }

    //取消拉黑用户
    public function cancelUserBlock(Request $request): object
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $block_user_id = $request->post('block_user_id');
        if (empty($block_user_id)) {
            return ApiReturn::error('请选择要取消封禁的用户');
        }
        $remark = $request->post('remark');
        return Manager::cancelUserBlock($user_id, $block_user_id, $remark);
    }

    //操作记录
    public function operationLog(Request $request)
    {
        $userId = $request->get('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $page = $request->get('page', 1);
        $limit = $request->get('limit', 10);
        return Manager::operationLog($userId, $page, $limit);
    }
    //屏蔽文章
    public function blockArticle(Request $request)
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $article_id = $request->post('article_id');
        if (empty($article_id)) {
            return ApiReturn::error('请选择要屏蔽的文章');
        }
        $remark = $request->post('remark');
```

```
        return Manager::blockArticle($user_id, $article_id, $remark);
    }


}

```

```
<?php

namespace App\Http\Controllers;

use App\Service\ReportAndBlock;
```

```php
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class ReportAndBlockController extends Controller
{
    //举报选项
    public function reportOptions()
    {
        return ReportAndBlock::reportOptions();
    }

    /**
     * 举报文章
     * @param Request $request
     * @return Object
     */
    public function addArticleReport(Request $request): object
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $article_id = $request->post('article_id');
        if (empty($article_id)) {
            return ApiReturn::error('请选择要举报的文章');
        }
        $reason = $request->post('reason');
        if (empty($reason)) {
            return ApiReturn::error('请选择举报理由');
        }
        $device_num = $request->post('device_num', '');
        return ReportAndBlock::addArticleReport($user_id, $article_id, $reason, $device_num);
    }

    //拉黑用户
    public function addUserBlock(Request $request): object
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $block_user_id = $request->post('block_user_id');
        if (empty($block_user_id)) {
```

```php
            return ApiReturn::error('请选择要拉黑的用户');
        }
        if ($block_user_id == $user_id) {
            return ApiReturn::error('您不能拉黑自己');
        }
        return ReportAndBlock::addUserBlock($user_id, $block_user_id);
    }

```

```
//拉黑游戏
public function addGameBlock(Request $request): object
{
    $user_id = $request->post('user_id');
    if (empty($user_id)) {
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $block_game_id = $request->post('block_game_id');
    if (empty($block_game_id)) {
        return ApiReturn::error('请选择要拉黑的游戏');
    }
    return ReportAndBlock::addGameBlock($user_id, $block_game_id);
}

//用户黑名单列表
public function userBlockList(Request $request): object
{
    $user_id = $request->post('user_id');
    if (empty($user_id)) {
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $page = $request->post('page', 1);
    $limit = $request->post('limit', 10);
    return ReportAndBlock::userBlockList($user_id, $page, $limit);
}

//取消拉黑用户
public function cancelUserBlock(Request $request): object
{
    $user_id = $request->post('user_id');
    if (empty($user_id)) {
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $block_user_id = $request->post('block_user_id');
    if (empty($block_user_id)) {
        return ApiReturn::error('请选择要取消拉黑的用户');
    }
    return ReportAndBlock::cancelUserBlock($user_id, $block_user_id);
}

//举报评论
public function addCommentReport(Request $request)
```

```
{
    Log::info('举报评论入参', $request->all());
    $user_id = $request->post('user_id');
    if (empty($user_id)) {
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $comment_id = $request->post('comment_id');
    if (empty($comment_id)) {
        return ApiReturn::error('请选择要举报的评论');
    }
    $reason = $request->post('reason');
```

```
        if (empty($reason)) {
            return ApiReturn::error('请选择举报理由');
        }
        $device_num = $request->post('device_num', '');
        return ReportAndBlock::addCommentReport($user_id, $comment_id, $reason, $device_num);
    }

    //举报评论
    public function addCommentReplyReport(Request $request)
    {
        Log::info('举报评论回复入参', $request->all());
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $comment_reply_id = $request->post('comment_reply_id');
        if (empty($comment_reply_id)) {
            return ApiReturn::error('请选择要举报的评论');
        }
        $reason = $request->post('reason');
        if (empty($reason)) {
            return ApiReturn::error('请选择举报理由');
        }
        $device_num = $request->post('device_num', '');
        return ReportAndBlock::addCommentReplyReport($user_id, $comment_reply_id, $reason, $device_num);
    }


}

```

```
<?php

namespace App\Http\Controllers;

use App\Service\ArticleService;
use App\Service\v1\ApiReturn;
use App\Service\v1\Article as V1Article;
use App\Service\v1\Code;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class ReviewController extends Controller
{
    //发表评价
    public function add(Request $request)
    {
        Log::info('添加测评传参',$request->all());
        $game_id = $request->post('game_id');
        if(empty($game_id)){
            return ApiReturn::error('请选择要评价的游戏');
        }
        $content = $request->post('content');
        if(empty($content)){
            return ApiReturn::error('请输入评价内容');
        }
        $score = $request->post('score');
        if(empty($score)){
            return ApiReturn::error('请输入评价分数');
        }
        $images = $request->post('images',[]);
        $labels = $request->post('labels');
        $user_id = $request->post('user_id');
        $ip_location = $request->post('ip_location');
        return ArticleService::addReview($user_id,$game_id,$content,$score,$images,$labels,$ip_location);
    }
}

```

```
<?php

namespace App\Http\Controllers;

use App\Service\Steam;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Code;
use App\Service\v1\CommonFnc;
use Illuminate\Http\Request;
use Illuminate\Support\Str;

class SteamController extends Controller
{
    //绑定Steam账号
    public function bindAccount(Request $request)
    {
        $userId = $request->post('user_id');
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $steamid = $request->input('steamid');
        if(empty($steamid)){
            return ApiReturn::error('请选择要绑定的Steam账号');
        }
        return Steam::bindAccount($userId,$steamid);
    }
    //解绑Steam账号
    public function unbindAccount(Request $request)
    {
        $userId = $request->post('user_id');
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return Steam::unbindAccount($userId);
    }

    //获取Steam用户信息
    public function getUserInfo(Request $request)
    {
        $solo_userid = $request->post('solo_userid');
        if(empty($solo_userid)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $userId = $request->post('user_id');
        return Steam::getUserInfo($solo_userid,$userId);
    }

    //获取Steam好友列表
    public function getFriendList(Request $request)
```

```
    {
        $soloUserId = $request->get('solo_userid');
        if(empty($soloUserId)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $page = $request->input('page',1);
        $limit = $request->input('limit',8);
        $username = $request->input('username','');
        $user_id = $request->get('user_id');
        return Steam::getFriendList($soloUserId,$user_id,$username,$page,$limit);
    }
    //获取 Steam 用户游戏列表
    public function getUserGames(Request $request)
    {
        $soloUserId = $request->get('solo_userid');
        if(empty($soloUserId)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $page = $request->input('page',1);
        $limit = $request->input('limit',8);
        $order = $request->input('order','id:desc');
        $user_id = $request->get('user_id');
        $lang = CommonFnc::lang();
        return Steam::getUserGames($soloUserId,$user_id,$lang,$page,$limit,$order);
    }
    //同步steam账号信息
    public function syncUserInfo(Request $request)
    {
        $userId = $request->post('user_id');
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $syncKey = $request->post('sync_key');
        if(empty($syncKey)){
            $syncKey = Str::uuid();
        }
        return Steam::syncUserInfo($userId,$syncKey);
    }
    public function getUserSyncProgress(Request $request)
    {
        $userId = $request->post('user_id');
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $syncKey = $request->post('sync_key');
        return Steam::getUserSyncProgress($userId,$syncKey);
    }
    //获取Steam用户隐私设置
    public function getUserPrivacy(Request $request)
    {
```

```
        $userId = $request->get('user_id');
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }

        return Steam::getPrivacySettings($userId);
    }
    //设置Steam用户隐私设置
    public function setUserPrivacy(Request $request)
    {
        $userId = $request->get('user_id');
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $privacy = $request->post('privacy');
        if(empty($privacy)){
            return ApiReturn::error('请选择隐私设置');
        }
        return  Steam::setPrivacySettings($userId,$privacy);
    }

    //添加到库
    public function addToLibrary(Request $request)
    {
        $userId = $request->post('user_id');
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $game_id = $request->post('game_id');
        if(empty($game_id)){
            return ApiReturn::error('请选择要添加的Steam游戏');
        }
        return Steam::addToLibrary($userId,$game_id);
    }

}

```

```
<?php

namespace App\Http\Controllers;

use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\User;
use Illuminate\Http\Request;
```

```php
use Illuminate\Support\Facades\Redis;

class UserAttentionController extends Controller
{
    /**
     * 关注作者
     * @param Request $request
     * @return Object
     * @throws \Throwable
     */
    public function addAttentionUser(Request $request): object
    {
        try {
            $authorId = (int)$request->post('author_id',0);
            if(empty($authorId)){
                return ApiReturn::rdata(ApiCode::PARAM_ERROR);
            }
            $userId = $request->post('user_id');
//            if(Redis::exists('user:attention:user:'.$userId)){
//                return ApiReturn::error('操作太频繁');
//            }
//            Redis::set('user:attention:user:'.$userId,1,'PX',300,'NX');
            return User::addAttentionUser($userId,$authorId);
        } catch (\Exception $e) {
            return ApiReturn::error();
        }

    }
}

```

```php
<?php


namespace App\Http\Controllers\api;

use App\Events\JytFirstPublishTaskEvent;
use App\Http\Controllers\Controller;
use App\Models\v1\Article;
use App\Models\v1\Circle;
use App\Models\v1\CollectPlatformUrl;
use App\Models\v1\FanBase;
use App\Models\v1\Game;
use App\Models\v1\LongArticle;
use App\Models\v1\User;
use App\Models\v1\UserAttention;
use App\Models\v1\UserDiamondLog;
use App\Models\v1\UserFanBase;
use App\Models\v1\UserLike;
use App\Models\v1\UserRelevance;
use App\Service\v1\AliContentAudit;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\BaiDuSdk;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\ElasticsearchService;
use App\Service\v1\RedisKey;
use App\Service\v1\SoloCode;
use App\Service\v1\WechatPayService;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;
use stdClass;
use Swoole\Coroutine\Redis;

class IndexController extends Controller
{

    public function index(Request $request)
    {
        try {
            $re = LongArticle::where(['status' => 1])->chunk(300,function ($articles) {
                foreach ($articles as $article) {
                    if($article->type == 3) {
                        $arr = json_decode($article->content,true);
                        if(empty($arr)) continue;
                        $str = '';
                        foreach ($arr as $v) {
                            if($v['type'] == 'text') $str .= strip_tags($v['character_string']);
                        }
```

```
                        $article->text_content = $str;
                    }else{
                        $article->text_content = $article->content;
                    }
                    $article->save();
                }
            });
            return ApiReturn::success();
        }catch (\Throwable $throwable) {
            return ApiReturn::error($throwable->getMessage());
        }

//        $content = $request->get('content');
//        $result = AliContentAudit::textAudit([$content,'草泥马']);
//        if($result) return ApiReturn::success();
//        return ApiReturn::error(ApiCode::ALI_CONTENT_AUDIT['msg'],ApiCode::ALI_CONTENT_AUDIT['code']);
//        $re = \Illuminate\Support\Facades\Redis::get('banner:list');dd($re);
//        $redis = new Redis();
//        $redis->connect(config('redis'))
//        $userId = Redis::get('amway:walls');print_r($userId);
    }


    public function test(Request $request)
    {
$ip = CommonFnc::getRealIP();
```

```
        $address = BaiDuSdk::ipToAddress('113.214.193.98');dump($address);
//      $urls = CollectPlatformUrl::query()->get(['id','user_id','task_id','article_id']);
//      foreach ($urls as $url) {
//          $url->article_id = (int)Article::query()->where(['task_id' => $url->task_id,'user_id' => $url->user_id])->value('id');
//          $url->save();
//      }
//      CollectPlatformUrl::query()->where(['activity_center_platform_id' => 1])->select(['id','url'])->chunk(1000,function
($collectPlatforms) {
//          foreach ($collectPlatforms as $collectPlatform) {
//              if(preg_match("/http[s]?:\/\/[\w.]+[\w\/]*[\w.]*\??[\w=&\+\%]*/is",$collectPlatform->url,$match)) {
//                  try {
//                      $urlPath = parse_url($match[0]);
//                      $collectPlatform->url = $urlPath['scheme'].'://'.$urlPath['host'].$urlPath['path'];
//                      $collectPlatform->save();
//                  }catch (\Throwable $throwable) {
//
//                  }
//              }
//          }
//      });

        //mcn二维码数量统计
//      $userRelevances = UserRelevance::query()->with(['mcn_config'])->where(['check_status' => 1])
//          ->where('user_mcn_config_id','>',0)
//          ->selectRaw("parent_user_id,count(parent_user_id) as num")
```

```
//          ->groupBy('parent_user_id')->get();
//      foreach ($userRelevances as $relevance) {
//          if(isset($relevance->mcn_config->id)) {
//              User\UserMcnConfig::query()->where(['id' => $relevance->mcn_config->id])->update(['num' =>
$relevance->num]);
//          }
////          $relevance->mcn_config()->update(['num' => $relevance->num]);
//      }

        //老用户默认分配一个二维码
//      $userRelevances = UserRelevance::query()->with(['mcn_config','user'])
//          ->where(['check_status' => 1])
//          ->where('user_mcn_config_id','>',0)
//          ->get();
//      foreach ($userRelevances as $relevance) {
//          if(isset($relevance->mcn_config->id) && isset($relevance->user->id) && $relevance->user->user_mcn_config_id
== 0) {
//              $relevance->user()->update(['user_mcn_config_id' => $relevance->mcn_config->id]);
////              $relevance->user_mcn_config_id = $relevance->mcn_config->id;
////              $relevance->save();
//          }
//      }

//      $userRelevances = UserRelevance::query()->with(['user'])
//          ->where(['check_status' => 1])
//          ->where('user_mcn_config_id','>',0)
//          ->get();
//      foreach ($userRelevances as $relevance) {
```

```
//              $relevance->user()->update(['user_mcn_config_id' => $relevance->parent_user_id]);
//              $relevance->user_mcn_config_id = $relevance->parent_user_id;
//              $relevance->save();
//      }
//      $list = DB::table('user_mcn_configs_copy1')->get();
//      $data = [];
//      foreach ($list as $v) {
//          $data[] = [
//              'id' => $v->user_id,
//              'user_id' => $v->user_id,
//              'functional_type' => $v->functional_type,
//              'commission' => $v->commission,
//              'first_publish_commission' => $v->first_publish_commission,
//              'qq_group' => $v->qq_group,
//              'desc' => $v->desc,
//              'tips' => $v->tips,
//              'num' => $v->num,
//              'name' => $v->name,
//              'created_at' => $v->created_at,
//              'updated_at' => $v->updated_at,
//          ];
//      }
//      User\UserMcnConfig::query()->insert($data);
//      $userMcnConfigs = User\UserMcnConfig::query()->get();
```

```
//      foreach ($userMcnConfigs as $userMcnConfig) {
//          $userMcnConfig->id = $userMcnConfig->user_id;
//          $userMcnConfig->save();
//      }

//      $images = $request->post('images');
//      $result = AliContentAudit::imageAudit($images);
//      switch ($result) {
//          case SoloCode::ALI_AUDIT_PASS :
//              $msg = '正常';
//              break;
//          case SoloCode::ALI_AUDIT_REVIEW :
//              $msg = '需要人工审核';
//              break;
//          case SoloCode::ALI_AUDIT_BLOCK :
//              $msg = '违规';
//              break;
//          case SoloCode::ALI_AUDIT_ERROR :
//              $msg = '错误';
//              break;
//      }
//      return ApiReturn::error($msg);
//      $circles = Circle::where('id','<',20)->select('id','circle_name','avatar')->get()->toArray();
//      $es = (new ElasticsearchService('circles'))->paginate(1,20);
//      $es->source(['id','circle_name','avatar','cover','articles_num','attention_num']);
//      $circleList = app('es')->search($es->getParams());
//      $circles = array_column($circleList['hits']['hits'],'_source');
//      $param = [
//          'body' => []
//        ];
//        foreach ($circles as $circle) {
//            array_push($param['body'],['index' => 'articles'],
//                [
//                    'query' => [
//                        'term' => ['circle_id' => $circle['id']]
//                    ],
//                    '_source' => ['id','title','circle_id'],
//                    'from' => 0,
//                    'size' => 2,
//                ]);
//        }
//        $response = app('es')->msearch($param);
//        foreach ($circles as &$circle) {
//            $circle['avatar'] = empty($circle['avatar']) ? '' : Common::concatUrlStr($circle['avatar']);
//            $circle['cover'] = empty($circle['cover']) ? '' : Common::concatUrlStr($circle['cover']);
//            $circle['articles_num'] = Common::disposeAmount($circle['articles_num']);
//            $circle['attention_num'] = Common::disposeAmount($circle['attention_num']);
//            $circle['article'] = [];
//            foreach ($response['responses'] as $v) {
//                foreach ($v['hits']['hits'] as $h) {
```

```
//                    if($circle['id'] == $h['_source']['circle_id']) {
//                        $circle['article'][] = $h['_source'];
//                    }
//                }
//            }
//        }
//        return ApiReturn::success($circles);
//        $jia = UserDiamondLog::where(['type' => 7])->selectRaw("user_id,SUM(diamond) as jia")->groupBy('user_id')->get()->toArray();
//        $jian = UserDiamondLog::whereIn('type',[5,6])->selectRaw("user_id,SUM(diamond) as jian")->groupBy('user_id')->get()->toArray();
//        try {
//            DB::beginTransaction();
//            foreach ($jia as $v) {
//                User::where(['user_id' => $v['user_id']])->increment($v['jia']);
//            }
//            foreach ($jian as $v) {
//                User::where(['user_id' => $v['user_id']])->decrement($v['jian']);
//            }
//            UserDiamondLog::whereIn('type',[5,6,7])->delete();
//            DB::commit();
//        }catch (\Throwable $throwable) {
//            DB::rollBack();
//            return ApiReturn::error($throwable->getMessage());
//        }


        //整理签到reids数据
//        $redisDayKeys = \Illuminate\Support\Facades\Redis::keys(RedisKey::SYSTEM_TASK_USER_SIGN_DAY.'*');
//        $redisTimeKeys = \Illuminate\Support\Facades\Redis::keys(RedisKey::SYSTEM_TASK_USER_SIGN_TIME.'*');
//        $tomorrow = strtotime('tomorrow');
//        $afterTomorrow = strtotime('+ 2 days midnight');
//      \Illuminate\Support\Facades\Redis::pipeline(function ($pipe) use($redisTimeKeys,$redisDayKeys,$tomorrow,$afterTomorrow) {
//          foreach ($redisTimeKeys as $redisTimeKey) {
//              $arr = explode(':',$redisTimeKey);
//              $userId = array_pop($arr);
//              $pipe->expireat(RedisKey::SYSTEM_TASK_USER_SIGN_TIME.$userId,$tomorrow);
//              $pipe->incr(RedisKey::SYSTEM_TASK_USER_SIGN_DAY.$userId);
//          }
//
//          foreach ($redisDayKeys as $redisDayKey) {
//              $arr = explode(':',$redisDayKey);
//              $userId = array_pop($arr);
//              $pipe->expireat(RedisKey::SYSTEM_TASK_USER_SIGN_DAY.$userId,$afterTomorrow);
//          }
//      });

        //达人开粉丝群
//      $users = User::where(['attestation_type' => 2])->select('id','username','avatar')->get();
//      $fanBaseUserId = FanBase::pluck('user_id')->toArray();
//      $fanBaseData = $userIds = [];
//      $time = date('Y-m-d H:i:s');
```

```
//      $userFanBaseTime = time();
//      foreach ($users as $user) {
//          if(!in_array($user->id,$fanBaseUserId)) {
//              $fanBaseData[] = [
//                  'name' => $user->username."粉丝群",
//                  'icon' => $user->avatar,
//                  'user_id' => $user->id,
//                  'user_count' => 1,
//                  'created_at' => $time,
//                  'updated_at' => $time
//              ];
//              $userIds[] = $user->id;
//          }
//      }
//      if(!empty($fanBaseData)) {
//          FanBase::insert($fanBaseData);
//          $userFanBaseData = FanBase::whereIn('user_id',$userIds)->select('user_id','id as fan_base_id')->get()->toArray();
//          foreach ($userFanBaseData as &$u) {
//              $u['is_master'] = 1;
//              $u['created_at'] = $userFanBaseTime;
//              $u['updated_at'] = $userFanBaseTime;
//          }
//          UserFanBase::insert($userFanBaseData);
//      }

        //点赞数据整理
//      $userLikes = UserLike::where(['status' => 1])->groupBy('relevance_id')->selectRaw("relevance_id,COUNT(*) as total")->get();
//      foreach($userLikes as $userLike) {
//          Article::where(['id' => $userLike->relevance_id])->update(['like_num' => $userLike->total]);
//      }
//dump(11);
```

```
//        $likes = Article::groupBy('user_id')->selectRaw("user_id,COUNT(*) as total")->get();
//        foreach($likes as $like) {
//            User::where(['id' => $like->user_id])->update(['like_num' => $like->total]);
//        }
//        for($i = 0; $i < 100000; $i++) {
//            $s1 = '真实体验：萌新开路';
//            $s2 = '真实体验：萌新开';
//            $re = similar_text($s1,$s2,$end);
//        }

//        dump($re);
//        dump($end);
//        dump(levenshtein($s1,$s2));

//        $re = UserAttention::where(['is_attention' =>
1])->groupBy('be_focused_user_id')->selectRaw("be_focused_user_id,count(*) as num")->get()->toArray();
//        foreach ($re as $v) {
//            User::where(['id' => $v['be_focused_user_id']])->update(['fans_num' => $v['num']]);
//        }
//        return ApiReturn::success($re);
```

```
//        $articleId = (int)$request->get('id',0);
//        $num = (int)$request->get('num',1);
//        $re = \App\Service\v1\Article::editArticleVariation($articleId,RedisKey::LIKE_NUM,$num);
//        $re = \App\Service\v1\Article::articleVariations($articleId);
//        return ApiReturn::success($re);
        //游戏名称添加到搜索词库
//        $re = Game::pluck('game_name')->toArray();
//        $data = [];
//        $time = time();
//        foreach ($re as $v) {
//            if(!empty($v)) {
//                $data[] = [
//                    'keyword' => $v,
//                    'created_at' => $time,
//                    'updated_at' => $time,
//                ];
//            }
//        }
//        DB::table('search_keywords')->insert($data);
//        return ApiReturn::success($data);
    }

    /**
     * 内部测试
     */
    public function ctest(Request $request)
    {
        $type = $request->get('type');
        if ($type == 'pay_test') {
            $data = new stdClass();
            $data->order_num = 'test01';
            $data->total_price = 0.01;
            $rs = WechatPayService::appCreateOrder($data);
            dd($rs);
        }
        if ($type == 'test') {
            return ApiReturn::rdata('paramError');
        }
        if ($type == 'paytest') {

            $option = [
                "appid" => "wx9b84cce916bbca85",
                "out_batch_no" => "ts01",
                "batch_name" => "测试",
                "batch_remark" => "测试",
                "total_amount" => 1,
                "total_num" => 1,
                "transfer_detail_list" => [
                    [
                        "out_detail_no" => "ts01",
```

```
                        "transfer_amount" => 1,
                        "transfer_remark" => "测试",
                        "openid" => "oaObN6m9GI7nshnyqJkGeelAasKU",
                        // "user_name" => "小那"
                    ]
                ]];
            $rs = WechatPayService::payToBalace($option);
            Log::info('paytest', $rs);
        } else if ($type == 'jftp') {
            JytFirstPublishTaskEvent::dispatch(41435);
        }
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\v1\ServiceConfig;
use App\Models\v1\TaskTab;
use App\Models\v1\User;
use App\Service\v1\ActivityCenter;
use App\Service\v1\ApiReturn;
use App\Service\v1\Code;
use App\Service\v1\Common;
use App\Service\v1\Task;
use Illuminate\Http\Request;

class ActivityCenterController extends Controller
{

    /**
     * 活动中心筛选页面
     * @return Object
     */
    public function filtrate()
    {
        try {
            $data = ActivityCenter::filtrate();
            return ApiReturn::success($data);
        }catch (\Throwable $throwable) {
            Common::log('活动中心筛选页面错误',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 活动中心任务列表
     * @param Request $request
     * @return Object
     */
    public function getTaskList(Request $request)
    {
        $userId = $request->get('user_id');
//        $taskTabId = $request->get('task_tab_id','');
        $taskTabId = 16;
        $name = (string)$request->get('name','');
        $limit = (int)$request->get('limit',10);
        $page = (int)$request->get('page',1);
        $taskStatus = (int)$request->get('task_status',0);
        $platformId = $request->get('platform_id','');
        $gameTypeId = $request->get('game_type_id','');
        $sort = (int)$request->get('sort',0);
```

```
        // 用户角色调取
        $userRoles = [Code::ROLE_TYPE_MAP['default']]; // 普通权限
        $page = ($page - 1) * $limit;
        try {
            $time = $request->server('REQUEST_TIME');
            $model = \App\Models\v1\Task::query();
//            if($taskTabId > 0) $model->where(['task_tab_id' => $taskTabId]);
//            if(!empty($articleType)) $model->whereIn('type',explode(',',$articleType));
            $model->where('task_status', 1)->where('participation_type', '<>', 1); // 状态为显示 去除写手活动
            $model->orderBy('task_expired', 'asc');
            if($sort > 0) {
                $sort = $sort == 1 ? 'asc' : 'desc';
                $model->orderBy('end_time',$sort);
            }
            // 写手活动筛选
            if ($userId > 0) {
                $userInfo = User::find($userId);
                $roles = $userInfo->user_role()->pluck('user_role_id')->toArray();
                if ($roles) $userRoles = $roles;
                $model->where(function ($q) use ($userId) {
                    $q->where('writer_id', $userId)->orWhere('writer_id', 0);
                });
                if($taskStatus < 3) {
                    $createTime = strtotime($userInfo->create_time);
                    $model->where('start_exp','<=',$userInfo->experience)
                    ->where('end_exp','>=',$userInfo->experience)
                    ->where('register_start_time','<=',$createTime)
                    ->where('register_end_time','>=',$createTime);
            }
        }else{
            $model->where(['is_visitor' => 1,'writer_id' => 0]);
        }
        if(!empty($taskTabId)) {
            $model->whereIn('task_tab_id', explode(',',$taskTabId));
        }
        if(!empty($platformId)) {
            $model->whereHas('platform',function ($query) use ($platformId) {
                $query->whereIn('activity_center_platform_id',explode(',',$platformId));
            });
        }
        if(!empty($gameTypeId)) {
            $model->whereHas('game_type',function ($query) use ($gameTypeId) {
                $query->whereIn('activity_center_game_type_id',explode(',',$gameTypeId));
            });
        }
        if(!empty($name)) $model->where('name','like',"%{$name}%");
        $model->with('platform', 'game');
        switch ($taskStatus) {
            case 0:
                $model->where('start_time','<',$time);
```

```
                $taskEndTime = (int)ServiceConfig::where(['field_name' => 'task_end_time'])->value('value');
                if($taskEndTime > 0) $model->where('end_time','>',$time - $taskEndTime * 24 * 3600);
                $model->where(function ($model) use ($userRoles,$userId) {
                    $model->where(function ($model) use ($userRoles) {
                        $model->where(['role_type' => 1])->whereHas('user_role', function ($q) use ($userRoles) {
                            $q->whereIn('user_role_id', $userRoles);
                        });
                    })->orWhere(function ($model) use ($userRoles) {
                        $model->where(['role_type' => 2])->whereDoesntHave('not_user_role', function ($q) use ($userRoles) {
                            $q->whereIn('user_role_id', $userRoles);
                        });
                    })->orWhere(['role_type' => 0])->orWhere(function ($model) use ($userId) {
                        $model->where(['role_type' => 3])->whereHas('task_user', function ($q) use ($userId) {
                            $q->where(['user_id' => $userId]);
                        });
                    });
                });
                $model->orderBy('is_top','desc')
                    ->orderBy('is_hot','desc')
                    ->orderBy('end_time','desc')
                    ->orderBy('id','desc');

                $list = Task::getTaskList($model,$userId,$page,$limit);
                break;
            case 1:
//                 $model->whereHas('user_role', function ($q) use ($userRoles) {
//                     $q->whereIn('user_role_id', $userRoles);
//                 });
```

```
            $model->where(function ($model) use ($userRoles,$userId) {
                $model->where(function ($model) use ($userRoles) {
                    $model->where(['role_type' => 1])->whereHas('user_role', function ($q) use ($userRoles) {
                        $q->whereIn('user_role_id', $userRoles);
                    });
                })->orWhere(function ($model) use ($userRoles) {
                    $model->where(['role_type' => 2])->whereDoesntHave('not_user_role', function ($q) use ($userRoles) {
                        $q->whereIn('user_role_id', $userRoles);
                    });
                })->orWhere(['role_type' => 0])->orWhere(function ($model) use ($userId) {
                    $model->where(['role_type' => 3])->whereHas('task_user', function ($q) use ($userId) {
                        $q->where(['user_id' => $userId]);
                    });
                });
            });
            $model->orderBy('is_top','desc')
                ->orderBy('is_hot','desc')
                ->orderBy('id','desc');
            $model->where('start_time','<',$time)
                ->where('end_time','>',$time)
                ->whereColumn("astrict_participation_num",">","participation_num");
            $list = Task::getTaskList($model,$userId,$page,$limit);
```

```
            break;
        case 2:
//              $model->whereHas('user_role', function ($q) use ($userRoles) {
//                  $q->whereIn('user_role_id', $userRoles);
//              });
            $model->where(function ($model) use ($userRoles,$userId) {
                $model->where(function ($model) use ($userRoles) {
                    $model->where(['role_type' => 1])->whereHas('user_role', function ($q) use ($userRoles) {
                        $q->whereIn('user_role_id', $userRoles);
                    });
                })->orWhere(function ($model) use ($userRoles) {
                    $model->where(['role_type' => 2])->whereDoesntHave('not_user_role', function ($q) use ($userRoles) {
                        $q->whereIn('user_role_id', $userRoles);
                    });
                })->orWhere(['role_type' => 0])->orWhere(function ($model) use ($userId) {
                    $model->where(['role_type' => 3])->whereHas('task_user', function ($q) use ($userId) {
                        $q->where(['user_id' => $userId]);
                    });
                });
            });
            $model->orderBy('is_top','desc')
                ->orderBy('is_hot','desc')
                ->orderBy('id','desc');
            $list = Task::getTaskListEnd($model,$userId,$page,$limit);
            break;
        case 3:
            if ($userId == 0) return ApiReturn::success([]); // 已参与
            $model->where('start_time','<',$time);
            $model->orderBy('is_top','desc')
                ->orderBy('is_hot','desc');

                $list = Task::getTaskListParticipated($model,$userId,$page,$limit);
                break;
            }
            // 详情 h5 链接
            $h5 = env('WEB_H5_URL');
            foreach ($list as &$v) {
                $v['h5_url'] = rtrim($h5, '/').'/taskdetail/'.$v['id'];
            }
            return ApiReturn::success($list);
        }catch (\Throwable $throwable) {
            Common::log('活动中心任务列表错误',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 活动中心tab列表
     * @return Object
```

```
     */
    public function getTaskTabList()
    {
        try {
            $taskTabs = TaskTab::select('id','name')->whereIn('type', [0,5])->orderBy('sort')->get()->toArray();
            array_unshift($taskTabs,['id' => 0,'name' => '全部']);
            return ApiReturn::success($taskTabs);
        } catch (\Throwable $throwable) {
            Common::log('活动中心tab列表错误',$throwable);
            return ApiReturn::error();
        }

