

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Service\AppUpdate;
6  use App\Service\v1\ApiReturn;
7  use Illuminate\Http\Request;
8
9  class AppUpdateController extends Controller
10 {
11     /**
12      * ios更新
13      * @param Request $request
14      * @return Object
15      */
16     public function iosUpdate(Request $request)
17     {
18         $version = $request->get('version_code','');
19         $user_id = $request->get('user_id',0);
20         $device_num = $request->get('device_num','');
21         return AppUpdate::iosUpdate($version,$user_id,$device_num);
22     }
23     /**
24      * uni更新
25      * @param Request $request
26      * @return Object
27      */
28     public function uniUpdate(Request $request)
29     {
30         $version = $request->get('version','');
31         $channel = $request->get('channel','');
32         $user_id = $request->get('user_id',0);
33         $device_num = $request->get('device_num','');
34         return AppUpdate::uniUpdate($version,$channel,$user_id,$device_num);
35     }
36
37     //设置模式
38     public function setModel(Request $request){
39         $user_id = $request->get('user_id',0);
40         $device_num = $request->get('device_num','');
41         $status = $request->get('status');
42         if(!isset($status)){
43             return ApiReturn::error('请选择要设置的模式');
44         }
45         return AppUpdate::setModel($user_id,$device_num,$status);
46     }
47 }
48
```

```
1  <?php
2
3
4  namespace App\Http\Controllers;
5
```

```

6  use App\Models\v1\Article;
7  use App\Models\v1\ArticleCommentOne;
8  use App\Models\v1\ArticleCommentTwo;
9  use App\Service\Comment;
10 use App\Service\v1\ApiCode;
11 use App\Service\v1\ApiReturn;
12 use Illuminate\Http\Request;
13 use Illuminate\Support\Facades\Log;
14
15 class CommentController extends Controller
16 {
17
18     /**
19      * 删除评论
20      * @param Request $request
21      * @return Object
22      */
23     public function deleteComment(Request $request)
24     {
25         $userId = $request->post('user_id');
26         if (empty($userId)) {
27             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
28         }
29         $commentId = $request->post('comment_id', 0);
30         if (empty($commentId)) {
31             return ApiReturn::error('请选择要删除的评论');
32         }
33         return Comment::deleteComment($commentId, $userId);
34
35
36     }
37
38     /**
39      * 删除评论回复
40      * @param Request $request
41      * @return Object
42      */
43     public function deleteReply(Request $request)
44     {
45         $userId = $request->post('user_id');
46         if (empty($userId)) {
47             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
48         }
49         $commentReplyId = $request->post('comment_reply_id', 0);
50         if (empty($commentReplyId)) {