    }


    /**
     * 邀请页活动展示
     * @param Request $request
     * @return Object
     */
    public function getInviteTaskList(Request $request)
    {
        $tab = (int)$request->get('tab',0); // 0 默认 返利活动 1 鉴游团活动
        $limit = (int)$request->get('limit',50);
        $page = (int)$request->get('page',1);
        $page = ($page - 1) * $limit;
        // 用户角色调取
        try {
            $time = $request->server('REQUEST_TIME'); // 脚本开始执行时间
            $model = \App\Models\v1\Task::query();
            $model->where('task_status', 1); // 状态为显示
            $model->orderBy('end_time', 'desc');
            // 写手活动筛选
            $model->where(['writer_id' => 0]);
            $model->with('platform', 'game');
            if ($tab == 1) {
                $model->where('invite_reward_switch', 0);
                $model->where(['role_type' => 1]);
                $model->whereHas('jyt_user_role');
            } else $model->where('invite_reward_switch', 1);
            $model->where('start_time','<',$time)
                ->where('end_time','>',$time);
            $model->orderBy('is_top','desc')
                ->orderBy('is_hot','desc')
                ->orderBy('id','desc');
            $model->where('start_time','<',$time)
                ->where('end_time','>',$time)
                ->whereColumn("astrict_participation_num",">","participation_num");
            $list = Task::getTaskList($model,0,$page,$limit,'invite_reward_switch,invite_reward_diamond');

```

```
            // 详情 h5 链接
            $h5 = env('WEB_H5_URL');
            foreach ($list as &$v) {
                $v['h5_url'] = rtrim($h5, '/').'/taskdetail/'.$v['id'];
            }
            return ApiReturn::success($list);
        }catch (\Throwable $throwable) {
            Common::log('邀请页活动展示列表错误',$throwable);
            return ApiReturn::error();
        }
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;
use App\Http\Controllers\Controller;
use App\Service\v1\Ad;
use App\Service\v1\ApiReturn;
use Illuminate\Http\Request;


class AdController extends Controller
{
    //获取多个广告
    public function getByIds(Request $request): object
    {
        $ids = $request->get('ids',[1]);
        return Ad::getByIds($ids);
    }
    //获取单个广告
    public function getById(Request $request): object
    {
        $id = $request->get('id',1);
        return Ad::getAdById($id);
    }
    //上传广告用户行为数据
    public function addUserAction(Request $request): object
    {
        $param = $request->post();
        if(empty($param['ad_id'])){
            return ApiReturn::error("请上传广告 ID");
        }
        if(empty($param['source_type'])){
            return ApiReturn::error("请上传用户源类型");
        }
        if(empty($param['source_type'])){
            return ApiReturn::error("请上传用户行为源类型");
        }
        return Ad::addUserAction($param);
    }
    //首页广告
    public function getHomeAd(): object
    {
        return Ad::getHomeAd();
    }
    //个人中心广告
    public function getUserCenterAd(): object
    {
        return Ad::getUserCenterAd();
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Service\v1\AddressService;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use Illuminate\Http\Request;

class AddressController extends Controller
{
    /**
     * 全球地址列表
     * @param Request $request
     * @return Object
     */
    public function getAddress(Request $request)
    {
        try {
            $pid = $request->get('pid',0);
            $list = AddressService::getAddress((int)$pid);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('获取全球地址',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 积分商城地址（国内地址）
     * @param Request $request
     * @return Object
     */
    public function cnAddress(Request $request)
    {
        try {
            $pid = intval($request->get('pid',0));
            $list = $pid > 0 ? AddressService::getSetParentIdToAddress($pid) : AddressService::getSetProvinceAddress();
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('获取国内地址',$throwable);
            return ApiReturn::error();
        }
    }
}

```

```
<?php
declare(strict_types=1);

namespace App\Http\Controllers\api\v1;

use App\Events\BrowseArticleTaskEvent;
use App\Events\FirstPublishArticleEvent;
use App\Events\FirstPublishTaskEvent;
use App\Events\JytFirstPublishTaskEvent;
use App\Events\ShareArticleTaskEvent;
use App\Http\Controllers\Controller;
use App\Http\Requests\ArticleRequest;
use App\Http\Requests\GameRequest;
use App\Jobs\ErrorLog;
use App\Jobs\ReadArticle;
use App\Models\CircleAttention;
use App\Models\v1\Article;
use App\Models\v1\Circle;
use App\Models\v1\Game;
use App\Models\v1\Task as V1Task;
use App\Models\v1\User as V1User;
use App\Service\v1\AliContentAudit;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Article as V1Article;
use App\Service\v1\ArticleService;
use App\Service\v1\Code;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\LongArticleTask;
use App\Service\v1\Search;
use App\Service\v1\task\PublishDynamicTask;
use App\Service\v1\task\PublishVideoTask;
use App\Service\v1\SoloCode;
use App\Service\v1\User;
use Hhxsv5\LaravelS\Swoole\Task\Task;
use Illuminate\Auth\Access\AuthorizationException;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Route;
use Illuminate\Validation\ValidationException;

class ArticleController extends Controller
{

    /**
     * 发布长文
     * @param Request $request
     * @return Object
     */
```

```
    public function publishLongArticle(Request $request)
    {
        try {
            $isSource = intval($request->post('is_source',1));
            $param = $request->post();
            if($isSource == 3 && empty($param['circle_id'])) {
                return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            }else if($isSource == 2 && (empty($param['task_id']) || empty($param['circle_id']))) {
                return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            }
            $userInfo = V1User::find($param['user_id']);
            // $userInfo = (new \App\Models\v1\User())->getUserInfo($param['user_id'],['id','mobile','attestation_type','is_writer']);
            if (empty($userInfo->mobile)) return ApiReturn::error(ApiCode::NOT_BOUND_ERROR['msg'],
ApiCode::NOT_BOUND_ERROR['code']);
            if (isset($param['task_id']) && $param['task_id'] > 0) {
                $articleCount = User::isParticipation($param['user_id'],$param['task_id']);
                if($articleCount > 0) return ApiReturn::error(ApiCode::PARTICIPATION_ERROR['msg'],
ApiCode::PARTICIPATION_ERROR['code']);
                if (!User::isTaskEnroll($param['user_id'], $param['task_id'])) return
ApiReturn::error(ApiCode::TASK_ENROLL_ERROR['msg'], ApiCode::TASK_ENROLL_ERROR['code']); // 报名检测

                // 权限检测
                $task = V1Task::find($param['task_id']);
                $param['game_id'] = $task['game_id'];
                $circle = $task->circle;
                $param['circle_id'] = $circle ? $circle['id'] : 0;
                $userRole = $userInfo->user_role()->pluck('user_role_id')->toArray();
                if (!$userRole) $userRole = [Code::ROLE_TYPE_MAP['default']]; // 给个默认防止bug
                $taskRole = $task->user_role()->pluck('user_role_id')->toArray();
                if ($taskRole) {
                    if ($task['participation_type'] == 1) { // 写手标记
                        if (!in_array(Code::ROLE_TYPE_MAP['writer'], $userRole)) return
ApiReturn::rdata(ApiCode::TASK_PERMISSION_DENIED_ERROR);
                        return ApiReturn::rdata(ApiCode::TASK_WRITER_PLATFORM_ERROR);
                    }
                    if (!array_intersect($taskRole, $userRole)) return ApiReturn::rdata(ApiCode::TASK_PERMISSION_DENIED_ERROR); // 无交集无权限
                } else return ApiReturn::rdata(ApiCode::TASK_PERMISSION_DENIED_ERROR);
            } else if($param['circle_id'] > 0) {
                $param['game_id'] = Circle::where(['id' => $param['circle_id']])->value('game_id') ?: 0;
            }else{
                $param['game_id'] = 0;
            }
            $param['type'] = 3;
            $param['resource'] = is_string($param['long_text_content']) ? json_decode($param['long_text_content'],true) : $param['long_text_content'];
            unset($param['long_text_content']);
            $param['text_content'] = $param['video'] = '';
            $param['images'] = $param['video'] = $auditImages = [];
            $param['check_status'] = 2;
            $param['cover'] = empty($param['cover']) ? '' : parse_url($param['cover'],PHP_URL_PATH);
            foreach ($param['resource'] as &$v) {
                switch ($v['type']) {
                    case 'text':
                        $param['text_content'] .= empty($v['character_string']) ? '' : strip_tags($v['character_string']);
                        break;
```

```
                case 'image':
                    array_push($auditImages,$v['image']);
                    array_push($param['images'],parse_url($v['image'],PHP_URL_PATH));
                    break;
                case 'video':
                    $param['video'] = parse_url($v['url'],PHP_URL_PATH);
                    $param['check_status'] = 1;
                    break;
                case 'game':
                    $gameInfo = Game::with('game_labels')->where(['id' => $v['game_id']])->first();
                    if(empty($gameInfo)) {
                        $v['game_id'] = 0;
                        $v['game_name'] = '';
                        $v['game_icon'] = '';
                        $v['game_label'] = [];
                        $v['game_score'] = '';
                    }else{
                        $gameInfo = $gameInfo->toArray();
                        $v['game_id'] = $gameInfo['id'];
                        $v['game_name'] = $gameInfo['game_name'];
                        $v['game_icon'] = $gameInfo['game_icon'];
                        Search::scoreCount($gameInfo);
                        $v['game_score'] = $gameInfo['score'];
                        $gameLabels = array_slice($gameInfo['game_labels'],0,3);
                        foreach ($gameLabels as $gl) {
                            $v['game_label'][] = [
                                'label_id' => $gl['id'],
                                'label_name' => $gl['label_name']
                            ];
                        }
                    }
                    break;
                    case 'link':
                        $v['app_url'] = ArticleService::linkToArticleUrl($v['url']);
                        break;
                }
            }
            if(mb_strlen($param['text_content']) > 10000) return ApiReturn::error(ApiCode::LONG_ARTICLE_TEXT['msg'],ApiCode::LONG_ARTICLE_TEXT['code']);
            if(count($param['images']) > 50) return ApiReturn::error(ApiCode::LONG_ARTICLE_IMAGE['msg'],ApiCode::LONG_ARTICLE_IMAGE['code']);
            // 内容截取
            $param['abstract'] = mb_substr($param['text_content'], 0, Code::ARTICLE_LIST_CONTENT_LIMIT);
            $contentStatus = AliContentAudit::textAudit([$param['title'].$param['text_content']]);
            if($contentStatus === SoloCode::ALI_AUDIT_BLOCK) {
                return ApiReturn::error(ApiCode::ALI_CONTENT_AUDIT['msg'],ApiCode::ALI_CONTENT_AUDIT['code']);
            }
            if($param['check_status'] == 2){
                $param['check_status'] = $contentStatus == 1 ? 2 : 1;
            }
//          $auditImages = $param['images'];
            if(isset($param['cover']) && !in_array($param['cover'],$auditImages)) array_push($auditImages,$param['cover']);
            foreach ($auditImages as $image) {
                $imageStatus = AliContentAudit::imageAudit($image);
                if($imageStatus === SoloCode::ALI_AUDIT_BLOCK) return ApiReturn::error(ApiCode::ALI_IMAGE_AUDIT['msg'],ApiCode::ALI_IMAGE_AUDIT['code']);
                if($param['check_status'] == 2){
                    $param['check_status'] = $imageStatus == 1 ? 2 : 1;
                }
            }
            $task = new LongArticleTask($param,$userInfo->attestation_type);
            $result = Task::deliver($task);
            if($result) {
                FirstPublishTaskEvent::dispatch($param['user_id'],SoloCode::SYSTEM_TASK_FIRST_PUBLISH);
                FirstPublishArticleEvent::dispatch($param['user_id']);
                if(isset($param['task_id']) && $param['task_id'] > 0) JytFirstPublishTaskEvent::dispatch($param['user_id']);
                return ApiReturn::success((object)[]);
            }
            return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
        }catch (\Throwable $throwable) {
            Common::log('发布帖子',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 发布视频
     * @param Request $request
     * @return Object
     */
    public function publishVideoOld(Request $request)
    {
        try {
            $isSource = intval($request->post('is_source', 1));
            $title = (string)$request->post('title', '');
            $cover = $request->post('cover', '');
            $video = $request->post('video', '');
            $circleId = (int)$request->post('circle_id', 0);
            $taskId = intval($request->post('task_id', 0));
            $checkStatus = (empty($cover) || empty($video));
            if (($isSource == 1 || $isSource == 3) && $checkStatus) {
                return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            } else if ($checkStatus || ($isSource == 2 && $taskId <= 0)) {
                return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            }
            if($taskId > 0 && $circleId <= 0) {
                return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            }
            $userId = intval($request->post('user_id'));
            $userInfo = (new
\App\Models\v1\User())->getUserInfo($userId,['id','mobile','attestation_type','is_writer','create_time','experience']);
            if($taskId > 0) {
                $taskInfo = \App\Models\v1\Task::where(['id' =>
$taskId])->select('register_start_time','register_end_time','start_exp','end_exp')->first();
```

```
            if(empty($taskInfo)) ApiReturn::error(ApiCode::TASK_ERROR['msg'], ApiCode::TASK_ERROR['code']);
            if($taskInfo->register_start_time > strtotime((string)$userInfo->create_time) || $taskInfo->register_end_time <
strtotime((string)$userInfo->create_time) || $taskInfo->start_exp > $userInfo->experience || $taskInfo->end_exp <
$userInfo->experience) {
                return ApiReturn::error(ApiCode::ROLE_ERROR['msg'], ApiCode::ROLE_ERROR['code']);
            }
        }
        $param = $request->post();
        $param['circle_id'] = $circleId;
        $param['label_list'] = $param['label_list'] ?? [];
        $param['task_id'] = $taskId;
        $param['type'] = 1;
        $param['title'] = $title;
        $param['cover'] = empty($cover) ? '' : parse_url($cover,PHP_URL_PATH);
        $param['video'] = empty($video) ? '' : parse_url($video,PHP_URL_PATH);
        if (empty($userInfo->mobile)) return ApiReturn::error(ApiCode::NOT_BOUND_ERROR['msg'],
ApiCode::NOT_BOUND_ERROR['code']);
        if ($taskId > 0) {
            $articleCount = User::isParticipation($userId,$taskId);
            if($articleCount > 0) return ApiReturn::error(ApiCode::PARTICIPATION_ERROR['msg'],
ApiCode::PARTICIPATION_ERROR['code']);
            if (!User::isTaskEnroll($param['user_id'], $param['task_id'])) return
ApiReturn::error(ApiCode::TASK_ENROLL_ERROR['msg'], ApiCode::TASK_ENROLL_ERROR['code']); // 报名检测
            if (in_array($userInfo['is_writer'], [0, 5])) $param['is_writer_task'] = $userInfo['is_writer'];
        }
        $content = $title;
        if(isset($param['video_content']) && !empty($param['video_content'])) {
            $content .= $param['video_content'];
            $param['content'] = $param['video_content'];
            $param['content_len'] = mb_strlen($param['content']);
        }else {
            $param['content'] = '';
        }
```

```
            unset($param['video_content']);
            if(mb_strlen($param['content']) > 10000) return ApiReturn::error(ApiCode::LONG_ARTICLE_TEXT['msg'],ApiCode::LONG_ARTICLE_TEXT['code']);
            $contentStatus = AliContentAudit::textAudit([$content]);
            if($contentStatus === SoloCode::ALI_AUDIT_BLOCK) {
                return ApiReturn::error(ApiCode::ALI_CONTENT_AUDIT['msg'],ApiCode::ALI_CONTENT_AUDIT['code']);
            }
            $imageStatus = AliContentAudit::imageAudit($cover);
            if($imageStatus === SoloCode::ALI_AUDIT_BLOCK) return ApiReturn::error(ApiCode::ALI_IMAGE_AUDIT['msg'],ApiCode::ALI_IMAGE_AUDIT['code']);
            $task = new PublishVideoTask($param,$userInfo->attestation_type);
            $result = Task::deliver($task);
            if($result) {
                FirstPublishTaskEvent::dispatch($param['user_id'],SoloCode::SYSTEM_TASK_FIRST_PUBLISH);
                return ApiReturn::success((object)[]);
            }
            return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
        }catch (\Throwable $throwable) {
            Common::log('发布视频',$throwable);
            return ApiReturn::error();
        }
    }


```

```
    /**
     * showdoc
     * @catalog 发布|修改 动态
     * @title 修改动态
     * @description 修改动态的接口
     * @method POST
     * @url /api/publishDynamic
     * @param GameRequest $request
     * @param int $action 选填 动作 1(默认) 添加 2 修改
     * @param int $artcile_id 选填 文章id action 2 必填
     * @param string $dynamic_content 选填 内容
     * @return Object {"code":"200","msg":"ok","data":[]}
     */
    public function publishDynamic(Request $request)
    {
        try {
            $param  = $request->post();
            $param['action'] = $request->post('action', Code::ACTION_ADD);
            $param['type'] = Code::ARTICLE_TYPE_MAP['dynamic'];
            // 验证 设置
            Log::info('发布动态入口参数-----',$param);
            $err = (array)V1Article::checkArticleParam($param);
            if($err) return ApiReturn::rdata($err);
            // 插入到 article 表的数据
            $param['data'] = [
                'cover'        => $param['cover'],
                'check_status'  => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
                'content'       => $param['content'],
                // article 标题 截取
                'title'         => $param['title'] ?? '',
                'text_content'  => $param['text_content'],
                'content_len'   => $param['content_len'],
                'home_page'     => $param['home_page'],
                'label_list'    => $param['label_list'],
                'abstract'      => $param['abstract'],
                'user_id'       => $param['user_id'],
            ];
            if(!empty($data['title'])){
                if(!v1Article::checkContentBySensitiveWord($data['title'])){
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
                }
            }
            // 敏感词检测-内容
            if(!empty($data['content'])){
                if(!v1Article::checkContentBySensitiveWord($data['content'])){
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
                }
            }
            if(!empty($data['text_content'])){
                if(!v1Article::checkContentBySensitiveWord($data['text_content'])){
```

```
                $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
            }
        }
        if ($param['action'] == Code::ACTION_ADD) {
            // 新增需要的数据 (修改的时候不可更改的数据)
            $param['data'] = array_merge($param['data'],[
                'game_id'       => $param['game_id'],
                'task_id'       => $param['task_id'],
                'circle_id'     => $param['circle_id'],
                'type'          => $param['type'],
                'is_writer_task'=> $param['is_writer_task'],
                'ip_location' => $param['ip_location'] ?? '',
            ]);
            if(isset($param['created_at'])){
                $param['data']['create_time'] =  strtotime($param['created_at']);
                $param['data']['update_time'] =  strtotime($param['created_at']);
            }
        }
        $res = V1Article::addArticle($param);
        return $res;
    }catch (\Throwable $throwable) {
        Common::log('发布动态',$throwable);
        return ApiReturn::error();
    }
}

/**
 * showdoc
 * @catalog 发布|修改 动态
 * @title 修改动态
 * @description 修改动态的接口
 * @method POST
```

```
     * @url /api/publishVideo
     * @param GameRequest $request
     * @param int $action 选填 动作 1(默认) 添加 2 修改
     * @param int $artcile_id 选填 文章id action 2 必填
     * @param string $video_content 选填 内容
     * @return Object {"code":"200","msg":"ok","data":[]}
     */
    public function publishVideo(Request $request)
    {
        try {
            $param  = $request->post();
            Log::info('发布视频入口参数',$param);
            $param['action'] = $request->post('action', Code::ACTION_ADD);
            $param['type'] = Code::ARTICLE_TYPE_MAP['video'];
            // 验证 设置
            $err = (array)V1Article::checkArticleParam($param);
            if($err) return ApiReturn::rdata($err);
            // 插入到 article 表的数据
```

```
            $param['data'] = [
                'cover'         => $param['cover'],
                'check_status'  => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
                'content'       => $param['content'],
                // article 标题 截取
                'title'         =>  $param['title'] ?? '',
                'text_content'  => $param['text_content'],
                'content_len'   => $param['content_len'],
                'home_page'     => $param['home_page'],
                'label_list'    => $param['label_list'],
                'abstract'      => $param['abstract'],
                'user_id'       => $param['user_id'],
            ];
            if(!empty($data['title'])){
                if(!v1Article::checkContentBySensitiveWord($data['title'])){
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
                }
            }
            // 敏感词检测-内容
            if(!empty($data['content'])){
                if(!v1Article::checkContentBySensitiveWord($data['content'])){
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
                }
            }
            if(!empty($data['text_content'])){
                if(!v1Article::checkContentBySensitiveWord($data['text_content'])){
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
                }
            }
            if ($param['action'] == Code::ACTION_ADD) {
                // 新增需要的数据 (修改的时候不可更改的数据)
                $param['data'] = array_merge($param['data'],[
                    'game_id'       => $param['game_id'],
                    'task_id'       => $param['task_id'],
                    'circle_id'     => $param['circle_id'],
                    'type'          => $param['type'],
                    'is_writer_task'=> $param['is_writer_task'],
                    'ip_location' => $param['ip_location'] ?? ''
                ]);
            }
            return V1Article::addArticle($param);
        }catch (\Throwable $throwable) {
            Common::log('发布视频',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 分享文章
     * @param Request $request
```

```
 * @return Object
 */
public function shareArticle(Request $request)
{
    try {
        $articleId = intval($request->post('article_id',0));
        if($articleId >= 0) {
//              $articleType = intval(Article::where(['id' => $articleId,'check_status' => 2])->value('type'));
//              if(in_array($articleType,[1,3])) {
                $userId = $request->post('user_id');
                ShareArticleTaskEvent::dispatch($userId,$articleId);
//              }
        }
        return ApiReturn::success();
    }catch (\Throwable $throwable) {
        Common::log('任务中心分享文章记录',$throwable);
        return ApiReturn::error();
    }
}

/**
 * 社区推荐文章列表
 * @param Request $request
 * @return Object
 */
public function getCommunityRecommendArticleList(Request $request)
{
    try {
        $userId = $request->get('user_id');
        $platform = (int)$request->get('platform');
        $page = (int)$request->get('page',1);
        $limit = (int)$request->get('limit',10);
        $device_id = $request->get('device_num');
        $response = V1Article::getCommunityRecommendArticleList($userId,$page,$limit,$platform,$device_id);
        return response()->json([
            'code' => 200,
            'msg' => "OK",
            'data' => $response['articles'],
            'total_count' =>$response['total_count']
        ]);
    }catch (\Throwable $throwable) {
        Common::log('社区推荐文章列表',$throwable);
        return ApiReturn::error();
    }
}

/**
 * showdoc
 * @catalog 发布|修改 游戏评价
 * @title 发布|修改评价
```

```
 * @description 发布|修改评价的接口
 * @method POST
 * @url /api/article/publishGameEvaluate
 * @param GameRequest $request
 * @return Object
 * @throws AuthorizationException
 * @throws ValidationException
 */
public function publishGameEvaluate(GameRequest $request): object
{
    $request->validate('publishGameEvaluate');
    try {
        $param  = $request->post();
        $param['action'] = $request->post('action', Code::ACTION_ADD);
        $param['type'] = Code::ARTICLE_TYPE_MAP['evaluate'];
        // 验证 设置
        $err = (array)V1Article::checkGameArticleParam($param);
        if($err) return ApiReturn::rdata($err);

        // 插入到 article 表的数据
        $param['data'] = [
            'cover'         => $param['cover'],
            'game_score'    => $param['game_score'],
            'check_status'  => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
            'content'       => $param['content'],
            // article 标题 截取
            'title'         => $param['title'] ?? '',
            'text_content'  => $param['text_content'],
            'content_len'   => $param['content_len'],
            'home_page'     => $param['home_page'],
            'label_list'    => $param['label_list'],
            'abstract'      => $param['abstract'],
            'user_id'       => $param['user_id'],
        ];
        if(!empty($data['title'])){
            if(!v1Article::checkContentBySensitiveWord($data['title'])){
                $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
            }
        }
        // 敏感词检测-内容
        if(!empty($data['content'])){
            if(!v1Article::checkContentBySensitiveWord($data['content'])){
                $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
            }
        }
        if(!empty($data['text_content'])){
            if(!v1Article::checkContentBySensitiveWord($data['text_content'])){
                $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
            }
        }
```

```
            if ($param['action'] == Code::ACTION_ADD) {
                // 新增需要的数据 (修改的时候不可更改的数据)
                $param['data'] = array_merge($param['data'],[
                    'game_id'       => $param['game_id'],
                    'task_id'       => $param['task_id'],
                    'circle_id'     => $param['circle_id'],
                    'type'          => $param['type'],
                    'is_writer_task'=> $param['is_writer_task'],
                ]);
            }
            return V1Article::addArticle($param);
        }catch (\Throwable $throwable) {
            Common::log('发布动态',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 发布游戏评测
     * @param GameRequest $request
     * @param int $action 1添加 2 修改
     * @return Object
     * @throws \Illuminate\Auth\Access\AuthorizationException
     * @throws \Illuminate\Validation\ValidationException
     * @throws \Throwable
     */
    public function publishGameEvaluating(GameRequest $request)
    {
        $request->validate('publishGameEvaluating');
        $action = $request->post('action',1);
        $param = $request->post();
        $param['type'] = 5;
        $param['action'] = $action;
        $param['resource'] = is_string($param['resource']) ? json_decode($param['resource'],true) : $param['resource']; // ios 传过string

        // 验证 设置
        $err = V1Article::checkGameArticleParam($param);
        if($err) return ApiReturn::error($err);
        // 插入到 article 表的数据
        $param['data'] = [
            'cover'           => $param['cover'],
            'game_score'      => $param['game_score'],
            'image_score'     => $param['image_score'],
            'experience_score'  => $param['experience_score'],
            'playing_score'     => $param['playing_score'],
            'check_status'      => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
            'content'           => $param['content'],
            // article 标题 截取
            'title'             => '',
            'text_content'      => $param['text_content'],
```

```
            'content_len'       => $param['content_len'],
            'home_page'         => $param['home_page'],
            'label_list'        => $param['label_list'],
            'abstract'          => $param['abstract'],
            'spend'             => $param['spend'],
            'user_id'           => $param['user_id'],
        ];
        if(!empty($data['title'])){
            if(!v1Article::checkContentBySensitiveWord($data['title'])){
                $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
            }
        }
        // 敏感词检测-内容
        if(!empty($data['content'])){
            if(!v1Article::checkContentBySensitiveWord($data['content'])){
                $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
            }
        }
        if(!empty($data['text_content'])){
            if(!v1Article::checkContentBySensitiveWord($data['text_content'])){
                $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
            }
        }
        if ($param['action'] == Code::ACTION_ADD) {
            // 新增需要的数据 (修改的时候不可更改的数据)
            $param['data'] = array_merge($param['data'],[
                'game_id'       => $param['game_id'],
                'task_id'       => $param['task_id'],
                'circle_id'     => $param['circle_id'],
                'type'          => $param['type'],
                'is_writer_task'=> $param['is_writer_task'],
            ]);
        }

        return V1Article::addArticle($param);
    }


    /**
     * showdoc
     * @catalog 个人稿件管理
     * @title 个人稿件管理
     * @description 个人稿件管理的接口
     * @method POST
     * @url /api/article/publishGameEvaluate
     * @param ArticleRequest $request
     * @return Object
     * @throws AuthorizationException
     * @throws ValidationException
     */
```

```
    public function personalArticleList(ArticleRequest $request): object
    {
        $request->validate('personalArticleList');
        try {
            $limit          = $request->get('limit',10);
            $checkStatus    = $request->get('check_status',0);
            $uid            = $request->get('user_id');
            $where          = ['user_id' => $uid];
            $keyword        = $request->get('keyword');
            if($keyword){
                array_push($where,['text_content','like',"%{$keyword}%"]);
                \App\Service\v2\UserSearchLog::addUserSearchLog($keyword, $uid);
            }
            if ($checkStatus == 4) {
                $checkStatus = [1,3];
            } else if ($checkStatus != 0){
                $checkStatus = [$checkStatus];
            }
            $list = (new Article())->getUserAllArticleLists($where, $checkStatus,$limit);
            Article::handleResorce($list);
            foreach (Code::ARTICLE_TYPE as $v) {
                Comment::disposeLike($uid,$list['data'],$v);
            }
            return ApiReturn::success($list);
        }catch (\Exception $e) {

ErrorLog::dispatch(\Route::current()->getActionName(),CommonFnc::getRealIP(),json_encode($e),['file'=>$e->getFile(),'line'=>$e->getLine(),'code'=>$e->getCode(),'message'=>$e->getMessage()],time())->onQueue('error_log');
            return ApiReturn::error();
        }
    }

    /**
     * 圈子文章列
     * @param ArticleRequest $request
     * @return Object
     */
    public function getCircleArticleList(ArticleRequest $request): object
    {
        Log::info('圈子文章列表入参',$request->all());
        try {
            $limit = $request->get('limit',10);
            $order = $request->get('order',1);
            $circle_id = $request->get('circle_id');
            if(empty($circle_id)){
                return ApiReturn::rdata(ApiCode::PARAM_ERROR);
            }
            $uid = $request->get('user_id');
            $article_type = $request->get('article_type') ? $request->get('article_type') : 1;
            $elite = 0;
            $type = [];
            switch ($article_type){
```

```
                case 1:
                    break;
                case 2:
                    $elite = 1;
                    break;
                case 3:
                    $type = [2,3];
                    break;
                case 4:
                    $type = [1];
                    break;
                default:
                    return ApiReturn::rdata(ApiCode::PARAM_ERROR);
            }
            $device_id = $request->get('device_num');
            $list = V1Article::getCircleArticleLists($circle_id,$elite,$limit,$type,$uid,$order,$device_id);
            if($uid){
                $rs = CircleAttention::query()->where(['user_id' => $uid, 'circle_id' => $circle_id])->first();
                $role_type = $rs ? $rs['role_type'] : 1;
                foreach ($list['data'] as &$v) {
                    $v['role_type'] = $role_type;
                }
            }
            return ApiReturn::success($list);
        }catch (\Exception $e) {
ErrorLog::dispatch(Route::current()->getActionName(),CommonFnc::getRealIP(),json_encode($e),['file'=>$e->getFile(),'line'=>$e->getLine(),'code'=>$e->getCode(),'message'=>$e->getMessage()],time())->onQueue('error_log');
            return ApiReturn::error($e->getMessage());
        }
    }
    /**
     * 文章详情
     * @param Request $request
     * @return Object
     */
    public function getArticleInfo(Request $request): object
    {
        try {
            $uid = $request->get('user_id');
            $deviceNum = $request->get('device_num');
            $id = $request->get('article_id',0);
            $type = $request->get('type',1);
            if(empty($id)){
                return ApiReturn::rdata(ApiCode::PARAM_ERROR);
            }
            if(empty($uid)) $uid = 0;
            switch ($type) {
                case Code::ARTICLE_TYPE_MAP['video']:
                    $info = V1Article::getVideoInfo($id,$uid);
                    break;
                case Code::ARTICLE_TYPE_MAP['dynamic']:
```

```
                $info = V1Article::getDynamicInfo($id,$uid);
                break;
            case Code::ARTICLE_TYPE_MAP['long_text']:
                $info = V1Article::getLongTextInfo($id,$uid);
                break;
            case Code::ARTICLE_TYPE_MAP['evaluate']:
                $info = V1Article::getEvaluateInfo($id,$uid);
                break;
            case Code::ARTICLE_TYPE_MAP['evaluating']:
                $info = V1Article::getEvaluatingInfo($id,$uid);
                break;
            default:
                return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        if(!empty($info)){
            // 获奖判断
            $info['is_awarded'] = (bool)$info['user_award'];
            unset($info['user_award']);
            $info['is_attention'] = User::isAttention($uid,$info['user_id']);
            dispatch(new ReadArticle($uid,$info,$deviceNum))->onQueue('read_article');
            $info['create_time'] = Common::disposeDateTime(strtotime($info['create_time']));
            $info['like_num'] = Common::disposeAmount($info['like_num']);
            return ApiReturn::success($info);
        }else return ApiReturn::error('该内容已被删除');
    }catch (\Exception $e) {

ErrorLog::dispatch(Route::current()->getActionName(),CommonFnc::getRealIP(),json_encode($e),['file'=>$e->getFile(),'line'=>$e->getLine(),'code'=>$e->getCode(),'message'=>$e->getMessage()],time())->onQueue('error_log');
        return ApiReturn::error($e->getMessage());
    }
}
/**
 * showdoc
 * @catalog 社区关注信息流
 * @title 社区关注信息流
 * @description 社区关注信息流的接口
 * @method POST
 * @url /api/article/publishGameEvaluate
 * @param ArticleRequest $request
 * @return Object
 */
public function getAttentionArticleList(ArticleRequest $request): object
{
    try {
        $userId = $request->get('user_id');
        $platform = (int)$request->get('platform');
        $page = (int)$request->get('page',1);
        $limit = (int)$request->get('limit',10);
        $device_id = $request->get('device_num');
        $response = V1Article::getAttentionArticleList($userId,$page,$limit,$platform,$device_id);
        return response()->json([
                'code' => 200,
```

```
                'msg' => "OK",
                'data' => $response['articles'],
                'total_count' =>$response['total_count']
            ]);
        }catch (\Throwable $throwable) {
            Common::log('社区关注信息流',$throwable);
            return ApiReturn::error();
        }
    }
    //模版
    public function templates()
    {

        return V1Article::templates();
    }
    /**
     * showdoc
     * @catalog 社区关注信息流
     * @title 社区关注信息流
     * @description 社区关注信息流的接口
     * @method POST
     * @url /api/article/publishGameEvaluate
     * @param ArticleRequest $request
     * @return Object
     */
    public function getAttentionArticleListNew(ArticleRequest $request): object
    {
        try {
            $userId = $request->get('user_id');
            $platform = (int)$request->get('platform');
            $page = (int)$request->get('page',1);
            $limit = (int)$request->get('limit',10);
            $device_id = $request->get('device_num');
            return V1Article::getAttentionArticleListNew($userId,$page,$limit,$platform,$device_id);
        }catch (\Throwable $throwable) {
            Common::log('社区关注信息流',$throwable);
            return ApiReturn::error($throwable->getMessage());
        }
    }
    //推荐文章
    public function getRecommend(ArticleRequest $request)
    {
        $limit = $request->get('limit', 10);
        $user_id = $request->get('user_id');
        $game_id = $request->get('game_id');
        $elite = $request->get('elite');
        $page = $request->get('page',1);
        $device_id = $request->get('device_num');
        return V1Article::getRecommendEs($user_id, $page ,$limit, $game_id, $elite,0,$device_id);
    }
```

```
    //文章详情
    public function detail(ArticleRequest $request)
```

```
{
    $article_id = $request->get('article_id');
    if (!$article_id) {
        return ApiReturn::error('请选择要查看的文章');
    }
    $user_id = $request->get('user_id');
    return V1Article::detail($article_id,$user_id);
}
//评论列表
public function commentList(ArticleRequest $request)
{
    $article_id = $request->get('article_id');
    if(empty($article_id)){
        return ApiReturn::error('参数错误');
    }
    $limit = $request->get('limit', 10);
    $user_id = $request->get('user_id');
    return V1Article::commentList($article_id, $limit, $user_id);
}
//增加浏览记录
public function addReadRecord(ArticleRequest $request)
{
    $article_id = $request->get('article_id');
    $user_id = $request->get('user_id');
    $deviceNum = $request->post('device_num','');
    if(empty($article_id)){
        return ApiReturn::error('参数错误');
    }
    return V1Article::addReadRecord($article_id,$user_id,$deviceNum);
}
/**
 * 文章评论回复列表
 * @param Request $request
 * @return Object
 */
public function getCommentReplyList(Request $request): object
{
    try {
        $commentId = intval($request->get('comment_id',0));
        $limit = intval($request->get('limit',10));
        $userId = intval($request->get('user_id'));
        if(empty($commentId)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $exclude_ids = $request->get('exclude_ids','');
        return V1Article::getCommentReplyList($commentId,$userId,$limit,$exclude_ids);
    }catch (\Exception $e) {
        return ApiReturn::error();
```

```
    }
}
//游戏文章列表
public function gameGameArticleList(ArticleRequest $request)
{
    $game_id = $request->get('game_id');
    $circle_id = $request->get('circle_id');
    if(empty($game_id) && empty($circle_id)){
        return ApiReturn::error('参数错误');
    }
    $limit = $request->get('limit', 10);
    $user_id = $request->get('user_id');
    $elite = $request->get('elite');
    $is_review = $request->get('is_review');
    Log::info('获取游戏文章列表入参', $request->all());
    $device_id = $request->get('device_num');
    return V1Article::getRecommend($user_id, $limit, $game_id, $elite,$is_review,$circle_id,$device_id,true);
}
//同步来源于SoloGame的文章
public function syncArticleBySoloGame(Request $request)
{
    $param =  $request->all();
    if(empty($param['title'])){
        return ApiReturn::error('请输入文章标题');
    }
    if(empty($param['content'])){
        return ApiReturn::error('请输入文章内容');
    }
    if(empty($param['rss_guid'])){
        return ApiReturn::error('请输入rss_guid');
    }
    if(empty($param['link_url'])){
        return ApiReturn::error('请输入文章url');
    }
    if(empty($param['lang'])){
        return ApiReturn::error('请输入文章语言');
    }
    return V1Article::syncArticleBySoloGame($param);
}
public function syncArticleBySoloPublisher(Request $request)
{
    $params = $request->all();
    return V1Article::syncArticleBySoloPublisher($params);
}
//文章相关推荐
public function getRelatedArticle(ArticleRequest $request)
{
    $user_id = $request->get('user_id');
    $article_id = $request->get('article_id');
    if(empty($article_id)){
```

```
            return ApiReturn::success();
        }
        //社区ID
        $circle_id = $request->get('circle_id');
        //游戏ID
        $game_id = $request->get('game_id');
        $page = $request->get('page',1);
        $limit = $request->get('limit', 10);
        //是否测评
        $type = $request->get('type',0);
        if($type == '(null)'){
            $type = 0;
        }
        if($circle_id == '(null)'){
            $circle_id = 0;
        }
        $device_id = $request->get('device_num');
        return V1Article::getRelatedArticle($user_id,$article_id, $circle_id, $game_id, $page, $limit,$type,$device_id);
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1\BindingGame;

use App\Http\Controllers\Controller;
use App\Models\v1\BindingGame\GameGameLabel;
use App\Models\v1\BindingGame\GameImage;
use App\Models\v1\BindingGame\GameInfo;
use App\Models\v1\BindingGame\GameLabel;
use App\Models\v1\BindingGame\GameManufacturer;
use App\Models\v1\Circle;
use App\Models\v1\Game;
use App\Models\v1\GameDefaultPlatformGame;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\check_url\Tap;
use App\Service\v1\Common;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use function Stringy\create;

class BindingGameController extends Controller
{
    /**
     * 三方平台游戏绑定solo平台游戏
     * @param Request $request
     * @return Object
     */
    public function editBindingGame(Request $request)
    {
        $platformId = (int)$request->post('platform_id',0);
        $platformGameId = (int)$request->post('platform_game_id',0);

        if($platformId <= 0 || $platformGameId <= 0) return 
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $params = $request->post('game_info',[]);
        if(!isset($params['game_name']) || empty($params['game_name'])) return 
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        try {
            DB::beginTransaction();
            $gameInfo = Game::query()->withoutGlobalScope('status')->where(['game_name' =>
(string)$params['game_name']])->first();
            $labelIds = [];
            if(empty($params['game_label'])) {
                $params['game_label'] = '';
            }else{
                foreach ($params['game_label'] as $label) {
                    $labelInfo = GameLabel::query()->where(['label_name' => $label])->first();
                    if(empty($labelInfo)) {
                        $labelInfo = GameLabel::query()->create(['label_name' => $label]);
                    }
                    $labelIds[] = $labelInfo->id;
                }
```

```
            $params['game_label'] = GameLabel::query()->select('id as
label_id','label_name')->whereIn('id',$labelIds)->get()->toJson();
        }
        if(empty($gameInfo)) {
            $manufacturerName = (string)$params['manufacturer_name'];
            if(!empty($manufacturerName)) {
                $manufacturerInfo = GameManufacturer::query()->where(['name' => $manufacturerName])->first();
                if(empty($manufacturerInfo)) {
                    $manufacturerInfo = GameManufacturer::query()->create(['name' => $manufacturerName]);
                }
                $params['manufacturer_id'] = $manufacturerInfo->id;
            }
            $params['status'] = 0;
            $gameInfo = Game::query()->create($params);
            Circle::create([
                'game_id' => $gameInfo->id,
                'circle_name' => $gameInfo->game_name,
                'avatar' => $gameInfo->game_icon,
                'cover' => $gameInfo->game_cover,
            ]);
        }else{
            $gameInfo->game_cover = $params['game_cover'];
            $gameInfo->score = $params['score'];
            $gameInfo->game_icon = $params['game_icon'];
            $gameInfo->game_label = $params['game_label'];
            $gameInfo->save();
        }
        if (count($labelIds)) $gameInfo->game_labels()->attach($labelIds);
        GameInfo::query()->updateOrCreate([
            'game_id' => $gameInfo->id
        ],[
            'content' => $this->deleteBr($params['content']),
            'version' => $params['version'] ?: '',
            'size' => $params['size'] ?: 0,
            'service' => $params['service'] ?: '',
            'game_update_time' => $params['game_update_time'],
            'update_log' => $params['update_log'],
        ]);
        if(!empty($params['images'])) {
            GameImage::query()->where(['game_id' => $gameInfo->id,'type' => 1])->delete();
            $images = [];
            foreach ($params['images'] as $key => $image) {
                $images[] = [
                    'game_id' => $gameInfo->id,
                    'url' => $image,
                    'type' => 1,
                    'sort' => $key+1,
                    'status' => 2
                ];
                GameImage::query()->insert($images);
            }
```

```
        }
        if(!empty($params['videos'])) {
            GameImage::query()->where(['game_id' => $gameInfo->id,'type' => 2])->delete();
            $videos = [];
            foreach ($params['videos'] as $key => $video) {
                $videos[] = [
                    'game_id' => $gameInfo->id,
                    'url' => $video,
                    'type' => 2,
                    'sort' => $key+1,
                    'status' => 2
                ];
                GameImage::query()->insert($videos);
            }
        }
        GameDefaultPlatformGame::query()->updateOrCreate([
            'game_id' => $gameInfo->id,
            'platform_type' => $platformId
        ],['platform_game_id' => $platformGameId]);
        DB::commit();
        return ApiReturn::success();
    }catch (\Throwable $throwable) {
        DB::rollBack();
        Common::log('绑定游戏错误',$throwable);
        return ApiReturn::error();
    }
}

/**
 * 根据第三方平台id和游戏名称搜索游戏
 * @param Request $request
 * @return Object
 */
public function games(Request $request)
{
    $platformId = (int)$request->get('platform_id',0);
    $gameName = (string)$request->get('game_name','');
    if($platformId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
    $page = (int)$request->get('page',1);
    $limit = (int)$request->get('limit',20);
    $page = ($page - 1) * $limit;
    try {
            $games = Game::query()
                ->join('game_default_platform_games as pg','games.id','=','pg.game_id')
                ->where('games.game_name','like',"%{$gameName}%")
                ->where(['pg.platform_type' => $platformId])
                ->withoutGlobalScope('status')
                ->offset($page)->limit($limit)->get(['games.id','games.game_name','pg.platform_game_id']);
//            $games = GameDefaultPlatformGame::query()->with(['game' => function($query) use($gameName) {
//                $query->withoutGlobalScope('status')->where('game_name','like',"%{$gameName}%")->select('id','game_name');
```

```
//            }])->where(['platform_type' => $platformId])->offset($page)->limit($limit)->get();
//            $list = [];
//            foreach ($games as $game) {
//                if(!empty($game->game)) $list[] = [
//                    'id' => $game->game->id,
//                    'game_name' => $game->game->game_name,
//                    'platform_game_id' => $game->platform_game_id
//                ];
//            }
            return ApiReturn::success(['games' => $games]);
        }catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('三方平台游戏列表错误',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 内容过滤br标签
     * @param string $content
     * @return string
     */
    public function deleteBr(string $content) :string
    {
        if(empty($content)) return '';
        $preg = "/\<br.*?\/\>/";
        $content = preg_replace($preg,"\r\n",$content);
        return $content;
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Service\v1\ApiReturn;

class CaptchaController extends Controller
{
    public function getCaptcha()
    {
        return ApiReturn::success(app('captcha')->create('default', true));
    }
}
```

```
```

```php
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Service\CodeMessage;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Checkin;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Validator;

class CheckinController extends Controller
{
    //创建打卡
    public function add(Request $request)
    {
        $param = $request->all();
        if (empty($param['user_id'])) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        // 定义校验规则
        $rules = [
            'content' => 'required',
            'type' => 'required|numeric|min:1|max:3',
            'start_time' => ['required', 'date', 'after_or_equal:'.date('Y-m-d H:i'), 'before_or_equal:end_time'],
            'end_time' => ['required', 'date', 'after:start_time'],
            'community_id' => 'required_if:type,3',
            // 添加其他字段的校验规则
        ];
        $messages = [
            'content.required' => '请输入内容',
            'type.required' => '缺少参数',
            'type.numeric' => '参数格式不正确',
            'type.min' => '参数格式不正确',
            'type.max' => '参数格式不正确',
            'start_time.required' => '请输入开始时间',
            'start_time.date' => '开始时间格式不正确',
            'start_time.after_or_equal' => '开始时间不能早于当前时间',
            'start_time.before_or_equal' => '开始时间不能晚于结束时间',
            'end_time.required' => '请输入结束时间',
            'end_time.date' => '结束时间格式不正确',
            'end_time.after' => '结束时间必须晚于开始时间',
            'community_id.required_if' => '请选择社群'
        ];
        // 执行校验并获取校验后的数据
        $validator = Validator::make($param, $rules, $messages);
        if ($validator->fails()) {
            return ApiReturn::error($validator->errors()->first());
```

```php
        }
        return Checkin::add($param);
```

```
    }


    //打卡创建详情
    public function info(Request $request)
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        //打卡创建ID
        $checkin_creation_id = $request->get('checkin_creation_id');
        if (empty($checkin_creation_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        //第三方群类型
        $external_group_type = $request->get('external_group_type','wechat');
        //第三方群ID
        $external_group_id = $request->get('external_group_id');
        //加权因子
        $iv = $request->get('iv');
        //token
        $token = $request->bearerToken();
        //社群邀请用户ID
        $community_inviter_id = $request->get('community_inviter_id',0);
        return Checkin::info($user_id,
$checkin_creation_id,$external_group_type,$external_group_id,$iv,$token,$community_inviter_id);
    }

    //点击打卡
    public function click(Request $request)
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        //打卡创建ID
        $checkin_creation_id = $request->post('checkin_creation_id');
        if (empty($checkin_creation_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        //第三方群类型
        $external_group_type = $request->post('external_group_type','wechat');
        //第三方群ID
        $external_group_id = $request->post('external_group_id');
        //加密参数
        $iv = $request->post('iv');
        //token
        $token = $request->bearerToken();
```

```
        return Checkin::click($user_id, $checkin_creation_id, $external_group_type, $external_group_id,$iv,$token);
    }
    //已参与人员列表
    public function participants(Request $request){
        //打卡创建ID
        $checkin_creation_id = $request->get('checkin_creation_id');
        if (empty($checkin_creation_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        $limit = $request->get('limit',10);
        return Checkin::participants($checkin_creation_id,$limit);
    }
    //我参与的打卡
    public function myCheckinsParticipated(Request $request){
        $limit = $request->get('limit',10);
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return Checkin::myCheckinsParticipated($user_id,$limit);
    }
    //结束打卡
    public function stop(Request $request) {
        $checkin_creation_id = $request->get('checkin_creation_id');
        if (empty($checkin_creation_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return Checkin::stop($user_id,$checkin_creation_id);
    }
    //获取不受限制小程序分享海报
    public function getWechatMiniProgramSharePoster(Request $request)
    {
        $param = $request->all();
        if (empty($param['user_id'])) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        if (empty($param['checkin_creation_id'])) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        if (empty($param['page'])) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Checkin::getWechatMiniProgramSharePoster($param);
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Events\SearchEvent;
use App\Http\Controllers\Controller;
use App\Models\UserBrowseLog;
use App\Models\v1\BindingGame\GameGameLabel;
use App\Models\v1\Circle;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Article as V1Article;
use App\Service\v1\CircleService;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\Search;
use App\Service\v1\UserNotifications;
use Illuminate\Http\Request;

class CircleController extends Controller
{
    /**
     * 搜索圈子
     * @param Request $request
     * @return Object
     */
    public function circleSearch(Request $request)
    {
        try {
            $name = strval($request->get('circle_name',''));
            $userId = $request->get('user_id',1);
            $deviceNum = $request->get('device_num','');
            $page = $request->get('page',1);
            $limit = $request->get('limit',10);
            $field  = ["id","circle_name","avatar","game_id","articles_num","attention_num"];
            $circles = Search::circles($name,$page,$limit,$field);
            if(!empty($name)) SearchEvent::dispatch($name,(int)$userId,(string)$deviceNum);
            return ApiReturn::success($circles);
        }catch (\Throwable $throwable) {
            Common::log('搜索圈子',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 热门圈子
     * @param Request $request
     * @return Object
     */
```

```
    public function getHotCircleList(Request $request)
    {
        try {
            $userId = $request->get('user_id');
            $page = (int)$request->get('page',1);
            $limit = (int)$request->get('limit',15);
            $type = (int)$request->get('type',1);
            $list = CircleService::getHotCircleList($userId,$page,$limit,$type);
            $rdata = [
                'code' => 200,
                'msg'  => 'OK',
                'data' => $list['circles'],
                'total_count'=>$list['total_count']
            ];
            return response()->json($rdata);
        } catch (\Throwable $throwable) {
            Common::log('热门圈子',$throwable);
            return ApiReturn::error();
        }
    }
    //圈子列表
    public function getCircleList(Request $request)
    {
        $userId = $request->get('user_id');
        $page = (int)$request->get('page',1);
        $limit = (int)$request->get('limit',15);
        $type = (int)$request->get('type',1);
        return CircleService::getCircleList($userId,$page,$limit,$type);
    }
    /**
     * 最近预览的圈子
     * @param Request $request
     * @return Object
     */
    public function getUserLastBrowseList(Request $request): object
    {
        try {
            $uid = $request->get('user_id');
            $device_num = $request->get('device_num');
            $limit = $request->get('limit', 20);
            $list = (new UserBrowseLog())->getCircleBrowseList($uid,$device_num ? :'',$limit);
            foreach($list as &$v){
                $label = GameGameLabel::query()
                    ->where('game_id',$v['game_id'])
                    ->whereHas('game_labels')
                    ->with('game_labels' ,function ($query){
                        $query->select('id','label_name');
                    })
                    ->limit(3)
                    ->get()
```

```
                    ->toArray();
                $label = array_column($label,'game_labels');
                $label = array_column($label,'label_name');
                $v['game_label_txt'] = '';
                if(!empty($label)){
                    $v['game_label_txt'] = implode('·',$label);
                }
                $v['ut'] = date('Y.m.d',$v['ut']);
                $v['avatar'] = Common::concatUrlStr($v['avatar']);
                $v['cover'] = Common::concatUrlStr($v['cover']);
                $v['is_attention'] = CircleService::isAttention($uid,$v['id']);
                $v['articles_num'] = Common::disposeAmount($v['articles_num']);
                $v['attention_num'] = Common::disposeAmount($v['attention_num']);
                $v['circle_id'] = $v['id'];
            }
            $res = [
                'total_count' => (int)$limit,
                'circles' => $list
            ];
            return ApiReturn::success($res);
        } catch (\Exception $e) {
            return ApiReturn::error($e->getMessage());
        }
    }
    public function getRecommendCircleList(Request $request)
    {
        $game_label_ids = $request->get('game_label_ids','');
        $label_ids = [];
        if(!empty($game_label_ids)){
            $label_ids = explode(',',$game_label_ids);
            //去除空
            $label_ids = array_filter($label_ids);
            //去重
            $label_ids = array_unique($label_ids);
        }
        return  CircleService::getRecommendCircleList($label_ids);
    }
    /**
     * 圈子详情
     * @param Request $request
     * @return Object
     */
    public function getCircleInfo(Request $request): object
    {
        try {
            $userId = $request->get('user_id');
            $circleId = intval($request->get('circle_id',0));
            if(empty($circleId)){
                return ApiReturn::rdata(ApiCode::PARAM_ERROR);
            }
```

```
            $circleInfo = \App\Service\v1\Circle::getCircleInfo($userId,$circleId);
            return ApiReturn::success($circleInfo);
        }catch (\Exception $e) {
            return ApiReturn::error($e->getMessage());
        }
    }

    //圈子文章列表
    public function articleList(Request $request)
    {

        $limit = $request->get('limit', 10);
        $type = $request->get('type', 'all');
        $orderBy = $request->get('order_by', 'publish_time');
        $circleId = $request->get('circle_id');
        if (empty($circleId)) {
            $list = [];
        }else{
            $lang = CommonFnc::lang();
            $device_id = $request->get('device_num');
            $userId = $request->get('user_id');
            $list = CircleService::articleList($limit,$type,$orderBy,$circleId,$userId,$device_id,$lang);
        }
        return ApiReturn::success($list);
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;


use App\Http\Controllers\Controller;
use App\Service\CodeMessage;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Community;
use App\Service\v1\WechatMiniProgram;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;

class CommunityController extends Controller
{
    /**
     * 创建社群
     * @param Request $request
     * @return JsonResponse|Object
     */
    public function add(Request $request)
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $name = $request->post('name');
        if (empty($name)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Community::add($user_id, $name);
    }

    /**
     * 修改社群
     * @param Request $request
     * @return array|JsonResponse|Object
     */
    public function edit(Request $request)
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $community_id = $request->post('community_id');
        if (empty($community_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        $type = $request->post('type');
```

```
        if (empty($type)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        $data = $request->post('data');
        if (!isset($data) || $data === '') {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Community::edit($user_id, $community_id, $type, $data);
    }

    /**
     * 加入社群
     * @param Request $request
     * @return JsonResponse|Object
     */
    public function join(Request $request)
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $community_id = $request->post('community_id');
        if (empty($community_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        $nick_name = $request->post('nick_name');
        if (empty($nick_name)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Community::join($user_id, $community_id, $nick_name);
    }

    //社群详情
    public function info(Request $request)
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $community_id = $request->get('community_id');
        if (empty($community_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Community::info($user_id, $community_id);
    }

    //社群成员
    public function memberships(Request $request)
    {
        $user_id = $request->get('user_id');
```

```
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $community_id = $request->get('community_id');
        if (empty($community_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        $limit = $request->get('limit', 10);
        return Community::memberships($user_id, $community_id, $limit);
    }

    //我参与的社群
    public function myCommunities(Request $request)
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $limit = $request->get('limit', 10);
        return Community::myCommunities($user_id, $limit);
    }

    //我参与的全部社群
    public function myCommunitiesAll(Request $request)
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return Community::myCommunitiesAll($user_id);
    }

    //退出社群
    public function quit(Request $request)
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $community_id = $request->post('community_id');
        if (empty($community_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Community::quit($user_id, $community_id);
    }

    //解散群聊
    public function dissolve(Request $request)
    {
        $user_id = $request->post('user_id');
```

```
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $community_id = $request->post('community_id');
        if (empty($community_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Community::dissolve($user_id, $community_id);
    }

    //获取不受限制小程序分享海报
    public function getWechatMiniProgramSharePoster(Request $request)
    {
        $param = $request->all();
        if (empty($param['user_id'])) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        if (empty($param['community_id'])) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        if (empty($param['page'])) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        return Community::getWechatMiniProgramSharePoster($param);
    }
    //踢出群员
    public function kickMember(Request $request) {
        //群主Id
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        //社群ID
        $community_id = $request->post('community_id');
        if (empty($community_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        //成员ID
        $member_id = $request->post('member_id');
        if (empty($member_id)) {
            return ApiReturn::rdata(CodeMessage::common(30001));
        }
        //检测要被踢出成员是否为当前账号
        if($user_id == $member_id){
            return ApiReturn::error('您不能将自己退出群聊');
        }
        return Community::kickMember($user_id,$community_id,$member_id);
    }
}

```

```
<?php
```

```

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\RedisKey;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Redis;
use Predis\Client;

class ConversionExplainController extends Controller
{
    /**
     * 积分商城兑换说明
     * @return Object
     */
    public function conversionExplain()
    {
        try {
            $redisConf = config('database.redis.default');
            $client = new Client($redisConf);
            $info = $client->get(env('SOLO_ADMIN_REDIS_PREFIX').RedisKey::CONVERSION_EXPLAIN);
            return ApiReturn::success(['explain' => $info]);
        } catch (\Throwable $throwable) {
            Common::log('积分商城兑换说明',$throwable);
            return ApiReturn::error();
        }
    }
}

```

```
<?php


namespace App\Http\Controllers\api\v1;

use App\Events\ShareArticleTaskEvent;
use App\Http\Controllers\Controller;
use App\Models\v1\ChatLog;
use App\Models\v1\User;
use App\Models\v1\UserFanBase;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\FanBase;
use App\Service\v1\RedisKey;
use App\Service\v1\task\JoinFanBaseTask;
use App\Service\v1\task\LogoutFanBaseTask;
use App\Service\v1\task\ShareArticleTask;
use App\Service\v1\task\ShareImageTask;
use Hhxsv5\LaravelS\Swoole\Task\Task;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Redis;
```

```

class FanBaseController extends Controller
{
    /**
     * 聊天记录
     * @param Request $request
     * @return Object
     */
    public function getChatLogs(Request $request)
    {
        $fanBaseId = intval($request->get('fan_base_id',0));
        $chatLogId = intval($request->get('chat_log_id',0));
        if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->get('user_id');
        try {
            $startTime = UserFanBase::where(['user_id' => $userId,'fan_base_id' => $fanBaseId,'status' =>
1])->value('updated_at');
            if(empty($startTime)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
            $page = $request->get('page',1);
            $limit = $request->get('limit',20);
            $page = ($page - 1) * $limit;
            $where = [
                ['fan_base_id','=',$fanBaseId],
                ['created_at','>=',$startTime]
            ];
            if($chatLogId > 0) $where[] = ['id','<',$chatLogId];
            $chatLogs = (new ChatLog())->getChatLogs($where,$page,$limit);
            return ApiReturn::success($chatLogs);
        }catch (\Throwable $throwable) {
```

```
            Common::log('获取聊天记录',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 群成员
     * @param Request $request
     * @return Object
     */
    public function getFanBaseMembers(Request $request)
    {
        $fanBaseId = intval($request->get('fan_base_id',0));
        if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $page = intval($request->get('page',1));
        $limit = intval($request->get('limit',20));
        $page = ($page - 1) * $limit;
        try {
            $users = (new UserFanBase())->getFanBaseUsers($fanBaseId,$page,$limit);
            return ApiReturn::success($users);
        }catch (\Throwable $throwable) {
            Common::log('获取聊天记录',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 退群
     * @param Request $request
     * @return Object
     */
    public function logoutFanBase(Request $request)
    {
        $fanBaseId = intval($request->get('fan_base_id',0));
        if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->get('user_id');
        try {
//            $result = (new UserFanBase())->logoutFanBase($fanBaseId,$userId);
//            if($result == 0) return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
            $task = new LogoutFanBaseTask($userId,$fanBaseId);
            $result = Task::deliver($task);
            if($result) return ApiReturn::success();
            return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
        }catch (\Throwable $throwable) {
            Common::log('退群',$throwable);
            return ApiReturn::error();
        }
    }

```

```
    /**
     * 获取我的群列表
     * @param Request $request
     * @return Object
     */
    public function fanBases(Request $request)
    {
        try {
            $userId = $request->get('user_id');
            $page = $request->get('page',1);
            $limit = $request->get('limit',10);
            $page = ($page - 1) * $limit;
            $fanBases = FanBase::fanBases($userId,$page,$limit);
            return ApiReturn::success($fanBases);
        }catch (\Throwable $throwable) {
            Common::log('获取我的群列表',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 分享到群到群列表
     * @param Request $request
     * @return Object
     */
    public function shareFanBases(Request $request)
    {
        try {
            $userId = $request->get('user_id');
            $page = $request->get('page',1);
            $limit = $request->get('limit',20);
            $page = ($page - 1) * $limit;
            $fanBases = FanBase::shareFanBases($userId,$page,$limit);
            return ApiReturn::success($fanBases);
        }catch (\Throwable $throwable) {
            Common::log('分享到群到群列表',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 加入群聊
     * @param Request $request
     * @return Object
     */
    public function joinFanBase(Request $request)
    {
        try {
            $fanBaseId = intval($request->get('fan_base_id',0));
            if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
```

```
            $userId = $request->get('user_id');
            $redisNx = boolval(Redis::set(RedisKey::USER_JOIN_FAN_BASE_STATUS.$userId,1,'EX',2,'NX'));
            if($redisNx) {
                $status = FanBase::isFans($fanBaseId,$userId);
                if(empty($status)) {
                    $task = new JoinFanBaseTask($userId,$fanBaseId);
                    $result = Task::deliver($task);
                    if($result) return ApiReturn::success();
                }else{
                    return ApiReturn::success();
                }
            }
            return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
        }catch (\Throwable $throwable) {
            Common::log('加入群聊',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 分享群页面
     * @param Request $request
     * @return Object
     */
    public function sharePage(Request $request)
    {
        try {
            $fanBaseId = intval($request->get('fan_base_id',0));
            if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = $request->get('user_id');
            $re = FanBase::sharePage($userId,$fanBaseId);
            return ApiReturn::success($re);
        }catch (\Throwable $throwable) {
            Common::log('分享群页面',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * pc分享群页面
     * @param Request $request
     * @return Object
     */
    public function pcSharePage(Request $request)
    {
        try {
            $fanBaseId = intval($request->get('fan_base_id',0));
            $userId = intval($request->get('share_user_id',0));
            if($fanBaseId <= 0 || $userId <= 0) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $re = FanBase::sharePage($userId,$fanBaseId);
```

```
            return ApiReturn::success($re);
        }catch (\Throwable $throwable) {
            Common::log('pc分享群页面',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 分享文章到群
     * @param Request $request
     * @return Object
     */
    public function shareArticleToFanBase(Request $request)
    {
        try {
            $fanBaseId = (int)$request->post('fan_base_id',0);
            $articleId = (int)$request->post('article_id',0);
            if($fanBaseId <= 0 || $articleId <= 0) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = $request->post('user_id');
            $content = (string)$request->post('content','');
            $task = new ShareArticleTask($userId,$fanBaseId,$articleId,$content);
            $result = Task::deliver($task);
            if($result) {
                ShareArticleTaskEvent::dispatch($userId,$articleId);
                return ApiReturn::success();
            }
            return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
        }catch (\Throwable $throwable) {
            Common::log('分享文章到群错误',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 分享图片到群聊
     * @param Request $request
     * @return Object
     */
    public function shareImageToFanBase(Request $request)
    {
        try {
            $fanBaseId = (int)$request->post('fan_base_id',0);
            $image = (string)$request->post('image','');
            if($fanBaseId <= 0 || empty($image)) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = $request->post('user_id');
            $task = new ShareImageTask($userId,$fanBaseId,$image);
            $result = Task::deliver($task);
            if($result) {
                return ApiReturn::success();
            }
```