---


1         return ApiReturn::error('请选择要删除的评论');
2     }
3     return Comment::deleteReply($commentReplyId, $userId);
4 }
5
6
7 }
8

```

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Protocol;
6  use App\Service\Comment;
7  use App\Service\Common;
8  use App\Service\v1\ApiCode;
9  use App\Service\v1\ApiReturn;
10 use App\Service\v1\CommonFnc;
11 use Exception;
12 use Illuminate\Auth\Access\AuthorizationException;
13 use Illuminate\Http\Request;
14 use Illuminate\Support\Facades\Log;
15 use Illuminate\Validation\ValidationException;
16
17 class CommonController extends Controller
18 {
19     /**
20      * 首页轮播图
21      * @param Request $request
22      * @return Object
23      */
24     public function getBannerList(Request $request): object
25     {
26         try {
27             $redisKey = $request->get('type','banner-list');
28             $version = $request->get('version_code_num');
29
30             $list = Common::getNewBannerGameList($redisKey,$version);
31             return ApiReturn::success($list);
32         } catch (Exception $e) {
33             return ApiReturn::error();
34         }
35     }
36
37     /**
38      * 平台协议
39      * @param Request $request
40      * @return Object
41      */
42     public function getProtocolInfo(Request $request): object
43     {
44         $id = $request->get('type', 1);
45         if (!is_numeric($id) || $id < 1) {
46             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
47         }
48         try {
49             $info = Protocol::query()
50                 ->where('id', $id)
51
52                 ->where('status', 1)
53                 ->select('title', 'content', 'postscript', 'updated_at')
54                 ->first();
```

```
4         return ApiReturn::success($info);
5     } catch (\Exception $e) {
6         return ApiReturn::error();
7     }
8 }
9 //添加评论
10
11 /**
12  * 发布评论
13  * @param Request $request
14  * @return Object
15  */
16 public function addComment(Request $request): object
17 {
18     $user_id = $request->post('user_id');
19     if (empty($user_id)) {
20         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
21     }
22     $article_id = $request->post('article_id');
23     if (empty($article_id)) {
24         return ApiReturn::error('请选择要评论的文章');
25     }
26     $content = $request->post('content');
27     if (empty($content)) {
28         return ApiReturn::error('请输入要评论的内容');
29     }
30     $ip_location = $request->post('ip_location');
31     return Comment::addComment($user_id, $article_id, $content, $ip_location);
32 }
33
34
35 /**
36  * 发布评论回复
37  * @param Request $request
38  * @return Object
39  */
40 public function addReply(Request $request): object
41 {
42     $user_id = $request->post('user_id');
43     if (empty($user_id)) {
44         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
45     }
46     $article_id = $request->post('article_id');
47     if (empty($article_id)) {
48         return ApiReturn::error('请选择要评论的文章');
49     }
50     $reply_content = $request->post('reply_content');
51
52     if (empty($reply_content)) {
53         return ApiReturn::error('请输入要回复的内容');
54     }
55     $article_comment_one_id = $request->post('article_comment_one_id');
56     if (empty($article_comment_one_id)) {
57         return ApiReturn::error('请选择要回复的评论');
```

```

7      }
8      $by_evaluating_id = $request->post('by_evaluating_id');
9      $ip_location = $request->post('ip_location');
10     $type = $request->post('type');
11     if (empty($type)) {
12         return ApiReturn::error('请选择评论类型');
13     }
14     if(!in_array($type,[1,2])){
15         return ApiReturn::error('评论类型错误');
16     }
17     $data = [
18         'user_id' => $user_id,
19         'article_id' => $article_id,
20         'reply_content' => $reply_content,
21         'article_comment_one_id' => $article_comment_one_id,
22         'by_evaluating_id' => $by_evaluating_id,
23         'ip_location' => $ip_location,
24         'type' => $type,
25
26     ];
27     return Comment::addReply($data);
28 }
29 //PC游戏页面配置
30 public function getConfig(): object
31 {
32     try {
33         $lang = CommonFnc::lang();
34         $list = Common::getConfig($lang);
35         return ApiReturn::success($list);
36     } catch (Exception $e) {
37         return ApiReturn::error();
38     }
39 }
40 }
41

```

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
6  use Illuminate\Foundation\Bus\DispatchesJobs;
7  use Illuminate\Foundation\Validation\ValidatesRequests;
8  use Illuminate\Routing\Controller as BaseController;
9
10 class Controller extends BaseController
11 {
12     use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
13     //星际起源页面
14     public function AstralOrigin(){
15         return view('AstralOrigin');
16     }
17 }
18

```

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Mail\CodeSender;
6  use App\Service\Email;
7  use App\Service\v1\ApiCode;
8  use App\Service\v1\ApiReturn;
9  use App\Service\v1\CommonFnc;
10 use App\Service\v1\RedisKey;
11 use Illuminate\Http\Request;
12 use Illuminate\Support\Facades\Mail;
13 use Illuminate\Support\Facades\Redis;
14 use Illuminate\Support\Facades\Validator;
15
16 class EmailController extends Controller
17 {
18     //发送登录验证码
19     public function sendLoginCode(Request $request)
20     {
21         $lang = $request->header('Lang', 'en-CN');
22         $lang = CommonFnc::getLang($lang);
23         $email = $request->post('email');
24         $validator = Validator::make(['email' => $email], [
25             'email' => 'required|email',
26         ], [
27             'email.required' => '请输入邮箱',
28             'email.email' => '邮箱格式不正确'
29         ]);
30         if ($validator->fails()) {
31             // 获取验证失败的消息
32             $errors = $validator->errors();
33             return ApiReturn::error($errors->first());
34         }
35         $verifyCode = CommonFnc::randomNumeric();
36         if (Redis::exists(RedisKey::EMAIL_LOGIN_CODE . $email)) {
37             return ApiReturn::error('请勿重复发送');
38         }
39         try {
40             Redis::setex(RedisKey::EMAIL_LOGIN_CODE . $email, 600, $verifyCode);
41             Mail::to($email)->send(new CodeSender($verifyCode, $lang));
42             return ApiReturn::success('发送成功');
43         } catch (\Exception $e) {
44             return ApiReturn::error('发送失败' . $e->getMessage());
45         }
46     }
47
48 }
49 //邮箱登录
50 public function login(Request $request)
```

```
1  {
2      $email = $request->post('email');
3      $code = $request->post('code');
```

```

4      $platform = $request->post('platform', 0);
5      $isApp = $request->post('is_app', 1);
6      $validator = Validator::make(['email' => $email, 'code' => $code], [
7          'email' => 'required|email',
8          'code' => 'required|numeric|digits:6'
9      ], [
10         'email.required' => '请输入邮箱',
11         'email.email' => '邮箱格式不正确',
12         'code.required' => '请输入验证码',
13         'code.numeric' => '验证码格式不正确',
14         'code.digits' => '验证码格式不正确'
15     ]);
16     if ($validator->fails()) {
17         // 获取验证失败的消息
18         $errors = $validator->errors();
19         return ApiReturn::error($errors->first());
20     }
21     if (!Redis::exists(RedisKey::EMAIL_LOGIN_CODE . $email)) {
22         //验证码过期重新发送验证码
23         return ApiReturn::rdata(ApiCode::VERIFY_CODE_NIL);
24     }
25     //校验验证码是否正确
26     if ($code != Redis::get(RedisKey::EMAIL_LOGIN_CODE . $email)) {
27         return ApiReturn::rdata(ApiCode::VERIFY_CODE_ERROR);
28     }
29     $inviteCode = (string)$request->post('invite_code', '');
30     return Email::login($email, $platform, $isApp, $inviteCode);
31
32
33     }
34 }
35