```
            return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
        }catch (\Throwable $throwable) {
            Common::log('分享文章到群错误',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 设置下线时间
     * @param Request $request
     * @return Object
     */
    public function setOffLineTime(Request $request)
    {
        $userId = $request->get('user_id');
        try {
            User::where(['id' => $userId])->update(['off_line_time' => $request->server('REQUEST_TIME')]);
            return ApiReturn::success();
        }catch (\Throwable $throwable) {
            Common::log('设置下线时间错误',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 任务中心分享群聊
     * @param Request $request
     * @return Object
     */
    public function shareFanBase(Request $request)
    {
        try {
            $fanBaseId = intval($request->post('fan_base_id',0));
            if($fanBaseId >= 0) {
                $userId = $request->post('user_id');
                $count = \App\Models\v1\FanBase::where(['id' => $fanBaseId])->count();
                if($count > 0) {
                    Redis::eval(<<<'LUA'
                        redis.call("INCR",KEYS[1])
                        if redis.call("TTL",KEYS[1]) < 0 then
                            redis.call("EXPIREAT",KEYS[1],ARGV[1])
                        end
                    LUA, 1, RedisKey::SYSTEM_TASK_EVERYDAY_SHARE_GROUP.$userId,strtotime('tomorrow'));
                }
            }
            return ApiReturn::success();
        }catch (\Throwable $throwable) {
            Common::log('任务中心分享群聊',$throwable);
```

```
            return ApiReturn::error();
        }
    }


    /**
     * 群主创建的群列表
     * @param Request $request
     * @return Object
     */
    public function groupOwnerFanBases(Request $request)
    {
        $userId = $request->get('user_id');
        $groupOwnerId = (int)$request->get('group_owner_id',0);
        if($groupOwnerId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        try {
            $fanBases = FanBase::groupOwnerFanBases($userId,$groupOwnerId);
            return ApiReturn::success($fanBases);
        }catch (\Throwable $throwable) {
            Common::log('群主创建的群列表',$throwable);
            return ApiReturn::error();
        }
    }

}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Events\GameSearchEvent;
use App\Http\Controllers\Controller;
use App\Models\Leaderboard;
use App\Models\v1\ServiceConfig;
use App\Models\v1\UserGame;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\Game;
use App\Service\v1\ProductService;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class GameController extends Controller
{

    /**
     * 我的游戏页/ios发布动态页--搜索游戏
     * @param Request $request
     * @return Object
     */
    public function searchGames(Request $request)
    {
        try {
            $gameName = $request->get('game_name');
            if (empty($gameName)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            $userId = $request->get('user_id');
            $page = $request->get('page', 1);
            $limit = $request->get('limit', 20);
            $games = Game::searchGames($userId, $gameName, $page, $limit);
            GameSearchEvent::dispatch($gameName, $userId, $request->get('device_num', '') ?? '');
            return ApiReturn::success($games);
        } catch (\Throwable $throwable) {
            Common::log('搜索游戏错误', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 我的游戏
     * @param Request $request
     * @return Object
     */
    public function myGames(Request $request)
    {
```

```
        try {
            $userId = $request->get('user_id');
            $page = $request->get('page', 1);
            $limit = $request->get('limit', 10);
            $page = ($page - 1) * $limit;
            $games = Game::myGames($userId, $page, $limit);
            return ApiReturn::success($games);
        } catch (\Throwable $throwable) {
            Common::log('我的游戏', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 加入我的游戏
     * @param Request $request
     * @return Object
     */
    public function addMyGame(Request $request)
    {
        try {
            $userId = $request->post('user_id');
            $gameIds = $request->post('game_ids', []);
            if (empty($gameIds) || !is_array($gameIds)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
            foreach ($gameIds as $gameId) {
                UserGame::updateOrCreate([
                    'user_id' => $userId,
                    'game_id' => $gameId
                ], ['status' => 1]);
            }
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('加入我的游戏', $throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 删除我的游戏
     * @param Request $request
     * @return Object
     */
    public function delMyGame(Request $request)
    {
        try {
            $userId = $request->post('user_id');
            $gameIds = $request->post('game_ids', []);
            if (empty($gameIds) || !is_array($gameIds)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
            UserGame::where(['user_id' => $userId, 'status' => 1])->whereIn('game_id', $gameIds)->update(['status' => 0]);
```

```
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('删除我的游戏', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 玩过的游戏（临时解决方案）
     * @param Request $request
     * @return Object
     */
    public function playedGames(Request $request)
    {
        try {
            $userId = $request->post('user_id');
            $games = Game::playedGames($userId);
            return ApiReturn::success($games);
        } catch (\Throwable $throwable) {
            Common::log('玩过的游戏', $throwable);
            return ApiReturn::error();
        }
    }

    //pc首页排行榜
    public function pcHomeRanking(): object
    {
        return Game::pcHomeRanking();
    }

    //相关游戏
    public function related(Request $request)
    {
        $game_id = $request->get('game_id', 0);
        return Game::related($game_id);
    }

    /**
     * 获取下载链接并新增下载记录
     * @param Request $request
     * @return object
     */
    public function getDownLoadUrl(Request $request): object
    {
        $game_id = $request->get('game_id');
        if (empty($game_id)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
```

```
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        //设备信息
        $platform = $request->header('Platform', 1);
        return Game::getDownloadUrl($user_id, $game_id, $platform);
    }

    //游戏推荐列表
    public function recommend(Request $request): object
    {
        $user_id = $request->get('user_id');
        $page = $request->get('page', 1);
        $limit = $request->get('limit', 10);
        return Game::recommendList($user_id, $page, $limit);
    }

    //游戏详情
    public function detail(Request $request): object
    {
        $lang = CommonFnc::lang();
        $game_id = $request->get('game_id');
        if (empty($game_id)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $user_id = $request->get('user_id');
        return Game::detail($game_id, $user_id,$lang);
    }

    //排行榜
    public function ranking(Request $request)
    {
        Log::error('排行榜', $request->all());
        $id = $request->get('id', 1);
        $page = $request->get('page', 1);
        $limit = $request->get('limit', 10);
        if ($page * $limit > 150) {
            return ApiReturn::success();
        }
        $leardboard = Leaderboard::query()->select(['id', 'name', 'cover', 'lang'])->find($id);
        if (!$leardboard) {
            return ApiReturn::success();
        }
        $leardboard->cover = CommonFnc::concatUrlStr($leardboard->cover);
        $leaderboardLang = $leardboard->lang;
        $list = $leardboard->games()->with([
            'circle' => function ($query) {
                $query->select(['id', 'game_id', 'circle_name', 'cover']);
            },
            'game_translation' => function ($query) use ($leaderboardLang) {
                $query->where('lang', $leaderboardLang);
```

```
            },
            'game_labels' => function ($query) use ($leaderboardLang) {
                $query->where('lang', $leaderboardLang);
            }
        ])->paginate($limit);
        $list = isset($list) ? $list->toArray() : [];
        $list['leardboard'] = $leardboard;
        CommonFnc::paginateReturnUnset($list);
        foreach ($list['data'] as $k => $v) {
            $game = [];
            $game['position'] = ($page - 1) * $limit + $k + 1;
            $game['id'] = $v['id'];
            $game['score'] = $v['score'];
            $game['name'] = empty($v['game_translation']['game_name']) ? $v['game_name'] :
$v['game_translation']['game_name'];
            $game['icon'] = CommonFnc::concatUrlStrAdmin($v['game_icon']);
            $game['circle_id'] = $v['circle']['id'] ?? 0;
            $game['circle'] = $v['circle'];
            $game['label_txt'] = implode('·', array_slice(array_column($v['game_labels'], 'label_name'), 0, 3));
            $game['rec_text'] = $v['game_translation']['rec_text'] ?? '';
            $list['data'][$k] = $game;
        }
        return ApiReturn::success($list);
    }

    //榜单设置
    public function leaderboard(Request $request)
    {
        $user_id = $request->get('user_id');
        return Game::leaderboard($user_id);
    }

    //今日游戏
    public function calendar(Request $request)
    {
        $page = $request->get('page', 0);
        $user_id = $request->get('user_id');
        return Game::calendar($page, $user_id);
    }

    //游戏日历更多游戏
    public function calendarMore(Request $request)
    {
        $page = $request->get('page', 0);
        $user_id = $request->get('user_id');
        return Game::calendarMore($page, $user_id);
    }

    /**
     * 热门游戏
     * @return Object
```

```
     */
    public function getHotGame(): object
    {
        try {
            $hotGame = ServiceConfig::query()->where(['field_name' => 'hot_recommend'])->value('value');
            $hotGame = explode('$$', $hotGame);
            $arr = [];
            foreach ($hotGame as $k => $v) {
                $arr[] = [
                    'sort' => $k + 1,
                    'game_name' => $v
                ];
            }
            return ApiReturn::success($arr);
        } catch (\Exception $e) {
            return ApiReturn::error();
        }
    }

    //游戏列表
    public function list(Request $request)
```

```
{
    $limit = $request->get('limit', 10);
    $label = $request->get('label');
    $user_id = $request->get('user_id');
    $category = $request->get('category');
    $order = $request->get('order', 'time');
    $lang = CommonFnc::lang();
    if($request->get('version_code_num') >= 339){
        return Game::newList($limit, $label, $user_id, $category, $order, $lang);
    }else{
        return Game::list($limit, $label, $user_id, $category, $order, $lang);
    }
}

//关注游戏
public function attention(Request $request)
{
    $user_id = $request->get('user_id');
    if (empty($user_id)) {
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $game_id = $request->get('game_id');
    if (empty($game_id)) {
        return ApiReturn::error('请选择要关注的游戏');
    }
    return Game::attention($user_id, $game_id);
}

//游戏筛选选择集
```

```
public function filterSelects()
{
    return Game::filterSelects();
}

//预约
public function reserve(Request $request)
{
    $user_id = $request->get('user_id');
    if (empty($user_id)) {
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $game_id = $request->get('game_id');
    if (empty($game_id)) {
        return ApiReturn::error('请选择要预约的游戏');
    }
    $status = $request->get('status', 1);
    if (!in_array($status, [1, 2])) {
        return ApiReturn::error('预约状态错误');
    }
    return Game::reserve($user_id, $game_id, $status);
}

//校验游戏是否存在
```

```
public function check(Request $request)
{
   $game_id = $request->get('game_id');
   if (empty($game_id)) {
      return ApiReturn::error('游戏不存在');
   }
   return Game::check($game_id);
}

//游戏商品详情
public function productList(Request $request)
{
   //游戏ID
   $gameId = $request->get('game_id',0);
   if(empty($gameId)){
      return ApiReturn::error('请选择要查看的游戏商品');
   }
   $list = Game::productList($gameId);
   return ApiReturn::success($list);
}
//热门免费游戏列表
public function hotFreeList(Request $request)
{
   $limit = $request->get('limit', 10);
   $lang = CommonFnc::lang();
   $list = Game::hotFreeList($limit,$lang);
```

```
   return ApiReturn::success($list);
}

//特惠栏
public function specialOffer(Request $request)
{
   $limit = $request->get('limit', 10);
   $type = $request->get('type', 'all');
   $lang = CommonFnc::lang();
   $version = $request->get('version_code_num');
   $list = Game::specialOffer($limit,$lang,$type,$version);
   return ApiReturn::success($list);
}

//特惠栏筛选标签集
public function specialOfferFilterSelects()
{
   $list = Game::specialOfferFilterSelects();
   return ApiReturn::success($list);
}
//限时免费游戏
public function getFreeTemporary(Request $request)
{
   $limit = $request->get('limit', 10);
   $lang = CommonFnc::lang();
   $user_id = $request->get('user_id');
   $list = Game::getFreeTemporary($limit,$user_id,$lang);
```

```
        return ApiReturn::success($list);
    }
    //添加心愿单
    public function addWishlist(Request $request)
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $game_id = $request->get('game_id');
        if (empty($game_id)) {
            return ApiReturn::error('请选择要添加的游戏');
        }
        $type = $request->get('type', 'steam');
        return Game::addWishlist($user_id,$game_id,$type);
    }
    //移除心愿单
    public function removeWishlist(Request $request)
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
```

```
        $game_id = $request->get('game_id');
        if (empty($game_id)) {
            return ApiReturn::error('请选择要移除的心愿单游戏');
        }
        $type = $request->get('type', 'steam');
        return Game::removeWishlist($user_id,$game_id,$type);
    }
    //心愿单列表
    public function wishlist(Request $request)
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $solo_userid = $request->get('solo_userid');

        $limit = $request->get('limit', 10);
        $lang = CommonFnc::lang();
        $type = $request->get('type', 'all');
        $list = Game::wishlist($user_id,$lang,$limit,$type,$solo_userid);
        return ApiReturn::success($list);
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\v1\GameLabel;
use App\Models\v1\UserGameLabel;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\Game;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class GameLabelController extends Controller
{
    /**
     * 游戏推荐标签
     * @return Object
     */
    public function recommendLabels(Request $request)
    {
        $lang = CommonFnc::lang();
        try {
            $type = $request->get('type',1);
            $recommendLabels = GameLabel::where(['is_recommend' => 1])
                ->where('lang',$lang)
                ->select('id as label_id','label_name')
                ->get()
                ->toArray();
            if($type == 2){
                $all = '全部';
                if($lang == 2){
                    $all = 'All';
                }elseif($lang == 3){
                    //日文
                    $all = 'すべて';
                }
                //在数组增加 全部
                array_unshift($recommendLabels,['label_id' => 0,'label_name' => $all]);
            }
            if($request->header('platform') == 2){
                return ApiReturn::success(['data' => $recommendLabels]);
            }else{
                return ApiReturn::success( $recommendLabels);
            }
        }catch (\Throwable $throwable) {
            Common::log('游戏推荐标签',$throwable);
            return ApiReturn::error();
```

```
        }
    }

    /**
     * 用户喜欢的游戏标签
     * @param Request $request
     * @return Object
     */
    public function likeLabels(Request $request)
    {
        try {
            $userId = $request->get('user_id');
            $labels = Game::likeLabels($userId);
            return ApiReturn::success($labels);
        }catch (\Throwable $throwable) {
            Common::log('用户喜欢的游戏标签',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 我的游戏--用户标签编辑
     * @param Request $request
     * @return Object
     */
    public function addLikeLabel(Request $request)
    {
        try {
            $userId = $request->post('user_id');
            $gameLabelIds = $request->post('game_label_ids',[]);
//            if(empty($gameLabelIds) || !is_array($gameLabelIds)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $oldGameLabelIds = UserGameLabel::where(['user_id' => $userId,'status' => 1])->pluck('game_label_id')->toArray();
            $del = array_diff($oldGameLabelIds,$gameLabelIds);
            $add = array_diff($gameLabelIds,$oldGameLabelIds);
            if(!empty($del)) {
                UserGameLabel::whereIn('game_label_id',$del)->where(['user_id' => $userId])->update(['status' => 0]);
            }
            if(!empty($add)) {
                foreach ($add as $v) {
                    if(empty($v)){
                        continue;
                    }
                    UserGameLabel::updateOrCreate([
                        'user_id' => $userId,
                        'game_label_id' => $v
                    ],['status' => 1]);
                  }
            }
            return ApiReturn::success();
        }catch (\Throwable $throwable) {
            Common::log('我的游戏--用户标签编辑',$throwable);
            return ApiReturn::error();
        }
    }

}

```

```
<?php

namespace App\Http\Controllers\api\v1;

```

```php
use App\Http\Controllers\Controller;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Information;
use Illuminate\Http\Request;

class InformationController extends Controller
{
    //头条
    public function headlines(Request $request): object
    {
        $id = $request->get('id',0);
        return Information::headlines($id);
    }
    //推荐
    public function recommended(Request $request): object
    {
        $catId = $request->get('cat_id',0);
        $id = $request->get('id',0);
        return Information::recommended($catId,$id);
    }
    //列表
    public function list(Request $request): object
    {
        $page = $request->get('page',1);
        $limit = $request->get('limit',2);
        $catId = $request->get('cat_id',0);
        $use_default_img = $request->get('use_default_img',1);
        $id = $request->get('id',0);
        return Information::list($limit,$catId,$use_default_img,$id);
    }
    //详情
    public function info(Request $request): object
    {
        $id = $request->get('id');
        if(empty($id)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $user_id = $request->get('user_id');
        return Information::info($id,$user_id);
    }
    public function catList(): object
    {
        return Information::catList();
    }
    //评论
```

```php
    public function addComment(Request $request){
        $commentable_id = $request->post('commentable_id');
        if(empty($commentable_id)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $content = $request->post('content');
        if(empty($content)){
            return ApiReturn::error('请输入内容');
        }
        $user_id = $request->post('user_id');
        return Information::addComment($user_id,$commentable_id,$content);
    }
    //回复
    public function addReply(Request $request){
        $commentable_id = $request->post('commentable_id');
        if(empty($commentable_id)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $content = $request->post('content');
        if(empty($content)){
            return ApiReturn::error('请输入内容');
        }
        $user_id = $request->post('user_id');
        return Information::addReply($user_id,$commentable_id,$content);
    }
    //评论列表
    public function commentList(Request $request): object
    {
        $information_id = $request->get('information_id');
        if(empty($information_id)){
            return ApiReturn::success();
        }
        $limit = $request->get('limit',10);
        return Information::commentList($information_id,$limit);

    }
    //点赞
    public function like(Request $request){
        $user_id = $request->post('user_id');
        if(empty($user_id)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $information_id = $request->post('information_id');
        if(empty($information_id)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        return Information::like($user_id,$information_id);
    }
    //收藏
    public function collect(Request $request){
```

```
        $user_id = $request->post('user_id');
        if(empty($user_id)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $information_id = $request->post('information_id');
        if(empty($information_id)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        return Information::collect($user_id,$information_id);
    }
    //分享
    public function share(Request $request){
        $user_id = $request->post('user_id');
        if(empty($user_id)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $information_id = $request->post('information_id');
        if(empty($information_id)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        return Information::share($user_id,$information_id);
    }
    //回复列表
    public function replyList(Request $request): object
    {
        $comment_id = $request->get('comment_id');
        if(empty($comment_id)){
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $limit = $request->get('limit',10);
        return Information::replyList($comment_id,$limit);
    }
    //今日要闻
    public function getTodayInformation(): object
    {
        return Information::getTodayInformation();
    }
    //新游预约
    public function newGameReservation(): object
    {
        return Information::newGameReservation();
    }
    //行业资讯
    public function getIndustry(): object
    {
        return Information::getIndustry();
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\v1\UserDiamondLog;
use App\Models\v1\UserInviteLog;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\Invite;
use App\Service\v1\RedisKey;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Redis;

class InviteController extends Controller
{
    /**
     * 邀请页面接口
     * @param Request $request
     * @return Object
     */
    public function invitePage(Request $request)
    {
        $userId = $request->get('user_id');
        try {
            $invitePage = Invite::invitePage($userId);
            return ApiReturn::success($invitePage);
        }catch (\Throwable $throwable) {
            Common::log('邀请页面接口',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 邀请页面接口
     * @param Request $request
     * @return Object
     */
    public function bindUser(Request $request)
    {
        $userId = $request->post('user_id');
        $inviteCode = $request->post('invite_code');
        if (!$inviteCode) return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        try {
            $rs = Invite::bindUser($userId, $inviteCode);
            if (!$rs['status']) return ApiReturn::rdata($rs['data']);
            else return ApiReturn::success();
        }catch (\Throwable $throwable) {
```

```
            Common::log('绑定用户',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 好友贡献
     * @param Request $request
     * @return Object
     */
    public function inviteContributionInfo(Request $request)
    {
        $userId = $request->get('user_id');
        try {
            $data = [
                'people_total' => UserInviteLog::query()->where(['pid' => $userId])->count(),
                'diamond_total' => UserDiamondLog::query()->where(['user_id' => $userId,'type' => 22])->sum('diamond')
            ];
            return ApiReturn::success($data);
        }catch (\Throwable $throwable) {
            Common::log('好友贡献错误',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 佣金汇总
     * @param Request $request
     * @return Object
     */
    public function commissionLogs(Request $request)
    {
        $page = (int)$request->get('page',1);
        $limit = (int)$request->get('limit',10);
        $userId = $request->get('user_id');
        try {
            $page = ($page - 1) * $limit;
            $commissionLogs = UserDiamondLog::query()->with(['user' => function($query) {
                $query->select('id','username');
            }])->where(['user_id' => $userId,'type' => 30])
                ->select('diamond','describe','source_id')
                ->limit($limit)->offset($page)->get();
            return ApiReturn::success($commissionLogs);
        }catch (\Throwable $throwable) {
            Common::log('佣金汇总错误',$throwable);
            return ApiReturn::error();
        }
    }

```

```
    /**
     * 邀请规则
     * @return Object
     */
    public function inviteRule()
    {
        $awards = CommonFnc::getServiceConfigData(RedisKey::INVITE_CONFIG);
        $rule = empty($awards) ? '' : json_decode($awards,true)['rule'];
        return ApiReturn::success(['rule' => $rule]);
    }

    public function inviteBanner()
    {
        $arr = [
            ['name' => "我***尚    提现",'money' => "707.59"],
            ['name' => "安***远    提现",'money' => "505.03"],
            ['name' => "曹***哥    提现",'money' => "323.38"],
            ['name' => "浪***会    提现",'money' => "403.86"],
            ['name' => "毋***山    提现",'money' => "350.71"],
            ['name' => "心***了    提现",'money' => "508.53"],
['name' => "乡***族    提现",'money' => "601.82"],
['name' => "你***唯    提现",'money' => "1101.9"],
['name' => "老***6    提现",'money' => "450.28"],
['name' => "去***卡    提现",'money' => "600.79"],
['name' => "勿***心    提现",'money' => "252.05"],
['name' => "剑***子    提现",'money' => "805.72"],
['name' => "无***美    提现",'money' => "414.34"],
['name' => "奋***小    提现",'money' => "453.31"],
['name' => "拾***睡    提现",'money' => "1061.4"],
['name' => "不***菜    提现",'money' => "303.59"],
['name' => "云***会    提现",'money' => "300.52"],
['name' => "清***清    提现",'money' => "506.18"],
['name' => "青***的    提现",'money' => "300.88"],
['name' => "我***字    提现",'money' => "351.01"],
['name' => "翔***作    提现",'money' => "404.17"],
['name' => "姓***兄    提现",'money' => "300.61"],
['name' => "挺***初    提现",'money' => "502.26"],
['name' => "睿***欢    提现",'money' => "402.83"],
['name' => "无***何    提现",'money' => "302.88"],
['name' => "肚***了    提现",'money' => "222.19"],
['name' => "我***逼    提现",'money' => "808.83"],
['name' => "江***月    提现",'money' => "702.41"],
['name' => "张***爸    提现",'money' => "504.42"],
['name' => "可***加    提现",'money' => "505.82"],
['name' => "春***了    提现",'money' => "703.07"],
['name' => "小***老    提现",'money' => "250.88"],
['name' => "伟***克    提现",'money' => "301.67"],
['name' => "初***猫    提现",'money' => "350.98"],
['name' => "琼***怕    提现",'money' => "252.42"],
['name' => "戴***特    提现",'money' => "402.23"],
```

```
['name' => "家***钱    提现",'money' => "524.36"],
['name' => "你***坏    提现",'money' => "502.93"],
['name' => "大***江    提现",'money' => "363.73"],
['name' => "老***口    提现",'money' => "754.99"],
['name' => "小***1    提现",'money' => "553.92"],
['name' => "张***g    提现",'money' => "554.72"],
['name' => "禾***æ    提现",'money' => "454.03"],
['name' => "北***王    提现",'money' => "909.58"],
['name' => "心***白    提现",'money' => "252.78"],
['name' => "冷***会    提现",'money' => "853.75"],
['name' => "上***魔    提现",'money' => "500.04"],
['name' => "自***全    提现",'money' => "303.52"],
['name' => "小***★    提现",'money' => "261.55"],
['name' => "丑***罗    提现",'money' => "504.37"],
['name' => "十***髅    提现",'money' => "301.08"],
['name' => "小***d    提现",'money' => "230.47"],
['name' => "远***英    提现",'money' => "301.37"],
['name' => "星***入    提现",'money' => "551.56"],
['name' => "仔***无    提现",'money' => "601.31"],
['name' => "一***闯    提现",'money' => "504.06"],
['name' => "火***木    提现",'money' => "404.59"],
['name' => "幻***的    提现",'money' => "210.71"],
['name' => "长***这    提现",'money' => "300.14"],
            ['name' => "修***灵     提现",'money' => "510.37"],
            ['name' => "张***的     提现",'money' => "503.91"],
            ['name' => "陈***他     提现",'money' => "500.58"],
            ['name' => "神***不     提现",'money' => "362.02"],
            ['name' => "刘***解     提现",'money' => "650.61"],
            ['name' => "暑***巅     提现",'money' => "700.17"],
            ['name' => "一***唯     提现",'money' => "300.13"],
            ['name' => "浅***倾     提现",'money' => "260.19"],
            ['name' => "他***好     提现",'money' => "1007.3"],
            ['name' => "飞***甲     提现",'money' => "273.19"],
            ['name' => "王***耀     提现",'money' => "322.22"],
            ['name' => "猜***是     提现",'money' => "805.15"],
            ['name' => "瑶***静     提现",'money' => "252.33"],
            ['name' => "小***加     提现",'money' => "301.12"],
            ['name' => "陈***老     提现",'money' => "403.43"],
            ['name' => "你***系     提现",'money' => "804.22"],
            ['name' => "调***儿     提现",'money' => "301.04"],
            ['name' => "了***红     提现",'money' => "450.84"],
            ['name' => "鸣***永     提现",'money' => "504.13"],
            ['name' => "小***一     提现",'money' => "302.55"],
            ['name' => "咯***日     提现",'money' => "252.19"],
            ['name' => "宇***带     提现",'money' => "302.34"],
            ['name' => "今***开     提现",'money' => "500.25"],
            ['name' => "闫***的     提现",'money' => "250.91"],
            ['name' => "疯***里     提现",'money' => "252.78"],
            ['name' => "实***3     提现",'money' => "312.02"],
            ['name' => "哥***下     提现",'money' => "402.02"],
```

```
            ['name' => "快***诞     提现",'money' => "302.04"],
            ['name' => "为***出     提现",'money' => "273.32"],
            ['name' => "机***女     提现",'money' => "354.23"],
            ['name' => "宝***油     提现",'money' => "403.47"],
            ['name' => "段***段     提现",'money' => "501.91"],
            ['name' => "外***世     提现",'money' => "303.05"],
            ['name' => "李***凤     提现",'money' => "251.99"],
            ['name' => "全***阿     提现",'money' => "453.03"],
        ];
        return ApiReturn::success($arr);
    }


}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Http\Requests\NewActivityCenterRequest;
use App\Models\v1\ServiceConfig;
use App\Models\v1\UserUserRole;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Article;
use App\Service\v1\Code;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\NewTask;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Throwable;

/**
 * 新任务中心
 */
class NewActivityCenterController extends Controller
{
    /**
     * 获取Tab分类设置的奖励
     * @param Request $request
     * @return object
     */
    public function getTaskTabList(Request $request): object
    {
        //获取用户Id
        $userId = $request->get('user_id');
        return NewTask::getTaskTabList($userId);
    }

    /**
     * 活动列表
     * @param Request $request
     * @return object
     */
    public function getTaskList(Request $request): object
    {
        //获取用户Id
        $userId = $request->get('user_id');
        //获取tap分类
        $taskTabId = $request->get('task_tab_id', 0);
        $limit = (int)$request->get('limit', 10);
        $page = (int)$request->get('page', 1);
        $page = ($page - 1) * $limit;
```

```
        $taskStatus = (int)$request->get('task_status', 0);
        $time = $request->server('REQUEST_TIME');
        return NewTask::getTaskList($userId, $taskTabId, $time, $page, $limit, $taskStatus);
    }

    /**
     * 任务详情
     * @param Request $request
     * @return Object
     */
    public function taskInfo(Request $request): object
    {
        $taskId = intval($request->get('task_id', 0));
        $userId = $request->get('user_id', 0);
```

```
//        if(empty($userId)){
//            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
//        }
        if ($taskId <= 0) return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        $platform = $request->header('Platform');
        return NewTask::taskInfo($taskId, $userId,$platform);
    }

    //参与任务
    public function participatingTask(NewActivityCenterRequest $request): object
    {
        Log::info('用户参与任务入参', $request->all());
//        $is_creative_camp = $request->post('is_creative_camp', 0);
        $is_creative_camp = 0;
        $param = $request->post();
        if (empty($param['type'])) {
            return ApiReturn::error('任务参与类型错误');
        }
        $param['action'] = $request->post('action', Code::ACTION_ADD);
        if ($param['type'] == Code::ARTICLE_TYPE_MAP['evaluating']) {
            $param['type'] = Code::ARTICLE_TYPE_MAP['evaluate'];
        } else {
            $param['type'] = intval($param['type']);
        }
        if ($param['type'] == Code::ARTICLE_TYPE_MAP['evaluate'] || $param['type'] == Code::ARTICLE_TYPE_MAP['evaluating']) {
            if ($is_creative_camp <= 0) {
                $request->validate('participatingTask');
            }
            if(empty($param['game_score'])){
                return ApiReturn::error('请输入游戏评分');
            }
        } else {
            if ($is_creative_camp > 0) {
                return ApiReturn::error('此任务类型暂时不支持创作营');
            }
        }
```

```
        $ip = CommonFnc::getRealIP();
        try {
            $param['action'] = $request->post('action', Code::ACTION_ADD);
            $param['status'] = $request->post('status', 0);
            Log::info('参与任务==入口参数=====' ,$param);
            // 验证 设置
            $err = NewTask::checkParticipatingTask($param, $is_creative_camp,$ip);
            if ($err) {
                Log::info('参与任务==校验失败=====' ,$err);
                return ApiReturn::rdata($err);
            }
            // 插入到 article 表的数据
            $param['data'] = [
                'cover' => $param['cover'],
                'game_score' => empty($param['game_score']) ? 0 : $param['game_score'],
                'check_status' => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
                'content' => $param['content'],
                // article 标题 截取
                'title' => '',
                'text_content' => $param['text_content'],
                'content_len' => $param['content_len'],
                'home_page' => $param['home_page'],
                'label_list' => $param['label_list'],
                'abstract' => $param['abstract'],
                'user_id' => $param['user_id'],
                'creator_source_id' => $param['creator_source_id'],
                'task_article_status' => $param['task_article_status'],
                'state' => 1,
                'task_tab_id' => $param['task_tab_id']
            ];
            if ($param['action'] == Code::ACTION_ADD) {
                // 新增需要的数据 (修改的时候不可更改的数据)
                $param['data'] = array_merge($param['data'], [
                    'parent_id' => empty($param['parent_id']) ? 0 : $param['parent_id'],
                    'game_id' => $param['game_id'],
                    'task_id' => $param['task_id'],
                    'circle_id' => $param['circle_id'],
                    'type' => $param['type'],
                    'task_tab_id' => $param['task_tab_id'],
                    'task_article_status' => $param['task_article_status'],
                    'status' => $param['status'],
                    'is_writer_task' => $param['is_writer_task'],
                    'creator_source_id' => $param['creator_source_id']
                ]);
            }
            Log::info('参与任务==进入数据库参数=====' , $param);
            return Article::addNewArticle($param,$ip);
        } catch (Throwable $throwable) {
            Common::log('发布动态', $throwable);
            return ApiReturn::error();
```

```
    }
}

/**
 * 新增/编辑收集地址
 * @param NewActivityCenterRequest $request
 * @return Object
 */
public function editCollectPlatformUrl(NewActivityCenterRequest $request): object
{
    $params = $request->post('activity_center_platform_url', []);
    if (is_string($params)) $params = json_decode($params, true); // ios 传惨问题
    $articleId = (int)$request->get('article_id', 0);
    if (empty($params)) {
        return ApiReturn::rdata(ApiCode::PARAM_ERROR);
    }
    $type = $request->get('type', 0);
    $userId = $request->post('user_id');
    return NewTask::editCollectPlatformUrl($userId, $articleId, $params, $type);
}
```

```

    //放弃任务
    public function abortMission(NewActivityCenterRequest $request): object
    {
        $userId = $request->post('user_id');
        $articleId = (int)$request->get('article_id', 0);
        if (empty($articleId)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        Log::info('放弃任务==入口参数==article_id=' . $articleId . '==user_id==' . $userId);
        return NewTask::abortMission($userId, $articleId);
    }