```

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\v1\Label;
6  use App\Service\v1\ApiReturn;
7  use Illuminate\Contracts\Pagination\LengthAwarePaginator;
8  use Illuminate\Http\JsonResponse;
9  use Illuminate\Http\Request;
10
11  class LabelController extends Controller
12  {
13
14      /**
15       * 后台发布标签列表
16       * @return JsonResponse|Object
17       */
18       public function getPublishLabel()
19       {
20           $list = Label::query()
21               ->where('status', '=', 1)

```

```

22     ->where('type','=', 1)
23     ->where('is_hot', 1)
24     ->orWhere('recommend_type', 1)
25     ->select('id as label_id','label_name')
26     ->get()
27     ->toArray();
28     return ApiReturn::success(['data'=>$list]);
29 }
30
31 /**
32  * 后台评价标签列表
33  * @return JsonResponse|Object
34  */
35 public function getEvaluateLabel()
36 {
37     $list = Label::query()
38         ->where('user_id',0)
39         ->where('status','=',1)
40         ->where('type','=',2)
41         ->select('id','label_name as text')
42         ->get()
43         ->toArray();
44     return ApiReturn::success($list);
45 }
46 //添加帖子/评价标签
47 public function addArticleLabel(Request $request)
48 {
49     try {
50         $labelName = $request->post('label_name');

```

```

1         if (empty($labelName)) {
2             return ApiReturn::error('请输入标签名称');
3         }
4         //标签类型
5         $type = $request->post('type');
6         if (empty($type)) {
7             return ApiReturn::error('请选择标签类型');
8         }
9         if(!in_array($type,[1,2])){
10             return ApiReturn::error('请选择正确标签类型');
11         }
12         $userId = $request->post('user_id');
13         return Label::addArticleLabel($userId,$labelName,$type);
14     }catch (\Exception $e) {
15         return ApiReturn::error($e->getMessage());
16     }
17 }
18 }
19

```

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Service\Manager;

```



```

6  use App\Service\ReportAndBlock;
7  use App\Service\v1\ApiCode;
8  use App\Service\v1\ApiReturn;
9  use Illuminate\Http\Request;
10
11 class ManagerController extends Controller
12 {
13     public function menu(Request $request)
14     {
15         $userId = $request->get('user_id');
16         if (empty($userId)) {
17             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
18         }
19         return Manager::menu($userId);
20     }
21
22     //黑名单列表
23     public function blackList(Request $request)
24     {
25         $userId = $request->get('user_id');
26         if (empty($userId)) {
27             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
28         }
29         $page = $request->get('page', 1);
30         $limit = $request->get('limit', 10);
31         return Manager::blackList($userId, $page, $limit);
32     }
33
34     //权限列表
35     public function permissionList(Request $request)
36     {
37         $userId = $request->get('user_id');
38         if (empty($userId)) {
39             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
40         }
41         return Manager::permissionList($userId);
42     }
43
44     //拉黑用户
45     public function addUserBlock(Request $request): object
46     {
47         $user_id = $request->post('user_id');
48         if (empty($user_id)) {
49             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
50
51         }
52         $block_user_id = $request->post('block_user_id');
53         if (empty($block_user_id)) {
54             return ApiReturn::error('请选择要拉黑的用户');
55         }
56         if ($block_user_id == $user_id) {
57             return ApiReturn::error('您不能拉黑自己');
58         }
59     }