    /**
     * 获取创作营内容
     * @param Request $request
     * @return Object
     */
    public function getChannelContent(Request $request): object
    {
        $taskId = $request->get('task_id');
        if (empty($taskId)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $userId = $request->get('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return NewTask::getChannelContent($taskId, $userId);
    }
```

```

    /**
     * 获取用户参与任务信息
     * @param Request $request
     * @return Object
     */
    public function getUserTaskInfo(Request $request): object
    {
        $user_id = $request->get('user_id', 0);
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $task_id = $request->get('task_id', 0);
        if (empty($task_id)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $platform = $request->header('Platform');
        return NewTask::getUserTaskInfo($user_id, $task_id,$platform);
    }

    /**
     * 用户任务列表
     * @param Request $request
     * @return Object
     */
    public function getUserTaskList(Request $request): object
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $limit = (int)$request->get('limit', 10);
        $page = (int)$request->get('page', 1);
        return NewTask::getUserTaskList($user_id, $page, $limit);
    }

    /**
     * 活动中心用户信息
     * @param Request $request
     * @return Object
     */
    public function getUserInfo(Request $request): object
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return NewTask::getUserInfo($user_id);
    }
    //检测非鉴游团用户是否显示任务
```

```
    public function userShowTask(Request $request): object
    {
        $data = [
            'show' => false
        ];
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            //获取游客用户是否显示任务入口配置
            if(ServiceConfig::query()->where('tab', 4)->where('field_name', 'is_show_task_for_guests')->value('value')){
                $data['show'] = true;
            }
            return ApiReturn::success($data);
        }
        //获取后台系统配置 如果后台设置非鉴游团用户显示，则对用户角色进行判断非鉴游团直接返回 false
        if (!ServiceConfig::query()->where('tab', 4)->where('field_name', 'show_task_functionality')->value('value')) {
            if (!UserUserRole::query()->where('user_id', $user_id)->where('user_role_id', 6)->first()) {
                return ApiReturn::success($data);
            }
        }
        $data['show'] = true;
        return ApiReturn::success($data);
    }

    /**
     * 检测用于用户参与活动权限
     * @param Request $request
     * @return Object
     */
    public function checkUserParticipationTask(Request $request): object
    {
        $userId = $request->get('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $taskId = $request->get('task_id');
        if (empty($taskId)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        return NewTask::checkUserParticipationTask($userId, $taskId);
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\Notification;
use App\Notifications\UserNotification;
use App\Service\v1\ApiReturn;
use App\Service\v1\UserNotifications;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

/**
 * 消息通知
 */
class NotificationController extends Controller
{
    public function categories(Request $request)
    {
        $user_id = $request->get('user_id');
        return UserNotifications::categories($user_id);
    }
    //消息列表
    public function list(Request $request)
    {
        $user_id = $request->get('user_id');
        $limit = $request->get('limit',10);
        $category = $request->get('category',0);
       return UserNotifications::list($user_id,$category,$limit);
    }
    //读取单条
    public function readOne(Request $request)
    {
        $user_id = $request->post('user_id');
        $notify_id  = $request->post('notify_id',0);
        return UserNotifications::readOne($user_id,$notify_id);
    }
    //读取所用通知
    public function readAll(Request $request)
    {
        $user_id = $request->post('user_id');
        $category = $request->get('category',0);
        if(empty($category)){
            return ApiReturn::success();
        }
        return UserNotifications::readAll($user_id,$category);
    }

    /**
     * 获取用户 是否有新消息
```

```
     * @param Request $request
     * @return Object
     */
    public function getNewUserNotic(Request $request): object
    {
        try {
            $uid = $request->get('user_id');
            $rdata = ['new' => 0];
            if($uid > 0){
                $rdata['new'] = Notification::query()->where('notifiable_id',$uid)->whereNull('read_at')->count();
            }
            return ApiReturn::success($rdata);
        } catch (\Exception $e) {
            return ApiReturn::success($rdata);
        }
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Events\OrderEvent;
use App\Http\Controllers\Controller;
use App\Models\v1\Cdk;
use App\Models\v1\Order;
use App\Models\v1\Product;
use App\Models\v1\ProductDiscountCoupon;
use App\Models\v1\User;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\OrderService;
use App\Service\v1\SoloCode;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;

class OrderController extends Controller
{
    /**
     * 钻石购买商品(积分支付)
     * @param Request $request
     * @return Object
     */
    public function diamondBuyProduct(Request $request)
    {
        try {
            $productId = intval($request->post('product_id',0));
            $userAddressId = intval($request->post('user_address_id',0));
            $userId = $request->post('user_id');
            if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $remark = strval($request->post('remark',''));
            $productInfo = Product::where(['id' => $productId])
                ->select('id','name','cover','product_type_id','price','reference_price','diamond','type','pay_type',
                    'quantity_sold','usable_inventory','total_inventor','cdk_status')->first();
            if(empty($productInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            if($productInfo->usable_inventory <= 0) return
ApiReturn::error(ApiCode::INVENTORY_ERROR['msg'],ApiCode::INVENTORY_ERROR['code']);
            if($productInfo->pay_type == 1) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
            if($productInfo->type == 1 && $userAddressId <= 0) return
ApiReturn::error(ApiCode::ADDRESS_ERROR['msg'],ApiCode::ADDRESS_ERROR['code']);
            $userInfo = User::query()->where(['id' => $userId])->first(['id','diamond_num','freezing_diamond']);
            if(($userInfo->diamond_num - $userInfo->freezing_diamond) < $productInfo->diamond) return
ApiReturn::error(ApiCode::DISCOUNT_ERROR['msg'],ApiCode::DISCOUNT_ERROR['code']);
            DB::beginTransaction();
            $ckdInfo = OrderService::pushCdk($productInfo->type,$productInfo->id,$userInfo->id,$productInfo->cdk_status);
            if(empty($ckdInfo['cdk'])) {
                DB::rollBack();
                return ApiReturn::error(ApiCode::CDK_ERROR['msg'],ApiCode::CDK_ERROR['code']);
            }
```

```
            $res = OrderService::diamondBuyVirtualProduct($userInfo,$userAddressId,$productInfo,$remark,$ckdInfo);
            if($res){
                DB::commit();
                return ApiReturn::success();
            }
            DB::rollBack();
            return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
        } catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('钻石购买商品',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 钱+积分支付
     * @param Request $request
     * @return Object
     */
    public function moneyBuyProduct(Request $request)
    {
        try {
            $productId = intval($request->post('product_id',0));
            $userAddressId = intval($request->post('user_address_id',0));
            $productDiscountCouponId = intval($request->post('product_discount_coupon_id',0));
            $type = intval($request->post('type',0));
            $userId = $request->post('user_id');
            if($productId <= 0 || !isset(SoloCode::WECHATPAY_TYPE[$type])) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $productInfo = Product::where(['id' => $productId])
                ->select('id','name','cover','product_type_id','price','reference_price','diamond','type','pay_type',
                    'quantity_sold','usable_inventory','total_inventor','discount_coupon_id','cdk_status')->first();
            if(empty($productInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            if($productInfo->usable_inventory <= 0) return ApiReturn::error(ApiCode::INVENTORY_ERROR['msg'],ApiCode::INVENTORY_ERROR['code']);

            if($productInfo->pay_type == 0) return ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);

            if($productInfo->type == 1 && $userAddressId <= 0) return ApiReturn::error(ApiCode::ADDRESS_ERROR['msg'],ApiCode::ADDRESS_ERROR['code']);
            if($productInfo->type == 0 && empty(Cdk::where(['product_id' => $productId,'user_cdk_id' => 0])->value('cdk'))) return ApiReturn::error(ApiCode::CDK_ERROR['msg'],ApiCode::CDK_ERROR['code']);
            if($productDiscountCouponId > 0) {
                $discountCouponInfo = ProductDiscountCoupon::where([
                    'id' => $productDiscountCouponId,
                    'discount_coupon_id' => $productInfo->discount_coupon_id,
                    'is_use' => 1
                ])->select('id','discount_coupon_price','is_use')->first();
                if(empty($discountCouponInfo)) return ApiReturn::error(ApiCode::DISCOUNT_COUPON_ERROR['msg'],ApiCode::DISCOUNT_COUPON_ERROR['code']);
                $productInfo->discount_coupon_price = $discountCouponInfo->discount_coupon_price;
            }else{
                $discountCouponInfo = '';
                $productInfo->discount_coupon_price = 0;
```

```
            }
            DB::beginTransaction();
            $remark = strval($request->post('remark',''));
            $res = OrderService::moneyBuyVirtualProduct($userId,$userAddressId,$productInfo,$discountCouponInfo,$type,$remark);
            if(empty($res)) {
                DB::rollBack();
                return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            }else if($res['code'] == 200){
                DB::commit();
                OrderEvent::dispatch($userId,SoloCode::OBLIGATION_TITLE,SoloCode::OBLIGATION_EXPLAIN);
                return ApiReturn::success($res['data']);
            }
            DB::rollBack();
            return ApiReturn::error($res['msg'],$res['code']);
        } catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('money+积分购买商品',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 订单列表
     * @param Request $request
     * @return Object
     */
    public function orders(Request $request)
    {
        try {
//            $provinceType = intval($request->get('province_type',0));
            $status = intval($request->get('status',0));
            if(!in_array($status,[0,1,2,3,4,5,6,7])) return 
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = $request->get('user_id');
            $page = intval($request->get('page',1));
            $limit = intval($request->get('limit',10));
            $page = ($page - 1) * $limit;
            $where = ['user_id' => $userId];
            if($status > 0) $where['status'] = $status;
            $field = [
                'id','order_num','total_diamond','discounts_total_price','total_price','pay_type',
                'status','created_at','product_pay_type','refuse_reason'
            ];
            $list = OrderService::orders($where,$field,$page,$limit);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('订单列表',$throwable);
            return ApiReturn::error();
        }
    }

```

```
    /**
     * 订单详情
     * @param Request $request
     * @return Object
     */
    public function orderInfo(Request $request)
    {
        try {
            $orderId = intval($request->get('order_id',0));
            if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = intval($request->get('user_id',0));
            $info = OrderService::orderInfo($orderId,$userId);
            return ApiReturn::success($info);
        } catch (\Throwable $throwable) {
            Common::log('订单详情',$throwable);
            return ApiReturn::error($throwable->getMessage());
        }
    }

    /**
     * 确认完成订单
     * @param Request $request
     * @return Object
     */
    public function finishOrder(Request $request)
    {
        try {
            $orderId = intval($request->post('order_id',0));
            if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = $request->post('user_id');
            $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId,'status' => 3])->first();
            if(empty($orderInfo)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
            $orderInfo->status = 4;
            $orderInfo->completion_time = time();
            $orderInfo->save();
            OrderEvent::dispatch($userId,SoloCode::FINISH_TITLE,SoloCode::FINISH_EXPLAIN);
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('确认完成订单',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 删除订单
     * @param Request $request
     * @return Object
     */
    public function deleteOrder(Request $request)
    {
```

```
        try {
            $orderId = intval($request->post('order_id',0));
            if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = intval($request->post('user_id',0));
            $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])->first();
            if(empty($orderInfo)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
            OrderService::deleteOrder($orderInfo);
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('删除订单',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 取消订单
     * @param Request $request
     * @return Object
     */
    public function cancelOrder(Request $request)
    {
        try {
            $orderId = intval($request->post('order_id',0));
            if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = intval($request->post('user_id',0));
            $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])->first();
            if(empty($orderInfo)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
            if($orderInfo->status == 1) {
                $orderInfo->status = 7;
                $orderInfo->save();
                return ApiReturn::success();
            }
            return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        } catch (\Throwable $throwable) {
            Common::log('取消订单',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 继续支付
     * @param Request $request
     * @return Object
     */
    public function continuePay(Request $request)
    {
        try {
            $orderId = intval($request->post('order_id',0));
            $type = intval($request->post('type',1));
```

```
            if($orderId <= 0 || !isset(SoloCode::WECHATPAY_TYPE[$type])) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = intval($request->post('user_id',0));
            $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])->first();
            if(empty($orderInfo) || $orderInfo->status != 1) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
            $response = OrderService::continuePay($orderInfo,$type);
            return ApiReturn::success($response['data']);
        } catch (\Throwable $throwable) {
            Common::log('继续支付',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 微信app支付回调
     * @param Request $request
     * @return \Illuminate\Http\JsonResponse
     */
    public function wechatOrderNotify(Request $request)
    {
        try {
            $wechatHeader = [
                'wechatpay_signature' => $request->header('wechatpay-signature'),
                'wechatpay_timestamp' => $request->header('wechatpay-timestamp'),
                'wechatpay_serial' => $request->header('wechatpay-serial'),
                'wechatpay_nonce' => $request->header('wechatpay-nonce'),
            ];
            $inBody = $request->getContent();
            Log::info('微信App支付回调接受参数',[$wechatHeader,$inBody]);
            DB::beginTransaction();
            $resource = OrderService::wechatOrderNotify($wechatHeader,$inBody);
            if($resource) {
                DB::commit();
                return response()->json(['code' => 'SUCCESS','message' => '成功']);
            }
            DB::rollBack();
            return response()->json(['code' => 'ERROR','message' => '失败']);
        } catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('微信支付回调',$throwable);
            return response()->json(['code' => 'ERROR','message' => '失败']);
        }
    }

    /**
     * 退款
     * @param Request $request
     * @return Object
     */
    public function orderRefund(Request $request)
    {
```

```
        try {
            $orderId = intval($request->post('order_id',0));
            if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = intval($request->post('user_id',0));
            $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])->first();
            if(empty($orderInfo) || !in_array($orderInfo->status,[2,3,4]) || !empty($orderInfo->refund_order_num)) return 
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
            $response = OrderService::orderRefund($orderInfo);
            if($response['code'] == 200) return ApiReturn::success();
            return ApiReturn::error($response['msg'],$response['code']);
        } catch (\Throwable $throwable) {
            Common::log('退款',$throwable);
            return ApiReturn::error();
        }
    }

}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
```

```php
use App\Models\Contact;
use App\Service\v1\ApiReturn;
use App\Service\v1\CommonFnc;
use App\Service\v1\Feishu;
use Illuminate\Http\Request;
use Mews\Captcha\Facades\Captcha;

class PcHomeController extends Controller
{
    //提交联系我们信息
    public function submitContact(Request $request): object
    {
        $captcha = $request->post('captcha');
        if (empty($captcha)) {
            return ApiReturn::error('请输入验证码');
        }
        $key = $request->post('key');
        if (empty($key)) {
            return ApiReturn::error('请输入验证码key');
        }
        $name = $request->post('name');
        if (empty($name)) {
            return ApiReturn::error('请填写姓名');
        }
        $contact = $request->post('contact');
        if (empty($contact)) {
            return ApiReturn::error('请填写联系方式');
        }
        $user_id = $request->post('user_id', 0);
        $isCaptchaValid = captcha_api_check($captcha, $key);
        if ($isCaptchaValid) {
            try {
                // CAPTCHA 验证成功，执行其他逻辑
                Contact::query()->create([
                    'name' => $name,
                    'contact' => $contact,
                    'ip' => CommonFnc::getRealIP(),
                    'user_id' => $user_id
                ]);
                $content = [
                    'msg_type' => 'interactive',
                    'card' => [
                        'header' => [
                            'title' =>[
                                'content' => '意向客户',
```

```php
                                'tag' => 'plain_text'
                            ]
                        ],
                        'elements' => [
                            [
                                'tag' => "div",
                                'text' => [
                                    'content' => '**姓名：**' . $name . '  **联系方式：**' . $contact,
                            "tag" => "lark_md"
                        ]
                    ],
                    [
                        'actions' => [
                            [
                                'tag' => 'button',
                                'text' => [
                                    'content' => '点击查看',
                                    'tag' => 'lark_md'
                                ],
                                'url' => env('SOLO_ADMIN_HOST').'admin/contacts',
                                'type' => 'default',
                                'value' => []
                            ]
                        ],
                        "tag" => "action"
                    ]
                ]
            ]
        ];
        Feishu::sendWebhookCrowdRobotMessage($content);
        return ApiReturn::success('提交成功');
    } catch
    (\Exception $e) {
        return ApiReturn::error('提交失败');
    }
} else {
    // CAPTCHA 验证失败，返回错误提示
    return ApiReturn::error('验证码错误');
}
}
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\ProductType;
use App\Models\v1\DiscountCoupon;
use App\Models\v1\Product;
use App\Models\v1\ProductDiscountCoupon;
use App\Models\v1\User;
use App\Models\v1\UserDiamondLog;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\ProductService;
use App\Service\v1\RedisKey;
use App\Service\v1\SoloCode;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Redis;
use Predis\Client;

class ProductController extends Controller
{
    /**
     * 商品分类
     * @return Object
     */
    public function productTypes()
    {
        try {
            $list =  Cache::store('common_redis')->rememberForever(RedisKey::PRODUCT_TYPE,function (){
                $list = ProductType::query()->where('status',1)->select('id','name')->get();
                return json_encode($list,JSON_UNESCAPED_UNICODE);
            });
            return ApiReturn::success(json_decode($list,true));
        } catch (\Throwable $throwable) {
            Common::log('商品分类',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 商品列表
     * @param Request $request
     * @return Object
     */
    public function products(Request $request)
    {
```

```
        try {
            $productTypeId = intval($request->get('product_type_id',0));
            $page = intval($request->get('page',1));
            $limit = intval($request->get('limit',10));
            $page = ($page - 1) * $limit;
            $list = ProductService::products($productTypeId,$page,$limit);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('商品列表',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 商品详情
     * @param Request $request
     * @return Object
     */
    public function productInfo(Request $request)
    {
        try {
            $productId = intval($request->get('product_id',0));
            if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $list = ProductService::productInfo($productId);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('商品详情',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 兑换优惠券
     * @param Request $request
     * @return Object
     */
    public function exchangeDiscountCoupon(Request $request)
    {
        try {
            $productId = intval($request->post('product_id',0));
            if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = $request->post('user_id');
            DB::beginTransaction();
            $productInfo = Product::where(['id' => $productId])
                ->select('diamond','pay_type','discount_coupon_id')->first();
            $userInfo = User::query()->where(['id' => $userId])->first(['id','diamond_num','freezing_diamond']);
            if($productInfo->pay_type !== SoloCode::PAY_TYPE_DISCOUNT_MONEY) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $discountCoupon = DiscountCoupon::where(['id' =>
$productInfo->discount_coupon_id])->select('id','discount_coupon_price','discount_coupon_num')->first();
            if($discountCoupon->discount_coupon_price > ($userInfo->diamond_num - $userInfo->freezing_diamond)) return
ApiReturn::error(ApiCode::DISCOUNT_ERROR['msg'],ApiCode::DISCOUNT_ERROR['code']);
```

```
            if($discountCoupon->discount_coupon_num <= 0) return
ApiReturn::error(ApiCode::INVENTORY_ERROR['msg'],ApiCode::INVENTORY_ERROR['code']);
            $discountCoupon->discount_coupon_num -= 1;
            $discountCoupon->save();
            $userInfo->diamond_num = $userInfo->diamond_num - $productInfo->diamond;
            $userInfo->save();
            ProductDiscountCoupon::create([
                'discount_coupon_id' => $productInfo->discount_coupon_id,
                'user_id' => $userId,
                'discount_coupon_price' => $discountCoupon->discount_coupon_price,
                'diamond' => $productInfo->diamond
            ]);
            $data = [
                'user_id' => $userId,
                'source_id' => $productId,
                'type' => 7,
                'describe' => '购买优惠券',
                'diamond' => $productInfo->diamond,
                'balance' => $userInfo->diamond_num,
                'action' => 2
            ];
            UserDiamondLog::create($data);
            DB::commit();
            User::where('id',$userId)->searchable();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('兑换优惠券',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 确认订单页
     * @param Request $request
     * @return Object
     */
    public function verifyOrder(Request $request)
    {
        try {
            $productId = intval($request->get('product_id',0));
            if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $userId = $request->get('user_id');
            $info = ProductService::verifyOrder($productId,$userId);
            return ApiReturn::success($info);
        } catch (\Throwable $throwable) {
            Common::log('确认订单页',$throwable);
            return ApiReturn::error();
        }
    }
```

```
}

```

```
<?php

namespace App\Http\Controllers\api\v1\Questionnaire;

use App\Http\Controllers\Controller;
use App\Http\Requests\AnswerRequest;
use App\Models\v1\Questionnaires\Questionnaire;
use App\Models\v1\Questionnaires\QuestionnaireQuestionOption;
use App\Models\v1\Questionnaires\QuestionnaireUserAnswer;
use App\Models\v1\Questionnaires\QuestionnaireUserAnswerOption;
use App\Models\v1\Questionnaires\QuestionnaireUserAnswerQuestion;
use App\Models\v1\Questionnaires\QuestionnaireUserRole;
use App\Models\v1\User;
use App\Models\v1\UserUserRole;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\BaiDuSdk;
use App\Service\v1\Code;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\RedisKey;
use App\Service\v1\UserNotifications;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Redis;

class QuestionnaireController extends Controller
{

    /**
     * 问卷页面
     * @catalog
     * @title 问卷页面
     * @description 描述信息
     * @method get
     * @url /api/questionnaire/questionnaireInfo
     * @param questionnaire_id 选填 int 问卷id
     * @return {"code": 200,"msg": "OK","data": {"id": 1,"type": 0,"title": "标题1","sub_title": "副标题1","submit_num": 2,"submit_message": "温柔温柔温柔我","description": "","start_time": "2022-07-08 17:48:36","end_time": "2022-07-10 17:48:36","question": [{"id": 8,"questionnaire_id": 1,"question_type": 1,"is_required": 1,"is_random": 1,"fraction": 11,"title": "timu","tips": "提示","upload_num": 11,"upload_max": 10,"upload_type": 0,"options": [{"id": 11,"questionnaire_id": 1,"questionnaire_question_id": 8,"write_able": 1,"correct": 1,"sort": 1,"title": "答案A"},{"id": 12,"questionnaire_id": 1,"questionnaire_question_id": 8,"write_able": 0,"correct": 0,"sort": 2,"title": "答案B"}]}]}}
     * @return_param id int 问卷id
     * @return_param type int 状态 1:考试 0:问卷
     * @return_param title string 标题
     * @return_param sub_title string 副标题
     * @return_param submit_num int 用户提交次数
     * @return_param submit_message string 提交成功提示
     * @return_param description string 描述
     * @return_param start_time string 开始时间
     * @return_param end_time string 结束时间
     * @return_param question.id int 试题id
     * @return_param question.question_type int 1:单选,2:多选,3:简答题,4:上传文件,5:填空题
     * @return_param question.is_required int 是否必填：1 是，0 否
```

```
     * @return_param question.is_random int 是否随机显示选项(单选，多选)：1 是，0 否
     * @return_param question.fraction int 分数 0-100
     * @return_param question.title string 标题
     * @return_param question.tips string 提示
     * @return_param question.upload_num int 上传文件数量
     * @return_param question.upload_max int 上传文件大小
     * @return_param question.upload_type int 上传文件类型：1 图片，2 视频，3 压缩文件
     * @return_param question.options.id int 选项id
     * @return_param question.options.write_able int 是否可以填空 1:可以 0:不可以
     * @return_param question.options.correct int 是否正确 1:正确 0:错误
     * @return_param question.options.sort int 排序
     * @return_param question.options.title string 标题
     */
    public function questionnaireInfo(Request $request)
    {
        try {
            $code = (string)$request->get('code','');
            if(empty($code)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $questionnaireInfo = Questionnaire::query()->with(['question','question.options'])
                ->where(['code' => $code])->first();
            if(empty($questionnaireInfo)) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $questionnaireInfo = $questionnaireInfo->toArray();
            $submitNum = QuestionnaireUserAnswer::query()
                ->where(['questionnaire_id' => $questionnaireInfo['id'],'user_id' => $request->get('user_id')])->count();
            if($submitNum >= $questionnaireInfo['submit_num']) return
ApiReturn::error(ApiCode::SUBMIT_NUM['msg'],ApiCode::SUBMIT_NUM['code']);
            $userRoleIds = QuestionnaireUserRole::query()->where(['questionnaire_id' =>
$questionnaireInfo['id']])->pluck('user_role_id')->toArray();
            if(!empty($userRoleIds)) {
                $userId = $request->get('user_id');
                $count = UserUserRole::query()->where(['user_id' => $userId])->whereIn('user_role_id',$userRoleIds)->count();
                if($count == 0) return
ApiReturn::error(ApiCode::TASK_PERMISSION_DENIED_ERROR['msg'],ApiCode::TASK_PERMISSION_DENIED_ERROR['code']);
            }
            foreach ($questionnaireInfo['question'] as &$question) {
                if(in_array($question['question_type'],[1,2]) && $question['is_random'] == 1) {
                    shuffle($question['options']);
                }
            }
            $time = time();
            if($time < strtotime($questionnaireInfo['start_time'])) {
                return ApiReturn::success(['status' => 1,'questionnaire' => new \stdClass()]);
            }elseif ($time > strtotime($questionnaireInfo['end_time'])) {
                return ApiReturn::success(['status' => 2,'questionnaire' => new \stdClass()]);
            }else{
                return ApiReturn::success(['status' => 3,'questionnaire' => $questionnaireInfo]);
            }
        }catch (\Throwable $throwable) {
            Common::log('调查问卷错误',$throwable);
            return ApiReturn::error();
        }
    }

```

```
    /**
     * 用户提交答案
     * @catalog
     * @title 用户提交答案
     * @description 描述信息
     * @method post
     * @url /api/questionnaire/submitAnswer
     * @param questionnaire_id 选填 int 问卷id
     * @param complete_degree 选填 string 完成度 0-100
     * @param question.questionnaire_question_id 选填 int 题目id
     * @param question.answer.questionnaire_question_option_id 选填 int 选项id
     * @param question.answer.answer 选填 string 答案
     * @param question.answer.default_answer 选填 string 选择填空答案
     * @return {"code": 200,"msg": "OK","data": {"fraction_total": 11,"certification": true}}
     * @return_param fraction_total int 总分, certification bool 是否通过认证
     */
    public function submitAnswer(AnswerRequest $request)
    {
        $send = false;
        $request->validate();
        $answers = $request->validated();
        try {
            $userRoleIds = QuestionnaireUserRole::query()->where(['questionnaire_id' => $answers['questionnaire_id']])->pluck('user_role_id')->toArray();
            $answers['user_id'] = $request->post('user_id');
            if(!empty($userRoleIds)) {
                $count = UserUserRole::query()->where(['user_id' => $answers['user_id']])->whereIn('user_role_id',$userRoleIds)->count();
                if($count == 0) return ApiReturn::error(ApiCode::TASK_PERMISSION_DENIED_ERROR['msg'],ApiCode::TASK_PERMISSION_DENIED_ERROR['code']);
            }
            $questionnaireInfo = Questionnaire::query()->with(['question','question.options'])
                ->where(['id' => $answers['questionnaire_id']])->first();
            if(empty($questionnaireInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            $submitNum = QuestionnaireUserAnswer::query()
                ->where(['questionnaire_id' => $answers['questionnaire_id'],'user_id' => $answers['user_id']])->count();
            if($submitNum >= $questionnaireInfo->submit_num) return ApiReturn::error(ApiCode::SUBMIT_NUM['msg'],ApiCode::SUBMIT_NUM['code']);
            $options = $questionTypes = $fractions = $mutipleOptions = [];
            foreach ($questionnaireInfo->question as $question) {
                if(in_array($question->question_type,[1,2])) {
                    foreach ($question->options as $option) {
                        if ($option->correct == 1) {
                            if ($question->question_type == 1) {
                                $options[$option->id] = $option->title;
                            } else {
                                if (isset($mutipleOptions[$question->id])) $mutipleOptions[$question->id][$option->id] = $option->title;
                                else $mutipleOptions[$question->id] = [$option->id => $option->title];
                            }
                        }
                    }
                }
                $questionTypes[$question->id] = $question->question_type;
                $fractions[$question->id] = $question->fraction;
```

```
            }
            DB::beginTransaction();
            $time = date('Y-m-d H:i:s');
            $answers['fraction_total'] = 0;
            $questionTotal = count($questionnaireInfo->question);
            $answerTotal = count($answers['question']);
            $answers['complete_degree'] = floor($answerTotal/$questionTotal*100);
            foreach ($answers['question'] as &$answer) {
                $answer['questionnaire_id'] = $answers['questionnaire_id'];
                $answer['user_id'] = $answers['user_id'];
                $answer['questionnaire_question_type'] = $questionTypes[$answer['questionnaire_question_id']];
                if(in_array($answer['questionnaire_question_type'],[1,2])) {
                    if ($answer['questionnaire_question_type'] == 1) {
                        foreach ($answer['answer'] as $option) {
                            if(isset($options[$option['questionnaire_question_option_id']])) {
                                $answer['correct'] = 1;
                                $answer['fraction'] = $fractions[$answer['questionnaire_question_id']];
                                $answers['fraction_total'] += $answer['fraction'];
                            }else{
                                $answer['correct'] = 0;
                                $answer['fraction'] = 0;
                                break;
                            }
                        }
                    } else {
                        // 获取数组key
                        if(isset($mutipleOptions[$answer['questionnaire_question_id']])) {
                            // 多选正答列表
                            $keys = array_keys($mutipleOptions[$answer['questionnaire_question_id']]);
                            sort($keys);
                            $optionIds = array_column($answer['answer'], 'questionnaire_question_option_id');
                            sort($optionIds);
                            if (json_encode($keys) == json_encode($optionIds)) {
                                $answer['correct'] = 1;
                                $answer['fraction'] = $fractions[$answer['questionnaire_question_id']];
                                $answers['fraction_total'] += $answer['fraction'];
                            } else {
                                $answer['correct'] = 0;
                                $answer['fraction'] = 0;
                            }
                        } else {
                            $answer['correct'] = 0;
                            $answer['fraction'] = 0;
                        }
                    }
                }
        //          $answer['answer'] = json_encode($answer['answer']);
        //          $answer['created_at'] = $time;
        //          $answer['updated_at'] = $time;
            }
```

```
    $ip = CommonFnc::getRealIP();
    $answers['user_ip'] = $ip;
    $answers['user_address'] = BaiDuSdk::ipToAddress($ip);
    $answerInfo = QuestionnaireUserAnswer::query()->create($answers);
    foreach ($answers['question'] as &$option) {
        $option['questionnaire_question_answer_id'] = $answerInfo->id;
        $questionInfo = QuestionnaireUserAnswerQuestion::query()->create($option);
        foreach ($option['answer'] as &$answer) {
            $answer['questionnaire_question_answer_id'] = $answerInfo->id;
            $answer['questionnaire_id'] = $answers['questionnaire_id'];
            $answer['questionnaire_question_id'] = $option['questionnaire_question_id'];
            $answer['user_id'] = $answers['user_id'];
            $answer['questionnaire_user_answer_question_id'] = $questionInfo->id;
            $answer['created_at'] = $time;
            $answer['updated_at'] = $time;
        }
        QuestionnaireUserAnswerOption::query()->insert($option['answer']);
    }
    $rdata = ['fraction_total' => $answers['fraction_total']];
    // 自动认证逻辑
```