```

```

9      $remark = $request->post('remark');
10     return Manager::addUserBlock($user_id, $block_user_id, $remark);
11 }
12
13 //取消拉黑用户
14 public function cancelUserBlock(Request $request): object
15 {
16     $user_id = $request->post('user_id');
17     if (empty($user_id)) {
18         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
19     }
20     $block_user_id = $request->post('block_user_id');
21     if (empty($block_user_id)) {
22         return ApiReturn::error('请选择要取消封禁的用户');
23     }
24     $remark = $request->post('remark');
25     return Manager::cancelUserBlock($user_id, $block_user_id, $remark);
26 }
27
28 //操作记录
29 public function operationLog(Request $request)
30 {
31     $userId = $request->get('user_id');
32     if (empty($userId)) {
33         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
34     }
35     $page = $request->get('page', 1);
36     $limit = $request->get('limit', 10);
37     return Manager::operationLog($userId, $page, $limit);
38 }
39 //屏蔽文章
40 public function blockArticle(Request $request)
41 {
42     $user_id = $request->post('user_id');
43     if (empty($user_id)) {
44         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
45     }
46     $article_id = $request->post('article_id');
47     if (empty($article_id)) {
48         return ApiReturn::error('请选择要屏蔽的文章');
49     }
50     $remark = $request->post('remark');
51
52     return Manager::blockArticle($user_id, $article_id, $remark);
53 }
54
55 }
56

```

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Service\ReportAndBlock;

```

```

6  use App\Service\v1\ApiCode;
7  use App\Service\v1\ApiReturn;
8  use Illuminate\Http\Request;
9  use Illuminate\Support\Facades\Log;
10
11 class ReportAndBlockController extends Controller
12 {
13     //举报选项
14     public function reportOptions()
15     {
16         return ReportAndBlock::reportOptions();
17     }
18
19     /**
20      * 举报文章
21      * @param Request $request
22      * @return Object
23      */
24     public function addArticleReport(Request $request): object
25     {
26         $user_id = $request->post('user_id');
27         if (empty($user_id)) {
28             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
29         }
30         $article_id = $request->post('article_id');
31         if (empty($article_id)) {
32             return ApiReturn::error('请选择要举报的文章');
33         }
34         $reason = $request->post('reason');
35         if (empty($reason)) {
36             return ApiReturn::error('请选择举报理由');
37         }
38         $device_num = $request->post('device_num', '');
39         return ReportAndBlock::addArticleReport($user_id, $article_id, $reason, $device_num);
40     }
41
42     //拉黑用户
43     public function addUserBlock(Request $request): object
44     {
45         $user_id = $request->post('user_id');
46         if (empty($user_id)) {
47             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
48         }
49         $block_user_id = $request->post('block_user_id');
50         if (empty($block_user_id)) {


---


1         return ApiReturn::error('请选择要拉黑的用户');
2     }
3     if ($block_user_id == $user_id) {
4         return ApiReturn::error('您不能拉黑自己');
5     }
6     return ReportAndBlock::addUserBlock($user_id, $block_user_id);
7 }
8

```

```
9      //拉黑游戏
10     public function addGameBlock(Request $request): object
11     {
12         $user_id = $request->post('user_id');
13         if (empty($user_id)) {
14             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
15         }
16         $block_game_id = $request->post('block_game_id');
17         if (empty($block_game_id)) {
18             return ApiReturn::error('请选择要拉黑的游戏');
19         }
20         return ReportAndBlock::addGameBlock($user_id, $block_game_id);
21     }
22
23     //用户黑名单列表
24     public function userBlockList(Request $request): object
25     {
26         $user_id = $request->post('user_id');
27         if (empty($user_id)) {
28             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
29         }
30         $page = $request->post('page', 1);
31         $limit = $request->post('limit', 10);
32         return ReportAndBlock::userBlockList($user_id, $page, $limit);
33     }
34
35     //取消拉黑用户
36     public function cancelUserBlock(Request $request): object
37     {
38         $user_id = $request->post('user_id');
39         if (empty($user_id)) {
40             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
41         }
42         $block_user_id = $request->post('block_user_id');
43         if (empty($block_user_id)) {
44             return ApiReturn::error('请选择要取消拉黑的用户');
45         }
46         return ReportAndBlock::cancelUserBlock($user_id, $block_user_id);
47     }
48
49     //举报评论
50     public function addCommentReport(Request $request)
51