```
            $rdata['certification'] = false;
            // 考试类型 && 设置了自动认证 && 通过率达到要求
            if ($questionnaireInfo->type == 1) {
                $rdata['questions'] = $answers['question'];
                if ($questionnaireInfo->certification_role && $answers['fraction_total'] >= $questionnaireInfo->certification_score) {
                    // $can = true;
                    // // 鉴游团黑名单
                    // if ($questionnaireInfo->certification_role_id == Code::ROLE_TYPE_MAP['jianyoutuan']) {
                    //     $blackList = Redis::lrange(RedisKey::JYT_BLACK_LIST,0,-1);
                    //     if ($blackList && in_array($answers['user_id'],$blackList)) $can = false;
                    // }
                    // if ($can) {
                    //     $rdata['certification'] = true;
                    //     UserUserRole::firstOrCreate(['user_id' => $answers['user_id'], 'user_role_id' => $questionnaireInfo->certification_role_id]);
                    //     if ($questionnaireInfo->certification_role_id == Code::ROLE_TYPE_MAP['jianyoutuan']) {
                    //         User::mcnAward($answers['user_id'], 'jyt_award', $answers['user_id'], false);
                    //         UserNotifications::adminPushMessage($answers['user_id'], '鉴游团通知', '您已通过鉴游团考核并解锁了更高级务开去参与吧', 'activityCenter');
                    //     }
                    // }
                    $can = true;
                    // 鉴游团黑名单
                    if ($questionnaireInfo->certification_role_id == Code::ROLE_TYPE_MAP['jianyoutuan']) {
                        $blackList = Redis::lrange(RedisKey::JYT_BLACK_LIST,0,-1);
                        if ($blackList && in_array($answers['user_id'],$blackList)) $can = false;
                    }
                    if ($can) {
                        $rdata['certification'] = true;
                        UserUserRole::firstOrCreate(['user_id' => $answers['user_id'], 'user_role_id' => $questionnaireInfo->certification_role_id]);
                        $send = true;
                    }
```

```
                }
            }
            DB::commit();
            if($send){
                UserNotifications::adminPushMessage($answers['user_id'], '鉴游团通知', '您已通过鉴游团考核并解锁了更高级任务开去参吧', 'activityCenter');
            }
            return ApiReturn::success($rdata);
        }catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('提交答案错误',$throwable);
            return ApiReturn::error();
        }
    }

}

```

```
<?php

```

```php
namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Service\v1\ApiReturn;
use App\Service\v1\Quiz;
use Illuminate\Http\Request;

class QuizController extends Controller
{
    //列表
    public function list(Request $request)
    {
        $user_id = $request->get('user_id');
        return Quiz::list($user_id);
    }

    //用户提交答案
    public function answer(Request $request)
    {
        //用户ID
        $user_id = $request->post('user_id');
        //问题ID
        $question_id = $request->post('question_id');
        if (empty($question_id)) {
            return ApiReturn::error('请传入问题ID');
        }
        //用户提交的答案
        $answer = $request->post('answer');
        if (empty($answer)) {
            return ApiReturn::error('请填写答案');
        }
        return Quiz::answer($user_id, $question_id, $answer);
    }

    //问题详情
    public function questionInfo(Request $request)
    {
        //用户ID
        $user_id = $request->post('user_id');
        //问题ID
        $question_id = $request->post('question_id');
        if (empty($question_id)) {
            return ApiReturn::error('请传入问题ID');
        }
        return Quiz::questionInfo($user_id, $question_id);
    }

    //开启宝藏
```

```php
    public function openTreasure(Request $request)
    {
        //用户ID
        $user_id = $request->post('user_id');
        $treasure_id = $request->post('treasure_id');
        if (empty($treasure_id)) {
            return ApiReturn::error('请选择要开启的宝藏');
        }
        $award_key = $request->post('award_key');
        if (empty($award_key)) {
            return ApiReturn::error('请选择要开启的宝藏');
        }
        return Quiz::openTreasure($user_id, $treasure_id, $award_key);
    }
    //用户详情
    public function userInfo(Request $request)
    {
        //用户ID
        $user_id = $request->post('user_id');
        return Quiz::userInfo($user_id);
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\v1\Article;
use App\Models\v1\User;
use App\Service\v1\ApiReturn;
use Illuminate\Http\Request;

class RobotController extends Controller
{
    public function syncEsArticle(Request $request): object
    {
        try {
            $userId = $request->post('robot_id');
            if(User::query()->where('id',$userId)->where('is_robot','!=',0)->first()){
                Article::where('user_id',$userId)->searchable();
                return ApiReturn::error('同步成功');
            }
            return ApiReturn::error('查不到该用户');
        }catch (\Exception $e){
            return ApiReturn::error('同步失败'.$e->getMessage());
        }
    }
    public function getYesterdayActiveUser(): object
    {
        $startOfDay = date('Y-m-d 00:00:00', strtotime('yesterday'));
        $endOfDay = date('Y-m-d 23:59:59', strtotime('yesterday'));

        $list = User::query()
            ->whereBetween('visit_time', [$startOfDay, $endOfDay])
            ->pluck('id')->toArray();
        return ApiReturn::success($list);
    }

}

```

```php
<?php

namespace App\Http\Controllers\api\v1;

use App\Events\SearchLogEvent;
use App\Http\Controllers\Controller;
use App\Jobs\ErrorLog;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\Search;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class SearchController extends Controller
{
    /**
     * 综合搜索页面
     * @param Request $request
     * @return Object
     */
    public function searchLogAndRecommend(Request $request)
    {
        $userId = $request->get('user_id');
        $deviceNum = (string)$request->get('device_num','');
        try {
            $data = Search::searchLogAndRecommend($userId,$deviceNum);
            return ApiReturn::success($data);
        }catch (\Throwable $throwable) {
            Common::log('综合搜索页面错误',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 搜索联想
     * @param Request $request
     * @return Object
     */
    public function searchAssociate(Request $request)
    {
        $keyword = (string)$request->post('keyword','');
        if(empty($keyword)) return ApiReturn::success();
        try {
            $data = Search::searchAssociate($keyword);
            return ApiReturn::success($data);
        }catch (\Throwable $throwable) {
            Common::log('搜索联想错误',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 综合搜索
     * @param Request $request
     * @return Object
     */
    public function synthesizeSearch(Request $request)
    {
        $name = $request->get('name','');
        if(empty($name)) return ApiReturn::success();
        $limit  = $request->get('limit',10);
        $page = $request->get('page',1);
        $type = $request->get('type',0);
        $userId = $request->get('user_id',0);
        $deviceNum = $request->get('device_num');
        try{
            switch ($type) {
                case 1:
                    $field  = ["id","game_name","game_label","game_icon","score","evaluate_num","evaluating_num","operation_score","lang"];
                    $data = Search::gameSearch($name,$page,$limit,$userId,$field);
                    break;
                case 2:
                    $field  = ["id","cover",'rss_guid','rss_task_id','link_type','link_url',"title","like_num","game_id","game_name","game_icon","no_like_num","label","user_id","username",
"attestation_type","avatar","video","share_num","create_time","update_time","content","is_hot","task_id","elite","images","circle_id","top","comment_num","type","check_status","videos","status","user","circle","rss_account"];
                    $data = Search::articleSearch($name,$page,$limit,$userId,$field,$deviceNum);
                    break;
                case 3:
                    $field  = ["id","username","avatar","description","fans_num","attestation_type"];
                    $data = Search::userSearch($name,$page,$limit,$userId,$field);
                    break;
                case 4:
                    $field  = ["id","circle_name","avatar","game_id","attention_num","articles_num","discussion_num"];
                    $data = Search::circleSearch($name,$page,$limit,$userId,$field);
                    break;
                default:
                    $data = Search::synthesizeSearch($name,$page,$limit,$userId,$deviceNum);
                    break;
            }
            SearchLogEvent::dispatch($name,$userId,$deviceNum ?? '');
            return ApiReturn::success($data);
        }catch (\Exception $e) {
            Common::log('综合搜索错误',$e);
            return ApiReturn::error();
        }
    }

}

```

```php
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\DomesticGameApproval;
use App\Models\Leaderboard;
use App\Models\v1\Circle;
use App\Models\v1\Game;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\CircleService;
use App\Service\v1\CommonFnc;
use App\Service\v1\SpecialSubject;
use App\Service\v1\User;
use Facade\FlareClient\Api;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class SpecialSubjectController extends Controller
{
    public function detail(Request $request)
    {
        $id = $request->get('id', 1);
        if (empty($id)) {
            return ApiReturn::error('请传入 Id');
        }
        return SpecialSubject::detail($id);
    }

    //视频
    public function videos(Request $request): object
    {
        $limit = $request->get('limit', 10);
        $special_subject_id = $request->get('special_subject_id', 0);
        if (empty($special_subject_id)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        return SpecialSubject::videos($special_subject_id, $limit);
    }

    //专题列表
    public function list(Request $request)
    {
        $page = $request->get('page', 1);
        $limit = $request->get('limit', 5);
        return SpecialSubject::list($limit);
    }

    //榜单配置
```

```php
    public function leardboard(Request $request)
    {
        $limit = $request->get('limit', 3);
        $lang = CommonFnc::lang();
        $user_id = $request->get('user_id');
        $list = Leaderboard::query()
            ->where('lang', $lang)
            ->where('type', 1)
            ->select(['id', 'name', 'cover']);
        if ($limit == 8) {
            $list = $list->limit(3);
        }
        if ($limit >= 8) {
            $list = $list->withCount('games')->orderBy('sort', 'desc')->get();
        } else {
            $list = $list->get();
        }
        $list->load(['games.game_labels' => function ($query) use ($lang) {
            $query->where('lang', $lang);
        }]);
        $list->each(function ($leaderboard) use ($limit) {
            $leaderboard->setRelation('games', $leaderboard->games->take($limit));
        });
        $list = isset($list) ? $list->toArray() : [];
        $data = [];
        foreach ($list as $k => $v) {
            $data[$k]['id'] = $v['id'];
            $data[$k]['name'] = $v['name'];
            $data[$k]['ranking_count'] = $v['games_count'] ?? 0;
            $data[$k]['cover'] = CommonFnc::concatUrlStr($v['cover']);
            $games = [];
            foreach ($v['games'] as $k1 => $v1) {
                $games[$k1]['position'] = $k1 + 1;
                if ($k1 == 0) {
                    $games[$k1]['check'] = true;
                } else {
                    $games[$k1]['check'] = false;
                }
                $games[$k1]['id'] = $v1['id'];
                $games[$k1]['game_name'] = $v1['game_name'];
                $games[$k1]['score'] = $v1['score'];
                $games[$k1]['game_icon'] = CommonFnc::concatUrlStrAdmin($v1['game_icon']);
                $game_label = '';
                foreach ($v1['game_labels'] as $k2 => $v2) {
                    if ($k2 <= 2) {
                        $game_label .= $v2['label_name'] . '·';
                    }
                }
                $circle = Circle::query()->where('game_id', $v1['id'])->select(['id', 'circle_name'])->first('id');
                if ($circle) {
                    $circle['is_attention'] = CircleService::isAttention($user_id, $circle['id']);
                }
                $games[$k1]['circle'] = $circle;
                $games[$k1]['circle_id'] = $circle['id'] ?? 0;
                $games[$k1]['game_label'] = rtrim($game_label, '·');
            }
            $data[$k]['ranking'] = $games;
        }
        return ApiReturn::success($data);
    }

    public function ranking(Request $request)
    {
        $id = $request->get('id', 1);
        $page = $request->get('page', 1);
        $limit = $request->get('limit', 10);
        $user_id = $request->get('user_id');
        if ($page * $limit > 150) {
            return ApiReturn::success();
        }
        $leardboard = Leaderboard::query()->select(['id', 'lang', 'name', 'cover'])->find($id);
        if (!$leardboard) {
            return ApiReturn::success();
        }
        $lang = $leardboard->lang;
        $leardboard->cover = CommonFnc::concatUrlStr($leardboard->cover);
        $list = $leardboard->games()->with(['circle' => function ($query) {
            $query->select(['id', 'circle_name', 'game_id', 'avatar', 'cover']);
        }, 'game_labels' => function ($query) use ($lang) {
            $query->where('lang', $lang);
        }])->orderBy('sort', 'desc')->paginate($limit);
        $list = isset($list) ? $list->toArray() : [];
        $list['leardboard'] = $leardboard;
        CommonFnc::paginateReturnUnset($list);
        foreach ($list['data'] as $k => $v) {
            $game = [];
            $game['position'] = ($page - 1) * $limit + $k + 1;
            $game['id'] = $v['id'];
            $game['score'] = $v['score'];
            $game['game_name'] = $v['game_name'];
            if (!empty($v['circle'])) {
                $v['circle']['is_attention'] = CircleService::isAttention($user_id, $v['circle']['id']);
            }
            $game['circle'] = $v['circle'] ?? [];
            $game['circle_id'] = $v['circle']['id'] ?? 0;
            $game['game_icon'] = CommonFnc::concatUrlStrAdmin($v['game_icon']);
            $game_label = '';
            foreach ($v['game_labels'] as $k1 => $v1) {
                if ($k1 <= 2) {
                    $game_label .= $v1['label_name'] . '·';
```

```
                }
            }
            $game['rec_text'] = '';
            $game['game_label'] = rtrim($game_label, '·');
            $list['data'][$k] = $game;
        }
        return ApiReturn::success($list);
    }

    public function domesticGamesApprovals(): object
    {
        $list = DomesticGameApproval::query()
            ->where('status', 1)
            ->select(['id', 'title', 'cover', 'publication_date', 'link_url'])
            ->orderBy('id', 'desc')
            ->get()
            ->toArray();
        foreach ($list as $k => $v) {
            $list[$k]['cover'] = CommonFnc::concatUrlStr($v['cover']);
        }
        return ApiReturn::success($list);
    }

}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Models\v1\SystemTask;
use App\Models\v1\UserDiamondLog;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\RedisKey;
use App\Service\v1\SystemTaskService;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
    use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Redis;
use Predis\Client;

class SystemTaskController extends Controller
{

    /**
     * 任务中心说明
     * @return Object
     */
    public function systemTaskExplain()
    {
        try {
            $redisConf = config('database.redis.default');
            $client = new Client($redisConf);
            $info = $client->get(env('SOLO_ADMIN_REDIS_PREFIX').RedisKey::SYSTEM_TASK_EXPLAIN);
            return ApiReturn::success(['explain' => $info]);
        } catch (\Throwable $throwable) {
            Common::log('任务中心说明',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 任务中心钻石统计
     * @param Request $request
     * @return Object
     */
    public function diamondStatistic(Request $request)
    {
        $userId = $request->get('user_id');
        try {
            $statisticInfo = SystemTaskService::diamondStatistic($userId);
            return ApiReturn::success($statisticInfo);
        } catch (\Throwable $throwable) {
```

```
            Common::log('任务中心钻石统计',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 收益/支出记录
     * @param Request $request
     * @return Object
     */
    public function diamondChangeLogs(Request $request)
    {
        $action = intval($request->get('action',1));
        if(!in_array($action,[1,2])) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        try {
            $userId = $request->get('user_id');
            $page = $request->get('page',1);
            $limit = $request->get('limit',10);
            $page = ($page - 1) * $limit;
            $where = [
                'user_id' => $userId,
                'action' => $action
            ];
            $field = ['source_id','type','describe','diamond','created_at'];
//            $diamondLogs = (new UserDiamondLog())->diamondLogs($where,$field,$page,$limit);
            $diamondLogs = SystemTaskService::diamondChangeLogs($where,$field,$page,$limit);
            return ApiReturn::success($diamondLogs);
        } catch (\Throwable $throwable) {
            Common::log('收益/支出记录',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 签到奖励列表
     * @param Request $request
     * @return Object
     */
    public function signTasks(Request $request)
    {
        $userId = $request->get('user_id');
        try {
            $list = SystemTaskService::signTasks($userId);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('签到任务',$throwable);
            return ApiReturn::error();
        }
    }

```

```
    /**
     * 签到
     * @param Request $request
     * @return Object
     */
    public function signed(Request $request)
    {
        $awardId = intval($request->post('award_id',0));
        $userId = $request->post('user_id');
        Log::info('签到-入参',['userId' => $userId,'awardId' => $awardId]);

        try {
            DB::beginTransaction();
            $re = $awardId > 0 ? SystemTaskService::signed($userId,$awardId) : SystemTaskService::autoSigned($userId);
            if($re['status']) {
                DB::commit();
                Log::info('签到-status为true时返回值', $re);
                return $awardId > 0 ? ApiReturn::success() : ApiReturn::success(['is_sign' => $re['is_sign'],'award' => $re['award']]);
            }
            DB::rollBack();
            Log::info('签到-status为false时返回值',$re);
            return ApiReturn::error($re['msg'],$re['code']);
        } catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('签到动作',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 任务中心任务列表
     * @param Request $request
     * @return Object
     */
    public function systemTasks(Request $request)
    {
        $taskType = intval($request->get('task_type',0));
        $userId = $request->get('user_id');
        $page = $request->get('page',1);
        $limit = $request->get('limit',10);
        $page = ($page - 1) * $limit;
        try {
            $list = SystemTaskService::systemTasks($taskType,$userId,$page,$limit);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('系统任务列表',$throwable);
            return ApiReturn::error();
        }
    }

```

```
    /**
     * 任务中心领取奖励
     * @param Request $request
     * @return Object
     */
    public function getAward(Request $request)
    {
        $taskId = intval($request->get('task_id',0));
        if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $taskInfo = SystemTask::where(['id' => $taskId])->select('id','task_category','award')->first();
        if(empty($taskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->get('user_id');
        try {
            $response = SystemTaskService::getAward($userId,$taskInfo);
            if($response) return ApiReturn::success();
            return ApiReturn::error(ApiCode::SYSTEM_TASK_REPEAT_GET['msg'],ApiCode::SYSTEM_TASK_REPEAT_GET['code']);
        } catch (\Throwable $throwable) {
            Common::log('任务中心-领取奖励',$throwable);
            return ApiReturn::error();
        }
    }

}

```

```
<?php

namespace App\Http\Controllers\api\v1;

use App\Http\Controllers\Controller;
use App\Http\Requests\NewActivityCenterRequest;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Article;
use App\Service\v1\Code;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\Task2;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Throwable;

/**
 * 任务
 */
class TaskController extends Controller
{
    /**
     * 任务列表
     * @param Request $request
     * @return object
     */
    public function list(Request $request): object
    {
        Log::info('获取任务列表入参', $request->all());
        //获取用户Id
        $userId = $request->get('user_id');
        $limit = (int)$request->get('limit', 10);
        $page = (int)$request->get('page', 1);
        $taskStatus = (int)$request->get('task_status', 0);
        $time = $request->server('REQUEST_TIME');
        return Task2::list($userId, $time, $page, $limit, $taskStatus);
    }

    /**
     * 任务详情
     * @param Request $request
     * @return Object
     */
    public function info(Request $request): object
    {
        $taskId = intval($request->get('task_id', 0));
        $userId = $request->get('user_id', 0);
        if(empty($taskId)){
            return ApiReturn::rdata(ApiCode::TASK_ERROR);
```

```
        }
        $platform = $request->header('Platform');
        Log::info('获取任务信息详情入参', $request->all());
        $version = $request->header('Version');
        return Task2::info($taskId, $userId,$platform,$version);
    }

    //参与任务
    public function participating(NewActivityCenterRequest $request): object
    {
        Log::info('用户参与任务入参', $request->all());
        $param = $request->post();
        $param['action'] = $request->post('action', Code::ACTION_ADD);
        $request->validate('participatingTask');
        $ip = CommonFnc::getRealIP();
        try {
            $param['action'] = $request->post('action', Code::ACTION_ADD);
            $param['status'] = $request->post('status', 0);
            Log::info('参与任务==入口参数=====' ,$param);
            // 验证 设置
            $err = Task2::checkParticipatingTask($param);
            if ($err) {
                Log::info('参与任务==校验失败=====',$err);
                return ApiReturn::rdata($err);
            }
            // 插入到 article 表的数据
            $param['data'] = [
                'cover' => $param['cover'],
                'game_score' => empty($param['game_score']) ? 0 : $param['game_score'],
                'check_status' => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
                'content' => $param['content'],
                // article 标题 截取
                'title' => '',
                'text_content' => $param['text_content'],
                'content_len' => $param['content_len'],
                'home_page' => $param['home_page'],
                'label_list' => $param['label_list'],
                'abstract' => $param['abstract'],
                'user_id' => $param['user_id'],
                'creator_source_id' => $param['creator_source_id'],
                'task_article_status' => $param['task_article_status'],
                'state' => 1,
                'task_tab_id' => $param['task_tab_id']
            ];
            if ($param['action'] == Code::ACTION_ADD) {
                // 新增需要的数据 (修改的时候不可更改的数据)
                $param['data'] = array_merge($param['data'], [
                    'parent_id' => empty($param['parent_id']) ? 0 : $param['parent_id'],
                    'game_id' => $param['game_id'],
                    'task_id' => $param['task_id'],
```

```
                    'circle_id' => $param['circle_id'],
                    'type' => $param['type'],
                    'task_tab_id' => $param['task_tab_id'],
                    'task_article_status' => $param['task_article_status'],
                    'status' => $param['status'],
                    'is_writer_task' => $param['is_writer_task'],
                    'creator_source_id' => $param['creator_source_id']
                ]);
            }
            Log::info('参与任务==进入数据库参数=====' , $param);
            return Article::addNewArticle($param,$ip);
        } catch (Throwable $throwable) {
            Common::log('参与任务' , $throwable);
            return ApiReturn::error($throwable);
        }
    }

    /**
     * 新增/编辑收集地址
     * @param NewActivityCenterRequest $request
     * @return Object
     */
    public function saveCollectPlatformUrl(NewActivityCenterRequest $request): object
    {
        $params = $request->post('activity_center_platform_url', []);
        if (is_string($params)) $params = json_decode($params, true); // ios 传惨问题
        $articleId = (int)$request->get('article_id', 0);
        if (empty($params)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
        $userId = $request->post('user_id');
        return Task2::saveCollectPlatformUrl($userId, $articleId, $params);
    }

    /**
     * 获取用户参与任务信息
     * @param Request $request
     * @return Object
     */
    public function userRecordInfo(Request $request): object
    {
        Log::info('获取用户参与任务信息详情入参', $request->all());
        $user_id = $request->get('user_id', 0);
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $task_id = $request->get('task_id', 0);
        if (empty($task_id)) {
            return ApiReturn::rdata(ApiCode::PARAM_ERROR);
        }
```

```
        $article_id = $request->get('article_id', 0);
        if(empty($article_id)){
            return ApiReturn::error('未查询到您的任务参与信息');
        }
        $platform = $request->header('Platform');
        $version = $request->header('Version');
        return Task2::userRecordInfo($user_id, $task_id,$article_id,$platform,$version);
    }

    /**
     * 用户参与记录
     * @param Request $request
     * @return Object
     */
    public function userRecords(Request $request): object
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $task_id = $request->get('task_id');
        $limit = (int)$request->get('limit', 10);
        Log::info('用户参与任务列表入参',['user_id'=>$user_id,'task_id'=>$task_id,'page' => $request->get('page', 1)]);
        $platform = $request->header('Platform');
        $version = $request->header('Version');
        return Task2::userRecords($user_id, $limit,$task_id,$platform,$version);
    }

    /**
     * 活动中心用户信息
     * @param Request $request
```

```php
     * @return Object
     */
    public function getUserInfo(Request $request): object
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return Task2::getUserInfo($user_id);
    }

    /**
     * 收藏/取消收藏任务
     * @param Request $request
     * @return object
     */
    public function toggleCollect(Request $request): object
    {
        $taskId = (int)$request->get('task_id', 0);
```

```php
        $userId = (int)$request->get('user_id', 0);
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        if(empty($taskId)){
            return ApiReturn::error('请选择要收藏的任务');
        }
        return Task2::toggleCollect($taskId, $userId);
    }
    //用户收藏列表
    public function collectList(Request $request){
        $userId = (int)$request->get('user_id', 0);
        if(empty($userId)){
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $limit = (int)$request->get('limit', 10);
        $time = $request->server('REQUEST_TIME');
        return Task2::collectList($userId,$limit,$time);
    }
}

```

```php
<?php

namespace App\Http\Controllers\api\v1;

use App\Biz\WechatUser;
use App\Events\LikeTaskEvent;
use App\Events\ParamErrorEvent;
use App\Events\PersonalTaskEvent;
use App\Http\Controllers\Controller;
use App\Http\Requests\Address;
use App\Http\Requests\v1\UserRequest;
use App\Jobs\ErrorLog;
use App\Models\Suggestion;
use App\Models\SuggestionImage;
use App\Models\UserBrowseLog;
use App\Models\UserCollect;
use App\Models\UserEvent;
use App\Models\UserModuleTab;
use App\Models\UserSocial;
use App\Models\ParamError;
use App\Models\v1\ApplicationAvatar;
use App\Models\v1\SearchKeywordLog;
use App\Models\v1\ServiceConfig;
use App\Models\v1\ShippingAddress;
use App\Models\v1\User as V1User;
use App\Models\v1\UserAlipush;
use App\Models\v1\UserGameSearchLog;
use App\Models\v1\UserLike;
use App\Models\v1\UserPushConfig;
use App\Models\v1\UserSearchLog;
use App\Models\v1\UserUserRole;
use App\Service\v1\AliContentAudit;
use App\Service\v1\AliPush;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Article;
use App\Service\v1\Code;
use App\Service\v1\Common;
use App\Service\v1\CommonFnc;
use App\Service\v1\RedisKey;
use App\Service\v1\SoloCode;
use App\Service\v1\task\ParamErrorTask;
use App\Service\v1\User;
use Dflydev\DotAccessData\Data;
use Hhxsv5\LaravelS\Swoole\Task\Task;
use Illuminate\Auth\Access\AuthorizationException;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Redis;
```

```
use Illuminate\Support\Facades\Route;
use Illuminate\Validation\ValidationException;
use Psr\Container\ContainerExceptionInterface;
use Psr\Container\NotFoundExceptionInterface;

class UserController extends Controller
{
    const EDIT_FIELD_TYPE = [
        1 => 'avatar',
        2 => 'username',
        3 => 'description',
        4 => 'sex',
        5 => 'address_country_id',
    ];

    /**
```

```
     * 我的页面个人信息
     * @param Request $request
     * @return Object
     */
    public function getPrivateUserInfo(Request $request)
    {
        $platForm = $request->input('platform', 1);
        $userId = $request->get('user_id');
        try {
            $userInfo = User::getPrivateUserInfo($userId);
            return ApiReturn::success($userInfo);
        } catch (\Throwable $throwable) {
            Common::log('我的个人主页', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 个人主页用户信息
     * @param Request $request
     * @return Object
     */
    public function getUserInfo(Request $request)
    {
        $userId = $request->get('user_id');
        $authorId = intval($request->get('author_id', 0));
        try {
            if ($authorId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            $userInfo = User::getUserInfo($userId, $authorId);
            if (empty($userInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            return ApiReturn::success($userInfo);
        } catch (\Throwable $throwable) {
            Common::log('个人主页用户信息', $throwable);
            return ApiReturn::error();
```

```
        }
    }

    //增加用户喜欢社区
    public function addUserLikeCircles(Request $request)
    {
        $userId = $request->post('user_id');
        $circleIds = $request->post('circle_ids');
        if (empty($circleIds) && !is_array($circleIds)) {
            return ApiReturn::success();
        }
        //去除空数据
        $circleIds = array_filter($circleIds);
        //去重
        $circleIds = array_unique($circleIds);
        if (empty($circleIds)) {
            return ApiReturn::success();
        }
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return User::addUserLikeCircles($userId, $circleIds);
    }

    /**
     * 认证列表
     * @param Request $request
     * @return Object
     */
    public function getUserCertification(Request $request)
    {
        $userId = $request->get('user_id');
        try {
            $userInfo = V1User::find($userId);
            $certification_jyt = UserUserRole::where(['user_id' => $userId, 'user_role_id' => 6])->exists();
            $jyturl = '';
            if (!$certification_jyt) {
                $jyturl = ServiceConfig::where('field_name', 'certification_janyoutuan')->value('value');
                if ($jyturl) $jyturl = 'solo://game:123/h5?url=' . $jyturl;
            }
            $data = [
                'attestation_expert' => [
                    'is_certification' => $userInfo->attestation_type == 2,
                    'scheme' => '',
                ],
                'certification_jyt' => [
                    'is_certification' => $certification_jyt,
                    'scheme' => $jyturl,
                ],
                'show_attestation_expert' => (bool)ServiceConfig::where('field_name', 'show_attestation_expert')->value('value'),
```

```
                'show_certification_jyt' => (bool)ServiceConfig::where('field_name', 'show_certification_jyt')->value('value')

            ];
            return ApiReturn::success($data);
        } catch (\Throwable $throwable) {
            Common::log('个人主页用户信息', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 综合搜索记录
     * @param Request $request
     * @return Object
     */
    public function userSearchLogs(Request $request)
    {
        $userId = $request->get('user_id', 0);
        $deviceNum = $request->get('device_num', '');
        $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
        try {
            $searchLogs = UserSearchLog::where($where)->orderBy('id', 'desc')->limit(15)->pluck('keyword');
            return ApiReturn::success($searchLogs);
        } catch (\Throwable $throwable) {
            Common::log('综合搜索记录', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 清除综合搜索历史记录
     * @param Request $request
     * @return Object
     */
    public function clearSearchLogs(Request $request)
    {
        $userId = $request->get('user_id', 0);
        $deviceNum = $request->get('device_num', '');
        $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
        try {
            SearchKeywordLog::where($where)->delete();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('清除综合搜索历史记录', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 游戏搜索历史记录
```

```
     * @param Request $request
     * @return Object
     */
    public function userGameSearchLogs(Request $request)
    {
        $userId = $request->get('user_id', 0);
        $deviceNum = $request->get('device_num', '');
        $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
        try {
            $searchLogs = UserGameSearchLog::where($where)->orderBy('id', 'desc')->limit(15)->pluck('keyword');
            return ApiReturn::success($searchLogs);
        } catch (\Throwable $throwable) {
            Common::log('我的游戏搜索记录', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 清除游戏搜索历史记录
     * @param Request $request
     * @return Object
     */
    public function clearGameSearchLogs(Request $request)
    {
        $userId = $request->get('user_id', 0);
        $deviceNum = $request->get('device_num', '');
        $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
        try {
            UserGameSearchLog::where($where)->delete();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('清除我的游戏搜索记录', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 编辑用户信息
     * @param Request $request
     * @return Object
     */
    public function editUserInfo(Request $request)
    {
        try {
            $type = $request->post('type', 0);
            $keyword = trim($request->post('keyword', null));
            if (!isset(static::EDIT_FIELD_TYPE[$type]) || is_null($keyword)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            $userId = $request->post('user_id');
            if ($type == 1) {
                if (ApplicationAvatar::where(['user_id' => $userId, 'check_status' => 1])->exists()) return ApiReturn::error(ApiCode::AVATAR_ERROR['msg'], ApiCode::AVATAR_ERROR['code']);
```

```
                $imageStatus = AliContentAudit::imageAudit($keyword);
                if ($imageStatus == SoloCode::ALI_AUDIT_BLOCK) return ApiReturn::error(ApiCode::ALI_IMAGE_AUDIT['msg'], ApiCode::ALI_IMAGE_AUDIT['code']);
                $avatarData = [
                    'user_id' => $userId,
                    'check_status' => $imageStatus == SoloCode::ALI_AUDIT_PASS ? 3 : 1,
                    'avatar' => $keyword,
                ];
                ApplicationAvatar::create($avatarData);
                if ($imageStatus == SoloCode::ALI_AUDIT_BLOCK) return ApiReturn::success();
            } else if ($type == 2) {
                $str = Redis::get(RedisKey::NICKNAME_FILTER);
                if (empty($keyword)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
                $arr = explode(' , ', $str);
                if (in_array($keyword, $arr)) return ApiReturn::error(ApiCode::NICKNAME_ERROR['msg'], ApiCode::NICKNAME_ERROR['code']);
            }
            if ($type == 2 || $type == 3) {
                $x = request()->header('X-Requested-From');
                if (!empty($x) && $x == 'MiniProgram') {
                    $userInfo = V1User::query()->where('id', $userId)->first();
                    if (empty($userInfo->xcx_openid)) {
                        return ApiReturn::error(ApiCode::LOGIN_ERROR['msg'], ApiCode::LOGIN_ERROR['code']);
                    }
                    //小程序
                    $res = (new WechatUser())->checkText($keyword, 1, $userInfo['xcx_openid']);
                    if (!empty($res)) {
                        return ApiReturn::error($res);
                    }
                }
            }
            V1User::where(['id' => $userId])->update([static::EDIT_FIELD_TYPE[$type] => $keyword]);
            if (V1User::find($userId)) {
                V1User::find($userId)->searchable();
            }
            PersonalTaskEvent::dispatch($userId);
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('编辑个人资料', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 修改用户个人中心背景图
     * @param Request $request
     * @return object
     */
    public function updateBackground(Request $request): object
    {
        $userId = $request->post('user_id');
        if (empty($userId)) {
```

```
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $backgroundUrl = $request->post('background_url');
    if (empty($backgroundUrl)) {
        return ApiReturn::rdata(ApiCode::PARAM_ERROR);
    }
    return User::updateBackground($userId, $backgroundUrl);
}

/**
 * 钻石数量
 * @param Request $request
 * @return Object
 */
public function diamonds(Request $request)
{
    $userId = $request->get('user_id');
    try {
        $diamondNum = V1User::where(['id' => $userId])->value('diamond_num');
        return ApiReturn::success(['diamond_num' => (int)$diamondNum]);
    } catch (\Throwable $throwable) {
        Common::log('我的个人主页', $throwable);
        return ApiReturn::error();
    }
}


```

```
    /**
     * 新增收货地址
     * @param Address $request
     * @return Object
     * @throws \Illuminate\Auth\Access\AuthorizationException
     * @throws \Illuminate\Validation\ValidationException
     */
    public function addShippingAddress(Address $request)
    {
        $request->validate();
        $param = $request->post();
        Log::info('新增收货地址入参==', $param);
        try {
            if ($param['default_address']) {
                ShippingAddress::where(['user_id' => $param['user_id'], 'default_address' => 1])->update(['default_address' => 0]);
            }
            $response = ShippingAddress::create($param);
            Log::info('新增收货地址成功');
            return ApiReturn::success($response);
        } catch (\Throwable $throwable) {
            Common::log('新增收货地址', $throwable);
            return ApiReturn::error();
        }
```

```
    }

    /**
     * 编辑我的收货地址
     * @param Address $request
     * @return Object
     * @throws \Illuminate\Auth\Access\AuthorizationException
     * @throws \Illuminate\Validation\ValidationException
     */
    public function editShippingAddress(Address $request)
    {
        $request->validate();
        $param = $request->post();
        Log::info('修改收货地址入参==', $param);
        try {
            $addressId = intval($request->post('id', 0));
            if ($addressId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            $addressInfo = ShippingAddress::where(['id' => $addressId, 'user_id' => $param['user_id']])->first();
            if (empty($addressInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            if ($param['default_address']) {
                ShippingAddress::where(['user_id' => $param['user_id'], 'default_address' => 1])->update(['default_address' => 0]);
            }
            $addressInfo->name = $param['name'];
            $addressInfo->mobile = $param['mobile'];
            $addressInfo->province_id = $param['province_id'];
            $addressInfo->city_id = $param['city_id'];
            $addressInfo->district_id = $param['district_id'];
            $addressInfo->detailed_address = $param['detailed_address'];
            $addressInfo->default_address = $param['default_address'];
            $addressInfo->save();
            Log::info('修改收货地址成功');
            return ApiReturn::success($addressInfo);
        } catch (\Throwable $throwable) {
            Common::log('修改收货地址报错', $throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 我到收货地址列表
     * @param Request $request
     * @return Object
     */
    public function shippingAddresses(Request $request)
    {
        try {
            $userId = $request->get('user_id');
            $address = ShippingAddress::query()
                ->where('user_id', $userId)
```

```
                ->select('id', 'province_id', 'city_id', 'district_id', 'detailed_address', 'name', 'mobile', 'default_address')
                ->get()->toArray();
            return ApiReturn::success($address);
        } catch (\Throwable $throwable) {
            Common::log('我的收货地址列表', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 设置默认地址
     * @param Request $request
     * @return Object
     */
    public function setDefaultAddress(Request $request)
    {
        try {
            DB::beginTransaction();
            $addressId = intval($request->post('address_id', 0));
            if ($addressId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            $userId = $request->post('user_id');
            $addressInfo = ShippingAddress::where(['user_id' => $userId, 'id' => $addressId])->first();
            if (empty($addressInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            if (empty($addressInfo->default_address)) {
                ShippingAddress::where(['user_id' => $userId, 'default_address' => 1])->update(['default_address' => 0]);
                $addressInfo->default_address = 1;
                $addressInfo->save();
            }
            DB::commit();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            DB::rollBack();
            Common::log('设置默认地址', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 点赞/取消点赞
     * @param Request $request
     * @return Object
     */
    public function userLike(Request $request)
    {
        try {
            $relevancId = (int)$request->post('relevance_id', 0);
            $type = (int)$request->post('type', 1);
            $authorId = intval($request->post('author_id', 0));
            $platform = $request->header('platform');
            $versionCode = $request->header('version-code');
```

```
            if ($platform == 2 && $versionCode == '3.0.0') {
                $rid = $relevancId;
                $relevancId = $authorId;
                $authorId = $rid;
            }
            Log::info("用户点赞和取消点赞入参", ['relevancId' => $relevancId, 'type' => $type, 'authorId' => $authorId]);
            if ($relevancId <= 0 || $authorId <= 0 || !in_array($type, SoloCode::LIKE_TYPE)) {
                ParamErrorEvent::dispatch($request, '点赞/取消点赞');
                return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
            }
            $userId = $request->post('user_id');
//            if (Redis::exists(RedisKey::CAN_LIKE_TIME . $userId)) return ApiReturn::error('操作太频繁，请稍后再试...');
//            Redis::set(RedisKey::CAN_LIKE_TIME . $userId, 1, 'PX', 300, 'NX');
//            $isCanLike = User::canLike($userId);
//            if(empty($isCanLike)) return ApiReturn::error('操作太频繁，请稍后再试...');
            $response = User::userLike($userId, $relevancId, $type, $authorId);
            if ($response == 1) LikeTaskEvent::dispatch($relevancId, $userId, $type);
            return ApiReturn::success(['status' => $response], $response == 1 ? '点赞成功' : '取消点赞');
        } catch (\Throwable $throwable) {
            Common::log('点赞', $throwable);
            return ApiReturn::error();
        }
    }

    public function robotLogin(UserRequest $request)
    {
        $uid = $request->post('uid');
        $userInfo = V1User::query()->where('is_robot', '!=', 0)->where('id', $uid)->first();
        if (!$userInfo) {
            return ApiReturn::error('当前用户非机器人');
        }
        $api_token = $userInfo->api_token;
        if (!Redis::exists('token:' . $api_token)) {
            //如果请求接口返回请先登录则去生成一个token并存入redis中
            $api_token = CommonFnc::generateToken();
            //如果登录过期时间一周
            Redis::setex('token:' . $api_token, 5184000, $userInfo['id']);
            V1User::query()->where('id', $userInfo->id)->update(['api_token' => $api_token]);
        }
        return ApiReturn::success(['api_token' => $api_token]);
    }

    /**
     * 设置用户活页时间
     * @param Request $request
     * @return Object
     */
    public function userVisitTime(Request $request)
    {
        $userId = $request->get('user_id');
```

```
        if ($userId > 0) {
            V1User::query()->where(['id' => $userId])->update(['visit_time' => date('Y-m-d H:i:s')]);
            UserEvent::query()
                ->insert([
                    'user_id' => $userId,
                    'created_at' => date('Y-m-d H:i:s'),
                    'type' => UserEvent::TYPE['visit']
                ]);
        }
        return ApiReturn::success();
    }

    /**
     * 新增修改认证信息
     * @catalog
     * @title 新增修改认证信息
     * @description 描述信息
     * @method post
     * @url /api/user/authentication
     * @param idcard 必选 string 身份证号码
     * @param name 必选 string 姓名
     * @return {"code":200,"message":"ok"}
     */
    public function authentication(UserRequest $request)
    {
        $userId = $request->post('user_id');
        $request->validate('authentication');
        $data = $request->validated();
        try {
            if (V1User::where('idcard', $data['idcard'])->where('id', '<>', $userId)->first()) return
ApiReturn::rdata(ApiCode::HAS_BIND_ADCARD);
            $rs = V1User::where('id', $userId)->update($data);
            $age = User::getAge($data['idcard']);
            //获取用户是否为青少年
            $res['is_teenager'] = $age <= 18;
            if ($rs) return ApiReturn::success($res);
            else return ApiReturn::error();
        } catch (\Exception $th) {
            Common::log('实名认证', $th);
            return ApiReturn::error();
        }
    }

    /**
     * 获取用户实名信息
     * @catalog
     * @title 获取用户实名信息
     * @description 获取用户实名信息
     * @method get
     * @url /api/user/getAuthenticationInfo
     * @return {"code":200,"message":"ok","data":{"idcard":8,"name": "姓名"}
```

```
     * @return_param idcard   string   身份证号码
     * @return_param name string   姓名
     */
    public function getAuthenticationInfo(Request $request)
    {
        $userId = $request->post('user_id');
        try {
            $info = V1User::find($userId);
            if (!$info) return ApiReturn::error();
            return ApiReturn::success([
                'idcard' => $info->idcard,
                'name' => $info->name,
            ]);
        } catch (\Exception $th) {
            Common::log('实名认证详情', $th);
            return ApiReturn::error();
        }
    }

    /**
     * 用户推送设置
     * @catalog
     * @title 用户推送设置
     * @description 描述信息
     * @method get
     * @url /api/user/pushConfigs
     * @return {"code": 200,"msg": "OK","data": {"id": 3,"user_id": 615,"comment_status": true,"like_status": true,"attention_status": true,"remind_status": true,"system_status": true}}
     * @return_param id   int   配置id
     * @return_param user_id int   用户id
     * @return_param comment_status boolean   评论开关：true 开，false 关
     * @return_param like_status boolean   点赞/收藏开关：true 开，false 关
     * @return_param attention_status boolean   新增关注开关：true 开，false 关
     * @return_param remind_status boolean   提醒(被@)开关：true 开，false 关
     * @return_param system_status boolean   系统开关：true 开，false 关
     */
    public function pushConfigs(Request $request)
    {
        $userId = $request->post('user_id');
        try {
            $pushConfig = UserPushConfig::query()->firstOrCreate(['user_id' => $userId], [
                'comment_status' => 1,
                'like_status' => 1,
                'attention_status' => 1,
                'remind_status' => 1,
                'system_status' => 1,
            ]);
            return ApiReturn::success($pushConfig);
        } catch (\Throwable $throwable) {
            Common::log('获取用户推送配置信息', $throwable);
            return ApiReturn::error();
```

```
        }
    }

    /**
     * 获取用户tap和qq的绑定信息
     * @param Request $request
     * @return Object
     */
    public function getPlatformTapAndQqBinding(Request $request): object
    {
        $user_id = $request->get('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        return User::getPlatformTapAndQqBinding((int)$user_id);
    }

    /**
     * 绑定tap账号和qq
     * @param Request $request
     * @return object
     */
    public function bindPlatformTapAndQq(Request $request): object
    {
        $user_id = $request->post('user_id');
        if (empty($user_id)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $qq_account = $request->post('qq_account');
        $tap_account = $request->post('tap_account');
        if (empty($tap_account)) {
            return ApiReturn::error('您还没有填写哦～');
        }
        if (empty($tap_account['platform_user_id'])) {
            return ApiReturn::error('您还没有填写哦～');
        }
        return User::bindPlatformTapAndQq($user_id, $tap_account, $qq_account);
    }
    /**
     * 小程序code换unionId
     *
     * @param [type] $code
     */
    // "openid": "o6_bmjrPTlm6_2sgVt7hMZOPxxxx",
    // "session_key": "tiihtNczf5v6AKRyjwExxxx=",
    // "unionid": "o6_bmasdasdsad6_2sgVt7hMZOxxxx",
    // "errcode": 0,
    // "errmsg": "ok"
    public function miniAppLoginSession($code): object
    {
```

```
        try {
            $app = new WechatUser();
            $result = $app->code2Session($code);
            return ApiReturn::success($result);
        } catch (\Throwable $th) {
            //throw $th;
            info($th);
            return ApiReturn::error();
        }
    }

    //微信小程序登录
    public function wechatMiniProgramLogin(UserRequest $request): object
    {
        try {
            $code = $request->post('code');
            if (empty($code)) {
                return ApiReturn::error('参数错误');
            }
            $app = new WechatUser();
            $result = $app->code2Session($code);
            if (empty($result['openid']) || empty($result['session_key']) || empty($result['unionid'])) {
                return ApiReturn::error('登录失败');
            }
            $isApp = $request->post('is_app', true);
            $data = [
                'union_wechat_open_id' => $result['unionid'],
                'wechat_open_id' => $result['openid']
            ];
            $where = ['wechat_open_id' => $data['union_wechat_open_id']];
            $userInfo = V1User::query()->where($where)->first();
            if (!$userInfo) {
                $userInfo = V1User::query()->where('wechat_open_id', $data['wechat_open_id'])->first();
                if ($userInfo) V1User::query()->where('wechat_open_id', $data['wechat_open_id'])->update(['wechat_open_id' =>
  $data['union_wechat_open_id']]); // 更换 openid 为 unionid
            }
            if ($userInfo && empty($userInfo->xcx_openid)) {
                V1User::query()->where('id', $userInfo->id)->update(['xcx_openid' => $data['wechat_open_id']]);
            }
            if ($userInfo && $userInfo->account_status == 2) {
                return ApiReturn::rdata(ApiCode::ACCOUNT_FREEZE);
            }
            // 添加/修改登录数据
            $time = time();
            $ip = Request::createFromGlobals()->getClientIp();
            // 生成 token
            while (true) {
                $token = CommonFnc::generateToken();
                if (!Redis::exists('token' . $token)) break;
            }
            $rdata = [
```

```
            'token' => $token
        ];
        // 查询是否 有该账户  没有 则 创建
        if ($userInfo) {
            $rdata['avatar'] = $userInfo['avatar'];
            $rdata['username'] = $userInfo['username'];
            $rdata['uid'] = $userInfo['id'];
            $rdata['mobile'] = $userInfo['mobile'];
            $rdata['wx'] = $isApp ? (bool)$userInfo['app_wx_openid'] : (bool)$userInfo['wechat_open_id'];  // 是否绑定 wx
            $rdata['qq'] = (bool)$userInfo['qq_open_id'];      // 是否绑定 qq
            $rdata['idcard'] = (bool)$userInfo['idcard'];          // 是否认证身份
            $rdata['qq_num'] = (bool)$userInfo['qq'];          // 是否填写qq号码
            $rdata['is_binding_mobile'] = true;
            $this->roles($rdata);
            $userInfo = $userInfo->toarray();

            Redis::setex('token:' . $token, Code::TOKEN_EXPIRE_TIME, $userInfo['id']);
            V1User::query()->where('id', $userInfo['id'])->update(['login_time' => $time, 'login_ip' => $ip, 'api_token' =>
$token]);
            // 存入redis 数据
            $userInfo['login_time'] = $time;
            $userInfo['login_ip'] = $ip;
            $userInfo['api_token'] = $token;
            // 存入 redis
            (new V1User())->setUserInfoToRedis($userInfo['id'], $userInfo);
            $rdata['user_role'] = $userInfo['user_role'];
        } else {
            $nickname = '用户' . CommonFnc::randomNumeric(10);
            $idata = [];
            $data = array_merge([
                'username' => $nickname,
                'login_time' => $time,
                'login_ip' => $ip,
                'is_robot' => 0, //是否为机器人
                'api_token' => $token,
                'xcx_openid' => $data['wechat_open_id']
            ], $idata);
            // 默认头像
            $data['avatar'] = CommonFnc::defaultAvatar();
            $rs = V1User::query()->firstOrCreate($where, $data)->toArray();
            $userInfo = V1User::query()->find($rs['id'])->toArray();
            Redis::setex('token:' . $token, Code::TOKEN_EXPIRE_TIME, $rs['id']);
            $rdata['avatar'] = $userInfo['avatar'];
            $rdata['username'] = $userInfo['username'];
            $rdata['uid'] = $userInfo['id'];
            $rdata['mobile'] = $userInfo['mobile'];
            $rdata['wx'] = (bool)$userInfo['wechat_open_id'];  // 是否绑定 wx
            $rdata['qq'] = (bool)$userInfo['qq_open_id'];      // 是否绑定 qq
```

```
                $rdata['idcard'] = (bool)$userInfo['idcard'];          // 是否认证身份
                $rdata['qq_num'] = (bool)$userInfo['qq'];          // 是否填写qq号码
                $rdata['is_binding_mobile'] = true;
```

```
                $this->roles($rdata);
                // 查询新数据 存入 redis
                (new V1User())->setUserInfoToRedis($userInfo['id'], $userInfo);
                $rdata['user_role'] = $userInfo['user_role'];
            }
            self::updateSessionKey($rdata['uid'], $result['session_key'], $result['openid']);
            $rdata['wx_login_info'] = [
                'openid' => $result['openid'],
                'unionid' => $result['unionid']
            ];
            return ApiReturn::success($rdata);
        } catch (\Throwable $th) {
            Common::log('微信小程序登录异常', $th);
            return ApiReturn::error();
        }
    }

    //更新微信sessionKey
    public static function updateSessionKey($user_id, $session_key, $openid)
    {
        if (!empty($user_id) && !empty($session_key)) {
            UserSocial::query()->updateOrCreate(['user_id' => $user_id, 'type' => 'wechat_mini_program', 'app_id' =>
'wxd0215facf5f99001'], ['session_key' => $session_key, 'openid' => $openid]);
        }
    }

    /**
     * 用户角色信息
     * @param array $data
     */
    public function roles(array &$data)
    {
        $re = UserUserRole::query()->has('user_role')->where(['user_id' => $data['uid']])->get();
        $data['roles']['is_channel'] = false;
        foreach ($re as $v) {
            if ($v->user_role_id == 7) {
                $data['roles']['is_channel'] = true;
                break;
            }
        }
    }

    /**
     * 个人中心功能区
     */
    public function getUserFunctionTab(Request $request)
    {
        $platform = $request->get('platform');
        try {
            $unButtons = [];
            $where['status'] = 1;
```

```
            $config = ServiceConfig::query()
                ->whereIn('field_name', [
                    'ios_show',
                    'android_show',
                    'ios_user_disable_module',
                    'android_user_disable_module'
                ])->pluck('value', 'field_name');
            if ($platform == 2) {   // ios 模块
                if (!$config['ios_show']) {
                    $unButtons = json_decode($config['ios_user_disable_module'], true);
                }
                $where['ios_status'] = 1;
            } else if ($platform == 1) {    // 安卓模块
                if (!$config['android_show']) {
                    $unButtons = json_decode($config['android_user_disable_module'], true);
                }
                $where['android_status'] = 1;
            }
            $data = UserModuleTab::query()
                ->where($where)
                ->with(['buttons' => function ($q) use ($unButtons, $where) {
                    if ($unButtons) $q->whereNotIn('link', $unButtons);
                    $q->where($where)->select('id', 'tab_id', 'icon', 'title', 'link', 'link', 'weblink', 'desc')->orderBy('sort')->orderBy('id');
                }])->select('id', 'tab_type', 'tab_name')->orderBy('sort')->orderBy('id')->get();
            return ApiReturn::success($data);
        } catch (\Throwable $e) {
            return ApiReturn::error();
        }
    }

    /**
     * 用户认证页面
     * @param Request $request
     * @return Object
     */
    public function userAttestation(Request $request): object
    {
        $userId = $request->get('user_id', 0);
        try {
            $userAttestationConfigs = User::userAttestation($userId);
            return ApiReturn::success($userAttestationConfigs);
        } catch (\Exception $e) {
            return ApiReturn::error();
        }
    }

    /**
     * 意见反馈
     * @param UserRequest $request
     * @return Object
```

```
     * @throws AuthorizationException
     * @throws ValidationException
```

```
 */
public function submitSuggestion(UserRequest $request): object
{
    $uid = $request->post('user_id');
    $content = $request->post('suggestion_content', '');
    $contact = $request->post('contact', '');
    if ($content == null) $content = '';
    if ($contact == null) $contact = '';
    $images = $request->post('images', []);
    $images = is_array($images) ? $images : json_decode($images, true);
    if ($content) {
        $request->validate('submitSuggestion');
    } else if (!$images) {
        return ApiReturn::error('请上传图片或内容');
    }
    try {
        DB::beginTransaction();
        $rs = Suggestion::query()->create([
            'user_id' => $uid,
            'content' => $content,
            'contact' => $contact
        ]);
        if ($images) {
            // 维护时间
            $time = time();
            $arr = [];
            foreach ($images as $k => $v) {
                $arr [] = [
                    'suggestion_id' => $rs->id,
                    'url' => $v,
                    'created_at' => $time,
                    'updated_at' => $time,
                    'sort' => ($k + 1)
                ];
            }
            // 图片限制 截取 前n个图片
            $arr = array_slice($arr, 0, 5);
            SuggestionImage::query()->insert($arr);
        }
        DB::commit();
        return ApiReturn::success((bool)$rs);
    } catch (\Exception $e) {
        DB::rollback();
        return ApiReturn::error();
    }
}

//用户是否设置青少年模式密码
```

```
public function isSetTeenModePas(Request $request): object
{
    $user_id = $request->post('user_id');
    $deviceId = $request->post('device_num');
    if (empty($deviceId) && empty($user_id)) {
        return ApiReturn::success();
        return ApiReturn::error('参数错误');
    }
    return User::isSetTeenModePas($user_id, $deviceId);
}

//校验用户青少年密码是否正确
public function checkTeenModePas(Request $request): object
{
    return ApiReturn::success();
    $user_id = $request->post('user_id');
    $deviceId = $request->post('device_num');
    if (empty($deviceId) && empty($user_id)) {
        return ApiReturn::error('参数错误');
    }
    $password = $request->post('password');
    if (empty($password)) {
        return ApiReturn::error('请输入密码');
    }
    if (!preg_match('/^[0-9]{4}$/', $password)) {
        return ApiReturn::error('密码必须是4位数字');
    }
    return User::checkTeenModePas($user_id, $deviceId, $password);
}

//充值青少年模式密码
public function resetTeenModePass(Request $request): object
{
    $deviceId = $request->post('device_num');
    $user_id = $request->post('user_id');
    $name = $request->post('name');
    if (empty($name)) {
        return ApiReturn::error('请输入姓名');
    }
    $id_card = $request->post('id_card');
    if (empty($id_card)) {
        return ApiReturn::error('请输入身份证号');
    }
    if (!Common::checkIdCard($id_card)) {
        return ApiReturn::error('身份证号格式不正确');
    }
    $age = User::getAge($id_card);
    if ($age < 18) {
        return ApiReturn::error('用户必须年满 18 岁');
    }
```

```
        $password = $request->post('password');
        if (empty($password)) {
            return ApiReturn::error('请输入密码');
        }
        if (!preg_match('/^[0-9]{4}$/', $password)) {
            return ApiReturn::error('密码必须是4位数字');
        }
        if (empty($deviceId) && empty($user_id)) {
            return ApiReturn::success();
            return ApiReturn::error('参数错误');
        }
        return User::resetTeenModePass($user_id, $deviceId, $password, $name, $id_card);
    }

    //设置模式
    public function setTeenMode(Request $request): object
    {
        $status = $request->post('status');
        if (!in_array($status, [0, 1])) {
            return ApiReturn::success();
            return ApiReturn::error('参数错误');
        }
        $password = $request->post('password');
        $type = $request->post('type');
        if ($status == 0 && empty($type)) {
            if (empty($password)) {
                Log::info('设置青少年模式请输入密码');
                return ApiReturn::error('请输入密码');
            }
            //密码必须是4位数字
            if (!preg_match('/^[0-9]{4}$/', $password)) {
                return ApiReturn::error('密码必须是4位数字');
            }
        }
        $deviceId = $request->post('device_num');
        $user_id = $request->post('user_id');
        if (empty($deviceId) && empty($user_id)) {
            return ApiReturn::success();
            return ApiReturn::error('参数错误');
        }
        return User::setTeenMode($deviceId, $user_id, $status, $password);
    }

    //检验青少年模式身份证号
    public function checkTeenModeIdCard(Request $request): object
    {
        $name = $request->post('name');
        if (empty($name)) {
            return ApiReturn::error('请输入姓名');
        }
```

```
        $id_card = $request->post('id_card');
        if (empty($id_card)) {
            return ApiReturn::error('请输入身份证号');
        }
        if (!Common::checkIdCard($id_card)) {
            return ApiReturn::error('身份证号格式不正确');
        }
        $age = User::getAge($id_card);
        if ($age < 18) {
            return ApiReturn::error('用户必须年满 18 岁');
        }
        return ApiReturn::success();
    }

    /**
     * 历史记录
     * @param Request $request
     * @return Object
     */
    public function getUserBrowseList(Request $request): object
    {
        $limit = $request->get('limit', 10);
        $uid = $request->get('user_id');
        $is_review = intval($request->get('is_review', 0));
        $deviceId = $request->get('device_num');
        return User::getUserBrowseList($uid, $is_review, $limit, $deviceId);
    }

    /**
     * 合并资源
     * @param array $articleList
     * @param $user_id
     * @return array
     * @throws ContainerExceptionInterface
     * @throws NotFoundExceptionInterface
     */
    public static function handleResorce($articleList, $user_id, $platform): array
    {
        $data = [];
        if (!empty($articleList['data'])) {
            foreach ($articleList['data'] as $k => $v) {
                $article = $v['article'];
                $article['label_list'] = empty($v['article']['label_list']) ? [] : json_decode($v['article']['label_list'], true);
                $article['is_like'] = UserLike::query()
                    ->where('relevance_id', $v['article']['id'])
                    ->where('user_id', $user_id)
                    ->where('type', $v['article']['type'])
                    ->where('status', 1)
                    ->exists();
                $article['is_no_like'] = 0;
```

```
                if (isset($v['article']['user_award'])) {
                    $article['is_awarded'] = $v['article']['user_award']['task_award_id'] > 0;
                }
                if (isset($v['article']['create_time'])) {
                    $article['create_time'] = CommonFnc::disposeDateTime(strtotime($v['article']['create_time']));
                }
                $article['content'] = $v['article']['text_content'] ? mb_substr($v['article']['text_content'], 0, 255) : '';
                unset($article['text_content']);
                // ios 要不同的数据格式
                if ($platform == 2) {
                    $article['video'] = empty($article['video']['url']) ? '' : $v['article']['video']['url'];
                    $article['images'] = empty($article['images']) ? [] : array_column($v['article']['images'], 'url');
                    foreach ($article['images'] as $k1 => $v1) {
                        $article['images'][$k1] = Common::concatUrlStr($v1);
                    }
                    if (!empty($v['article']['video']) && strpos($v['article']['cover'], 'sologame/webapp/transparent.png')) {
                        $article['cover'] = $v['article']['video'] . '?x-oss-process=video/snapshot,t_1000,m_fast';
                    }
                } else {
                    if (!empty($v['article']['video']) && !empty($v['article']['video']['url']) && strpos($v['article']['cover'], 'sologame/webapp/transparent.png')) {
                        $article['cover'] = $v['article']['video']['url'] . '?x-oss-process=video/snapshot,t_1000,m_fast';
                    }
                    if (!$article['video']) $v['article']['video'] = (object)[];
                    foreach ($v['article']['images'] as $k1 => $v1) {
                        $article['images'][$k1] = [
                            'article_id' => $article['id'],
                            'url' => Common::concatUrlStr($v1['url']),
                        ];
                    }
                }
                if (empty($v['article']['game'])) {
                    $article['game'] = (object)[];
                }
                if (empty($v['article']['circle'])) {
                    $article['circle'] = (object)[];
                }
                if (isset($v['article']['rss_guid'])) {
                    $article['content'] = $v['article']['title'];
                }
                $data[] = $article;
            }
        }
        return $data;
    }

    /**
     * 差评/取消差评
     * @param Request $request
     * @return Object
     * @throws \Throwable
```

```
     */
    public function addUserNoLike(Request $request): object
    {
        $authorId = intval($request->post('author_id', 0));
        $relevancId = intval($request->post('relevance_id', 0));
        $type = intval($request->post('type', 1));
        $userId = $request->post('user_id');
        try {
            if (Redis::exists('user:nolike:' . $userId)) {
                return ApiReturn::error('操作太频繁');
            }
            if (empty($authorId) || empty($relevancId)) {
                return ApiReturn::rdata(ApiCode::PARAM_ERROR);
            }
            if (!in_array($type, [1, 2, 3, 4, 5, 6, 7, 8, 9])) {
                return ApiReturn::rdata(ApiCode::PARAM_ERROR);
            }
            $re = User::addUserNoLike($userId, $authorId, $relevancId, $type);
            return ApiReturn::success(['status' => $re], $re == 1 ? '差评成功' : '取消差评');
        } catch (\Exception $e) {
            return ApiReturn::error();
        }
    }

    /**
     * 收藏列表
     * @param Request $request
     * @return Object
     * @throws ContainerExceptionInterface
     * @throws NotFoundExceptionInterface
     */
    public function getUserCollectList(Request $request): object
    {
        $limit = $request->get('limit', 10);
        $uid = $request->get('user_id');
        $platform = $request->get('platforma', 1);
        try {
            $type = intval($request->get('type', 0));
            $type = $type ? [$type] : array_values([1, 2, 3, 4, 5]);
            $list = UserCollect::query()
                ->whereIn('article_type', $type)
                ->where('user_id', $uid)
                ->where('status', 1)
                ->orderBy('update_time', 'desc')
                ->whereHas('article', function ($query) {
                    $query->where('status', 1);
                    $query->where('check_status', 2);
                })
                ->with('article', function ($query) {
                    $query->with([
```

```
                        'user' => function ($query) {
                            $query->select('id', 'username', 'avatar');
                        },
                        'game' => function ($query) {
                            $query->select('id', 'game_name', 'game_icon');
                        },
                        'circle' => function ($query) {
                            $query->select('id', 'circle_name', 'avatar');
                        },
                        'video' => function ($query) {
                            $query->select('id', 'url', 'article_id', 'duration');
                        },
                        'images']);
                    $query->select('id', 'title', 'user_id', 'cover', 'type', 'game_id', 'circle_id', 'text_content', 'label_list', 'create_time',
'rss_guid', 'rss_task_id', 'link_type', 'link_url');

                })
                ->paginate($limit)
                ->toArray();
```

```
            $list = empty($list) ? [] : $list;
            CommonFnc::paginateReturnUnset($list);
            $data = self::handleResorce($list, $uid, $platform);
            $res = [
                'total' => $list['total'] ?? 0,
                'current_page' => $list['current_page'] ?? 1,
                'last_page' => $list['last_page'] ?? 1,
                'data' => $data,
                'platform' => $platform
            ];
            return ApiReturn::success($res);
        } catch (\Exception $e) {
            return ApiReturn::error($e->getMessage());
        }
    }

    /**
     * 修改用户推送设置
     * @catalog
     * @title 修改用户推送设置
     * @description 描述信息
     * @method post
     * @url /api/user/editPushConfigs
     * @param field_name 必选 string 字端类型：comment_status 评论 ,like_status 点赞/收藏,attention_status 首次关
,remind_status 被@,system_status 系统
     * @param field_value 必选 boolean true 开，false 关
     * @return {"code":200,"message":"ok"}
     */
    public function editPushConfigs(Request $request)
    {
        $fieldName = (string)$request->post('field_name', '');
        $fieldValue = (bool)$request->post('field_value', true);
        if (!in_array($fieldName, ['comment_status', 'like_status', 'attention_status', 'remind_status', 'system_status'])) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
```

```
        $userId = $request->post('user_id');
        try {
            UserPushConfig::query()->where(['user_id' => $userId])->update([$fieldName => $fieldValue]);
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('修改用户推送配置信息', $throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 添加阿里云push绑定记录
     * @param Request $request
     * @return Object
     */
    public function addUserDevice(Request $request): object
    {
        $deviceNum = $request->post('device_num', '');
        $platform = $request->post('platform', 1);
        $uni_app = $request->post('uni_app', 0);
        $userId = $request->post('user_id', 0);
        Log::info('添加用户设备信息入参', [
            'device_num' => $deviceNum,
            'platform' => $platform,
            'uni_app' => $uni_app,
            'user_id' => $userId
        ]);
        try {
            if (!empty($deviceNum) && in_array($platform, [1, 2]) && $uni_app == 0) {
                $deviceIds = $platform == 1 ? AliPush::getAndroidDeviceIds($deviceNum) : AliPush::getIosDeviceIds($deviceNum);
                if (empty($deviceIds)) {
                    return ApiReturn::success('阿里推送设备ID为空');
                }
                $data = [
                    'ali_device_num' => $deviceIds['DeviceId'][0],
                    'device_type' => $uni_app != 0 ? 3 : $platform
                ];
                $info = UserAlipush::query()->where($data)->first();
                if (empty($info)) {
                    if ($userId > 0) {
                        UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
                    }
                    $data['user_id'] = $userId;
                    $data['device_num'] = $deviceNum;
                    $data['check_date'] = date('Y-m-d H:i:s');
                    UserAlipush::query()->create($data);
                } else {
                    $info->device_num = $deviceNum;
                    if ($userId > 0 && $userId != $info->user_id) {
                        UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
```

```
                        $info->user_id = $userId;
                    }
                    $info->check_date = date('Y-m-d H:i:s');
                    $info->status = 1;
                    $info->save();
                }
            }
            if (!empty($deviceNum) && $uni_app != 0) {
                $data = [
                    'ali_device_num' => $deviceNum,
                    'device_type' => 3
                ];
                $info = UserAlipush::query()->where($data)->first();
                if (empty($info)) {
                    if ($userId > 0) {
                        UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
                    }
                    $data['user_id'] = $userId;
                    $data['device_num'] = $deviceNum;
                    $data['check_date'] = date('Y-m-d H:i:s');
                    UserAlipush::query()->create($data);
                } else {
                    $info->device_num = $deviceNum;
                    if ($userId > 0 && $userId != $info->user_id) {
                        UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
                        $info->user_id = $userId;
                    }
                    $info->check_date = date('Y-m-d H:i:s');
                    $info->status = 1;
                    $info->save();
                }
            }
            return ApiReturn::success();
        } catch (\Exception $e) {
            Common::log('添加阿里云push绑定记录', $e);
            ErrorLog::dispatch(Route::current()->getActionName(), CommonFnc::getRealIP(), json_encode($e), ['file' => $e->getFile(), 'line' => $e->getLine(), 'code' => $e->getCode(), 'message' => $e->getMessage()], time())->onQueue('error_log');
            return ApiReturn::error();
        }
    }

    //用户发布文章
    public function userArticles(Request $request)
    {
        $userId = $request->get('user_id', 0);
        $limit = $request->get('limit', 10);
        $is_review = $request->get('is_review');
        $author_id = $request->get('author_id');
        $device_id = $request->get('device_num');
        return User::userArticles($userId, $limit, $is_review, $author_id, $device_id);
    }
```

```

    //用户收藏文章
    public function userCollectArticles(Request $request)
    {
        $userId = $request->get('user_id', 0);
        $limit = $request->get('limit', 10);
        $authorID = $request->get('author_id');
        $device_id = $request->get('device_num');
        return User::userCollectArticles($authorID, $userId, $limit, $device_id);
    }

    //用户点赞文章
    public function userLikeArticles(Request $request)
    {
        $userId = $request->get('user_id', 0);
        $limit = $request->get('limit', 10);
        $authorID = $request->get('author_id');
        $device_id = $request->get('device_num');
        return User::userLikeArticles($authorID, $userId, $limit, $device_id);
    }

    //勋章列表
    public function medals(Request $request)
```

```
{
    $userId = $request->get('user_id', 0);
    return User::medals($userId);
}

//佩戴和取消佩勋章
public function wearMedal(Request $request)
{
    $userId = $request->post('user_id');
    if (empty($userId)) {
        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
    }
    $medalId = $request->post('medal_id');
    if (empty($medalId)) {
        return ApiReturn::error('请选择要佩戴的勋章');
    }
    $type = $request->post('type', 1);
    return User::wearMedal($userId, $medalId, $type);
}

//用户关注圈子列表
public function circleAttentionList(Request $request)
{
    $authorId = $request->get('author_id', 0);
    $limit = $request->get('limit', 10);
    $user_id = $request->get('user_id', 0);
    $type = $request->get('type', 1);
```

```
    $name = $request->get('name');
    return User::userAttentionCircle($authorId, $user_id, $limit, $type, $name);
}

//用户关注用户列表
public function AttentionUserList(Request $request)
{
    try {
        $user_id = $request->get('user_id');
        $uid = $request->get('uid', 0); // 要查看的人
        if ($uid == 0) $uid = $user_id;
        $limit = $request->get('limit', 10);
        $name = $request->get('name');
        return User::AttentionUserList($uid, $user_id, $limit, $name);
    } catch (\Exception $e) {
        return ApiReturn::error($e->getMessage());
    }
}

//用户是否在国外
public function isForeign(Request $request)
{
    return User::isForeign();
}

//生成小程序generatescheme链接
```

```
public function generateScheme(Request $request)
{

   $path = $request->post('path');
   $query = $request->post('query');
   $env_version = $request->post('env_version');
   return User::generateScheme($path, $query, $env_version);
}

//通过手机号校验用户为新用户或者是否已绑定微信
public function checkMobile(Request $request)
{
   $mobile = $request->post('mobile');
   //校验手机号是否正确
   if (empty($mobile) || !preg_match('/^1[3456789][0-9]{9}$/', $mobile)) {
      return ApiReturn::error('请输入正确的手机号');
   }
   return User::checkMobile($mobile);
}

//获取微信手机号
public function getWxMobile(Request $request)
{
   $user_id = $request->post('user_id');
```

```
   if (empty($user_id)) {
      return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
   }
   $type = $request->post('type', 1);
   $code = $request->post('code');
   if ($type == 1) {
      if (empty($code)) {
         return ApiReturn::error('非法code');
      }
   }
   $app_code = $request->post('app_code');
   return User::getWxMobile($user_id, $code, $type, $app_code);
}

//通过code获取token
public function loginByCode(Request $request)
{
   Log::info('loginByCode',$request->all());
   $code = $request->post('code');
   if (empty($code)) {
      return ApiReturn::error('非法code');
   }
   return User::loginByCode($code);
}

//校验小程序的用户和app用户ID是否一致
public function checkAppUser(Request $request)
{
   $user_id = $request->get('user_id');
```

```
        $app_user_id = $request->get('app_user_id');
        if (empty($user_id) || empty($app_user_id)) {
            return ApiReturn::error('非法参数');
        }
        return User::checkAppUser($user_id, $app_user_id);
    }
    //绑定社媒账号
    public function bindSocialAccount(Request $request)
    {
        $userId = $request->post('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $type = $request->post('type');
        if($type == 'steam'){
            $code = $request->post('code');
            if(empty($code)){
                return ApiReturn::error('请输入好友码');
            }
            return User::bindPlatformFriendCode($userId, $code,  $type);
        }
```

```
        $accounts = $request->post('accounts');
        if(empty($accounts)){
            return ApiReturn::error('请选择要绑定的账号');
        }
        if(!is_array($accounts)){
            return ApiReturn::error('请选择要绑定的账号');
        }
        $domainRules = [
            'dy'   => ['douyin.com'],
            'xhs'  => ['xiaohongshu.com'],
            'wb'   => ['weibo.com'],
            'bili' => ['bilibili.com', 'b23.tv'],
        ];
        $cnMap = [
            'dy' => '抖音',
            'xhs' => '小红书',
            'wb'  => '微博',
            'bili' => '哔哩哔哩',
        ];

        foreach ($accounts as $k =>  $value) {
            if (!isset($domainRules[$value['type']])) {
                return ApiReturn::error("不支持的平台类型 ". $value['type']);
            }
            if(!empty($value['link'])){
                if (!filter_var($value['link'], FILTER_VALIDATE_URL)) {
                    preg_match('/https?:\/\/[^\s]+/u', $value['link'], $matches);
                    if (isset($matches[0])) {
                        $value['link'] = $matches[0]; // 提取第一个 URL 替换原字段
                        $accounts[$k]['link'] = $matches[0];
                    }
                }

                if (!filter_var($value['link'], FILTER_VALIDATE_URL)) {
                    return ApiReturn::error("{$cnMap[$value['type']]} 链接不是有效的网址格式");
                }
                $host = parse_url($value['link'], PHP_URL_HOST);
                $valid = false;
                foreach ($domainRules[$value['type']] as $domain) {
                    if (stripos($host, $domain) !== false) {
                        $valid = true;
                        break;
                    }
                }
                if (!$valid) {
                    return ApiReturn::error("{$cnMap[$value['type']]} 链接不合法");
                }
            }
        }
        return User::bindSocialAccount($userId, $accounts);
```

```
    }
    //绑定平台好友码
    public function bindPlatformFriendCode(Request $request)
    {
        $userId = $request->post('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $code = $request->post('code');
        if(empty($code)){
            return ApiReturn::error('请输入好友码');
        }
        $platform = $request->post('type','steam');
        return User::bindPlatformFriendCode($userId, $code,$platform);
    }
    //获取平台用户信息
    public function getPlatformUserInfo(Request $request)
    {
        $userId = $request->get('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $platform = $request->get('type','steam');
        return User::getPlatformUserInfo($userId,$platform);
    }
    //设置平台隐私设置
    public function setPlatformUserPrivacy(Request $request)
    {
        $userId = $request->post('user_id');
        if (empty($userId)) {
            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
        }
        $platform = $request->post('type','steam');
        $privacy_profile = $request->post('privacy_profile');
        if(empty($privacy_profile)){
            return ApiReturn::error('请选择隐私设置');
        }
        if(!in_array($privacy_profile,[1,2,3])){
            return ApiReturn::error('请选择正确的隐私设置');
        }
        return User::setPlatformUserPrivacy($userId,$platform,$privacy_profile);
    }



}

```

```
<?php

namespace App\Http\Controllers\api\v1\channel;

use App\Http\Controllers\Controller;
use App\Models\v1\channel\CreatorSource;
use App\Models\v1\channel\CreatorTask;
use App\Models\v1\channel\CreatorTaskImport;
use App\Models\v1\channel\UserCreatorResourceConfigAudit;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\Common;
use App\Service\v1\Imports\CreatorSourceImport;
use Illuminate\Http\Request;
use App\Service\v1\channel\Channel as ChannelService;
use Maatwebsite\Excel\Facades\Excel;

class ChannelController extends Controller
{
    /**
     * 创作营列表
     * @param Request $request
     * @return Object
     */
    public function tasks(Request $request)
    {
        $userId = $request->get('user_id');
        // $page = (int)$request->get('page',1);
        $limit = (int)$request->get('limit',10);
        // $page = ($page - 1) * $limit;
        $keyword = (string)$request->get('keyword','');
        $date = (string)$request->get('date','');
        try {
            // $list = ChannelService::tasks($userId,$page,$limit,$keyword,$date);
            $list = ChannelService::pagnateTasks($userId,$limit,$keyword,$date);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('创作营列表',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 创作营提费用列表
     * @param Request $request
     * @return Object
     */
    public function supplierTasks(Request $request)
    {
        $userId = $request->get('user_id');
```

```
        $page = (int)$request->get('page',1);
        $type = (int)$request->get('type',1);
        $limit = (int)$request->get('limit',10);
        $page = ($page - 1) * $limit;
        try {
            $list = ChannelService::supplierTasks($userId,$type,$page,$limit);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('创作营提费用列表',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 任务内容列表（渠道进度）
     * @param Request $request
     * @return Object
     */
    public function taskContents(Request $request)
    {
        $taskId = (int)$request->get('task_id',0);
        if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->get('user_id');
        $page = (int)$request->get('page',1);
        $limit = (int)$request->get('limit',10);
        $page = ($page - 1) * $limit;
        try {
            $list = ChannelService::taskContents($userId,$taskId,$page,$limit);
            return ApiReturn::success($list);
        } catch (\Throwable $throwable) {
            Common::log('任务内容列表（渠道进度）',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 渠道导入内容
     * @param Request $request
     * @return Object
     */
    public function creatorSourceImport(Request $request)
    {
        $taskId = (int)$request->input('task_id',0);
        $fileInfo = $request->file('file_path');
        if($taskId <= 0 || empty($fileInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        try {
            $importStatus = CreatorTask::query()->where(['id' => $taskId])->value('import_status');
            if(empty($importStatus)) return ApiReturn::error(ApiCode::ROLE_ERROR['msg'],ApiCode::ROLE_ERROR['code']);
            CreatorSource::query()->where(['creator_task_id' => $taskId])->delete();
```

```
            CreatorTaskImport::query()->updateOrCreate([
                'task_id' => $taskId
            ],[
                'name' => $fileInfo->getClientOriginalName(),
                'file_size' => ceil($fileInfo->getSize()/1024)
            ]);
            Excel::import(new CreatorSourceImport($taskId),$fileInfo);
            return ApiReturn::success();
        }catch (\Throwable $throwable) {
            return ApiReturn::error($throwable->getMessage());
        }
    }

    /**
     * 渠道修改内容链接
     * @param Request $request
     * @return Object
     */
    public function updateCreatorSource(Request $request)
    {
        $getContentId = (int)$request->get('get_content_id',0);
        $url = (string)$request->get('url','');
        if($getContentId <= 0 || !preg_match("/http[s]?:\/\/[\w.]+[\w\/]*[\w.]*\??[\w=&\+\%]*/is",$url,$match)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->get('user_id');
        try {
            $creatorSourceInfo = CreatorSource::where('id',$getContentId)->whereHas('creator_task', function ($q) use ($userId) {
                $q->where('user_id',$userId);
            })->first();
            if(empty($creatorSourceInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            if ($url == $creatorSourceInfo->url) return ApiReturn::success();
            $urlPath = parse_url($match[0]);
            $creatorSourceInfo->url = $urlPath['scheme'].'://'.$urlPath['host'].$urlPath['path'];
            // 初始化参数
            $creatorSourceInfo->remark = '';
            $creatorSourceInfo->url_game_name = '';
            $creatorSourceInfo->game_time_str = '';
            $creatorSourceInfo->game_time = 0;
            $creatorSourceInfo->url_status = 0;
            $creatorSourceInfo->practical_released_at = null;
            $creatorSourceInfo->save();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('渠道修改内容链接',$throwable);
            return ApiReturn::error();
    }
}

/**
 * 渠道资源配置列表
 * @param Request $request
```

```
 * @return Object
 */
public static function getResourceConfig(Request $request)
{
    $userId = $request->get('user_id');
    $limit = (int)$request->get('limit',10);
    $lists = ChannelService::resourceConfigList($userId,$limit);
    return ApiReturn::success($lists);
}

/**
 * 渠道资源配置 平台列表
 * @param Request $request
 * @return Object
 */
public function getResourceConfigPlatforms(Request $request)
{
    $userId = $request->get('user_id');
    $lists = ChannelService::resourceConfigPlatforms($userId);
    return ApiReturn::success($lists);
}

/**
 * 渠道资源配置 资源列表
 * @param Request $request
 * @return Object
 */
public function getResourceConfigResources(Request $request)
{
    $userId = $request->get('user_id');
    $lists = ChannelService::resourceConfigResources($userId);
    return ApiReturn::success($lists);
}

/**
 * 渠道资源配置 历史提交记录
 * @param Request $request
 * @param int $config_id 资源配置ID
 * @return Object
 */
public function getResourceConfigHistory(Request $request)
{
    $userId = $request->get('user_id');
    $configId = $request->get('config_id', 0);
    if ($configId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
    $lists = ChannelService::resourceConfigHistory($userId, $configId);
    return ApiReturn::success($lists);
    }

    /**
```

```
     * 渠道资源配置 新增审核
     * @param Request $request
     * @param int $creator_resource_config_id 资源 id
     * @param int $platform_id 平台ID
     * @return Object
     */
    public function addResourceConfig (Request $request)
    {
        $userId = $request->get('user_id');
        $configId = $request->get('creator_resource_config_id', 0);
        $platformId = $request->get('platform_id', 0);
        $unitPrice = $request->get('unit_price', 0);
        if (!preg_match('/^\d{1,4}(\.\d{1,4})?$/',$unitPrice)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
        if ($configId <= 0 || $platformId <= 0 || $unitPrice <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
        return ChannelService::addResourceConfig($userId, $configId, $platformId, $unitPrice);
    }

    /**
     * 渠道资源配置 修改配置审核
     * @param Request $request
     * @param int $config_id 资源配置ID
     * @param int $platform_id 平台ID
     * @param int $unit_price 单价
     * @return Object
     */
    public function updateResourceConfig (Request $request)
    {
        $userId = $request->get('user_id');
        $configId = $request->get('config_id', 0);
        $unitPrice = $request->get('unit_price', 0);
        if (!preg_match('/^\d{1,4}(\.\d{1,4})?$/',$unitPrice)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
        if ($configId <= 0 || $unitPrice <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
        return ChannelService::updateResourceConfig($userId, $configId, $unitPrice);
    }
}

```

```
<?php

namespace App\Http\Controllers\api\v1\channel;

use App\Http\Controllers\Controller;
use App\Http\Requests\ChannelBinding;
use App\Jobs\CheckCreatorSourceUrl;
use App\Models\v1\ActivityCenterPlatform;
use App\Models\v1\channel\CreatorSource;
use App\Models\v1\channel\CreatorTask;
```

```php
use App\Models\v1\channel\UserDefaultPlatform;
use App\Models\v1\ServiceConfig;
use App\Models\v1\User;
use App\Service\v1\ApiCode;
use App\Service\v1\ApiReturn;
use App\Service\v1\channel\Channel as ChannelService;
use App\Service\v1\Code;
use App\Service\v1\Common;
use App\Service\v1\RedisKey;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Redis;

class FramerController extends Controller
{

    /**
     * 创作者是否绑定平台账号
     * @param ChannelBinding $request
     * @return Object
     * @throws \Illuminate\Auth\Access\AuthorizationException
     * @throws \Illuminate\Validation\ValidationException
     */
    public function channelBinding(ChannelBinding $request)
    {
        $request->validate('task_check');
        $taskId = (int)$request->get('task_id',0);
        if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->get('user_id');
        try {
            $creatorTaskInfo = CreatorTask::query()->where(['id' => $taskId])->select('channel','end_status')->first();
            if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            if($creatorTaskInfo->end_status == 1) return ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
            if($creatorTaskInfo->task_type != 0) return ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);
            $info = ChannelService::channelBinding($userId,$taskId,$creatorTaskInfo->channel);
            return ApiReturn::success($info);
        } catch (\Throwable $throwable) {
            Common::log('创作者是否绑定平台账号',$throwable);
            return ApiReturn::error();
        }
    }
```

```php

    /**
     * 创作者领取词条
     * @param ChannelBinding $request
     * @return Object
     * @throws \Illuminate\Auth\Access\AuthorizationException
     * @throws \Illuminate\Validation\ValidationException
     */
    public function getContent(ChannelBinding $request)
    {
        $request->validate('task_check');
        $taskId = (int)$request->get('task_id',0);
        if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->get('user_id');
        try {
            $creatorTaskInfo = CreatorTask::query()->where(['id' => $taskId])->select('channel','end_status','get_time_limit','binding_platform_switch')->first();
            if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            if($creatorTaskInfo->end_status == 1) return ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
            if($creatorTaskInfo->task_type != 0) return ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);
            $count = CreatorSource::query()->where(['creator_task_id' => $taskId,'channel' => $creatorTaskInfo->channel,'user_id' => $userId])->count();
            if($count > 0) return ApiReturn::error(ApiCode::SYSTEM_TASK_REPEAT_GET['msg'],ApiCode::SYSTEM_TASK_REPEAT_GET['code']);
//            $getContentTime = ServiceConfig::query()->where(['field_name' => 'get_content_time'])->value('value');
            $redisTime = (int)Redis::get(RedisKey::GET_CONTENT_TIME.$taskId);
            $nowTime = time();
            $time = $redisTime + $creatorTaskInfo->get_time_limit - $nowTime;
            if($time > 0) return ApiReturn::success(['status' => false,'time' => $redisTime + $creatorTaskInfo->get_time_limit,'msg' => ApiCode::GET_CONTENT_TIME['msg'],'creator_source' => new \stdClass()]);
            $info = CreatorSource::query()->where(['creator_task_id' => $taskId,'channel' => $creatorTaskInfo->channel,'user_id' => 0])
                ->where('released_at','<=',date('Y-m-d'))
                ->select('id','content','score')->first();
            if(empty($info)) return ApiReturn::error(ApiCode::GET_CONTENT['msg'],ApiCode::GET_CONTENT['code']);
            $info->user_id = $userId;
            $info->get_time = date('Y-m-d H:i:s',$nowTime);
            $info->save();
            if($creatorTaskInfo->binding_platform_switch == 1) {
                $checkContentTime = ServiceConfig::query()->where(['field_name' => 'check_content_time'])->value('value');
                dispatch(new CheckCreatorSourceUrl($info->id))->onQueue('check_creator_source_url')->delay(now()->addSeconds($checkContentTime));
            }
            Redis::set(RedisKey::GET_CONTENT_TIME.$taskId,$nowTime,'EX',3600);
            return ApiReturn::success(['status' => true,'time' => 0,'msg' => '','creator_source' => $info]);
        } catch (\Throwable $throwable) {
            Common::log('创作者领取词条',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 创作者提交平台连接
     * @param Request $request
     * @return Object
     */
```

```
    public function addContentUrl(Request $request)
    {
        $getContentId = (int)$request->post('get_content_id',0);
        $url = (string)$request->post('url','');
        if($getContentId <= 0 || !preg_match("/http[s]?:\/\/[\w.]+[\w\/]*[\w.]*\??[\w=&\+\%]*/is",$url,$match)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->post('user_id');
        try {
            $creatorSourceInfo = CreatorSource::where(['id' => $getContentId,'user_id' =>
$userId])->with('creator_task')->select('id','url','creator_task_id')->first();
            if(empty($creatorSourceInfo)) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            // 需要收集截图
            if ($creatorSourceInfo->creator_task->need_pic) {
                $pic = (string)$request->post('pic','');
                if(empty($pic)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
                $creatorSourceInfo->pic = $pic;
            }
            $urlPath = parse_url($match[0]);
            $creatorSourceInfo->url = $urlPath['scheme'].'://'.$urlPath['host'].$urlPath['path'];
            $creatorSourceInfo->save();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('创作者提交平台连接',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 创作者提交图片
     * @param Request $request
     * @return Object
     */
    public function addOrUpdateContentPic(Request $request)
    {
        $pic = (string)$request->post('pic','');
        if(empty($pic)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->post('user_id');
        $taskId = (int)$request->get('task_id',0);
        if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        try {
            $creatorTaskInfo = CreatorTask::where(['id' =>
$taskId])->with('game')->select('channel','end_status','task_type','unit_price')->first();
            if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            if($creatorTaskInfo->end_status == 1) return
ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
            if($creatorTaskInfo->task_type != 1) return
ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);

            $creatorSourceInfo = CreatorSource::where(['creator_task_id' => $taskId,'user_id' =>
$userId])->with('creator_task')->select('id','url','creator_task_id')->first();
            if(empty($creatorSourceInfo)) {
                $creatorSourceInfo = new CreatorSource();
                // $creatorSourceInfo->url_status = 1;
                $creatorSourceInfo->price = $creatorTaskInfo->unit_price;
                $creatorSourceInfo->creator_task_id = $taskId;
                $creatorSourceInfo->user_id = $userId;
```

```
                $creatorSourceInfo->channel = $creatorTaskInfo->channel;
                $creatorSourceInfo->game_name = $creatorTaskInfo->game ? $creatorTaskInfo->game->game_name : '';
```

```
            }
            $creatorSourceInfo->pic = $pic;
            $creatorSourceInfo->save();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('创作者提交图片',$throwable);
            return ApiReturn::error();
        }
    }


    /**
     * 创作者提交文本任务
     * @param Request $request
     * @return Object
     */
    public function addOrUpdateCreatorFileTaskSource(Request $request)
    {
        $pic = (string)$request->post('pic','');
        $content = (string)$request->post('content','');
        $validCount = (int)$request->post('valid_count', 1);
        $file = (string)$request->post('file','');
        // 三个参数不能同时为空
        if(empty($pic) && empty($content) && empty($file)) return 
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        if ($validCount <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
        $userId = $request->post('user_id');
        $taskId = (int)$request->get('task_id',0);
        if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        try {
            $creatorTaskInfo = CreatorTask::where(['id' => 
$taskId])->with('game')->select('channel','end_status','task_type','unit_price')->first();
            if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
            if($creatorTaskInfo->end_status == 1) return 
ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
            if($creatorTaskInfo->task_type != 2) return 
ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);

            $creatorSourceInfo = CreatorSource::where(['creator_task_id' => $taskId,'user_id' => 
$userId])->with('creator_task')->select('id','url','creator_task_id')->first();
            if(empty($creatorSourceInfo)) {
                $creatorSourceInfo = new CreatorSource();
                // $creatorSourceInfo->url_status = 1;
                $creatorSourceInfo->price = $creatorTaskInfo->unit_price;
                $creatorSourceInfo->creator_task_id = $taskId;
                $creatorSourceInfo->user_id = $userId;
                $creatorSourceInfo->channel = $creatorTaskInfo->channel;
                $creatorSourceInfo->game_name = $creatorTaskInfo->game ? $creatorTaskInfo->game->game_name : '';
            }
            $creatorSourceInfo->pic = $pic;
            $creatorSourceInfo->content = $content;
            $creatorSourceInfo->valid_count = $validCount;
            if ($content) {
                $creatorSourceInfo->content_md5 = md5($content);
            }
            $creatorSourceInfo->file = $file;
            $creatorSourceInfo->save();
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('创作者提交图片',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 创作者绑定平台账户
     * @param ChannelBinding $request
     * @return Object
     * @throws \Illuminate\Auth\Access\AuthorizationException
     * @throws \Illuminate\Validation\ValidationException
     */
    public function addChannelBinding(ChannelBinding $request)
    {
        $request->validate('binding');
        $params = $request->post();
        try {
            $count = UserDefaultPlatform::query()->where(['user_id' => $params['user_id'],'platform_type' =>
$params['platform_type']])->count();
            if($count == 0) {
                UserDefaultPlatform::query()->create($params);
            }
            return ApiReturn::success();
        } catch (\Throwable $throwable) {
            Common::log('创作者绑定平台账户',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 绑定平台列表
     * @param Request $request
     * @return Object
     */
    public function channels(Request $request)
    {
        $userId = $request->get('user_id');
        try {
            $platformIds = UserDefaultPlatform::query()->where(['user_id' => $userId])->pluck('platform_type')->toArray();
            $platformTypes = ActivityCenterPlatform::query()->select('id','name','icon')->get();
            foreach ($platformTypes as &$platformType) {
                $platformType->is_binding = in_array($platformType->id,$platformIds) ? true : false;
            }
            return ApiReturn::success(['list' => $platformTypes]);
        } catch (\Throwable $throwable) {
```

```
            Common::log('绑定平台列表',$throwable);
            return ApiReturn::error();
```

```
        }
    }

    /**
     * 领取词条列表/创作的内容
     * @param Request $request
     * @return Object
     */
    public function getContents(Request $request)
    {
        $userId = $request->get('user_id');
        $page = (int)$request->get('page',1);
        $limit = (int)$request->get('limit',10);
        $page = ($page - 1) * $limit;
        try {
            $creatorSources = CreatorSource::where(['user_id' => $userId])
                ->select('id','get_time','content','file','pic','created_at')
                ->offset($page)->limit($limit)->orderBy('id','desc')->get();
            foreach ($creatorSources as &$creatorSource) {
                $creatorSource->content = mb_strlen($creatorSource->content) > 50 ?
mb_substr($creatorSource->content,0,50).'...' : $creatorSource->content;
                if ($creatorSource->get_time == null) {
                    $creatorSource->get_time = $creatorSource->created_at;
                    $creatorSource->get_time = $creatorSource->get_time->format('Y-m-d H:i:s');
                }
                $creatorSource->file = Common::concatUrlStr($creatorSource->file);
                $creatorSource->pic = Common::concatUrlStr($creatorSource->pic);
                $creatorSource->created_at = null;
            }
            return ApiReturn::success(['creator_sources' => $creatorSources]);
        } catch (\Throwable $throwable) {
            Common::log('领取词条列表',$throwable);
            return ApiReturn::error();
        }
    }

    /**
     * 创作者添加qq号
     * @param Request $request
     * @return Object
     */
    public function addQq(Request $request)
    {
        $qq = (string)$request->post('qq','');
        if(empty($qq)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
        $userId = $request->post('user_id');
        try {
            User::query()->where(['id' => $userId])->update(['qq' => $qq]);
            return ApiReturn::success();
```

```
        } catch (\Throwable $throwable) {
            Common::log('创作者添加qq号',$throwable);
            return ApiReturn::error();
        }
```

```
    }

}

```

```php
<?php

namespace App\Http\Controllers\api\v2;

use App\Http\Controllers\Controller;
use App\Http\Requests\GameRequest;
use App\Service\v1\ApiReturn;
use App\Service\v2\Article as V2Article;
use App\Service\v1\Code;
use App\Service\v1\Common;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class ArticleController extends Controller
{
    //发布动态
    public function publishDynamic(Request $request)
    {
        try {
            $param = $request->post();
            $param['action'] = $request->post('action', Code::ACTION_ADD);
            $param['type'] = Code::ARTICLE_TYPE_MAP['dynamic'];
            // 验证 设置
            Log::info('发布动态-入口参数-----' . json_encode($param, 256));
            $err = (array)V2Article::checkArticleParam($param);
            Log::info('发布动态-校验文章信息-----' . json_encode($err, 256));
            if ($err) return ApiReturn::rdata($err);
            // 插入到 article 表的数据
            $param['data'] = [
                'cover' => $param['cover'],
                'check_status' => Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
                'content' => $param['content'],
                // article 标题 截取
                'title' => $param['title'] ?? '',
                'text_content' => $param['text_content'],
                'content_len' => $param['content_len'],
                'home_page' => $param['home_page'],
                'label_list' => $param['label_list'],
                'abstract' => $param['abstract'],
                'user_id' => $param['user_id'],
            ];
            if (!empty($data['title'])) {
                if (!V2Article::checkContentBySensitiveWord($data['title'])) {
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
                }
            }
            // 敏感词检测-内容
            if (!empty($data['content'])) {
                if (!V2Article::checkContentBySensitiveWord($data['content'])) {
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
```

```
                }
            }
            if (!empty($data['text_content'])) {
                if (!V2Article::checkContentBySensitiveWord($data['text_content'])) {
                    $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
                }
            }
            if ($param['action'] == Code::ACTION_ADD) {
                // 新增需要的数据 (修改的时候不可更改的数据)
                $param['data'] = array_merge($param['data'], [
                    'game_id' => $param['game_id'],
                    'task_id' => $param['task_id'],
                    'circle_id' => $param['circle_id'],
                    'type' => $param['type'],
                    'is_writer_task' => $param['is_writer_task'],
                    'ip_location' => $param['ip_location'] ?? '',
                ]);
                if (isset($param['created_at'])) {
                    $param['data']['create_time'] = strtotime($param['created_at']);
                    $param['data']['update_time'] = strtotime($param['created_at']);
                }
            }
            Log::error('发布动态-插入数据-----' ,$param);
            $res = V2Article::addArticle($param);
            return $res;
        } catch (\Throwable $throwable) {
            Log::error('发布动态-插入数据-----' .$throwable->getMessage());
            return ApiReturn::error($throwable->getMessage());
        }
    }

}

```