---


1     {
2         Log::info('举报评论入参', $request->all());
3         $user_id = $request->post('user_id');
4         if (empty($user_id)) {
5             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
6         }
7         $comment_id = $request->post('comment_id');
8         if (empty($comment_id)) {
9             return ApiReturn::error('请选择要举报的评论');
10        }
11        $reason = $request->post('reason');
```

```

12     if (empty($reason)) {
13         return ApiReturn::error('请选择举报理由');
14     }
15     $device_num = $request->post('device_num', '');
16     return ReportAndBlock::addCommentReport($user_id, $comment_id, $reason, $device_num);
17 }
18
19 //举报评论
20 public function addCommentReplyReport(Request $request)
21 {
22     Log::info('举报评论回复入参', $request->all());
23     $user_id = $request->post('user_id');
24     if (empty($user_id)) {
25         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
26     }
27     $comment_reply_id = $request->post('comment_reply_id');
28     if (empty($comment_reply_id)) {
29         return ApiReturn::error('请选择要举报的评论');
30     }
31     $reason = $request->post('reason');
32     if (empty($reason)) {
33         return ApiReturn::error('请选择举报理由');
34     }
35     $device_num = $request->post('device_num', '');
36     return ReportAndBlock::addCommentReplyReport($user_id, $comment_reply_id, $reason, $device_num);
37 }
38
39
40 }
41

```

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Service\ArticleService;
6  use App\Service\v1\ApiReturn;
7  use App\Service\v1\Article as V1Article;
8  use App\Service\v1\Code;
9  use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Log;
11
12 class ReviewController extends Controller
13 {
14     //发表评价
15     public function add(Request $request)
16     {
17         Log::info('添加测评传参', $request->all());
18         $game_id = $request->post('game_id');
19         if(empty($game_id)){
20             return ApiReturn::error('请选择要评价的游戏');
21         }
22         $content = $request->post('content');
23         if(empty($content)){

```

```

24         return ApiReturn::error('请输入评价内容');
25     }
26     $score = $request->post('score');
27     if(empty($score)){
28         return ApiReturn::error('请输入评价分数');
29     }
30     $images = $request->post('images',[]);
31     $labels = $request->post('labels');
32     $user_id = $request->post('user_id');
33     $ip_location = $request->post('ip_location');
34     return ArticleService::addReview($user_id,$game_id,$content,$score,$images,$labels,$ip_location);
35 }
36 }
37

```

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Service\Steam;
6  use App\Service\v1\ApiCode;
7  use App\Service\v1\ApiReturn;
8  use App\Service\v1\Code;
9  use App\Service\v1\CommonFnc;
10 use Illuminate\Http\Request;
11 use Illuminate\Support\Str;
12
13 class SteamController extends Controller
14 {
15     //绑定Steam账号
16     public function bindAccount(Request $request)
17     {
18         $userId = $request->post('user_id');
19         if(empty($userId)){
20             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
21         }
22         $steamid = $request->input('steamid');
23         if(empty($steamid)){
24             return ApiReturn::error('请选择要绑定的Steam账号');
25         }
26         return Steam::bindAccount($userId,$steamid);
27     }
28     //解绑Steam账号
29     public function unbindAccount(Request $request)
30     {
31         $userId = $request->post('user_id');
32         if(empty($userId)){
33             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
34         }
35         return Steam::unbindAccount($userId);
36     }
37
38     //获取Steam用户信息
39     public function getUserInfo(Request $request)

```

```

40     {
41         $solo_userid = $request->post('solo_userid');
42         if(empty($solo_userid)){
43             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
44         }
45         $userId = $request->post('user_id');
46         return Steam::getUserInfo($solo_userid,$userId);
47     }
48
49     //获取Steam好友列表
50     public function getFriendList(Request $request)
51
52     {
53         $soloUserId = $request->get('solo_userid');
54         if(empty($soloUserId)){
55             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
56         }
57         $page = $request->input('page',1);
58         $limit = $request->input('limit',8);
59         $username = $request->input('username','');
60         $user_id = $request->get('user_id');
61         return Steam::getFriendList($soloUserId,$user_id,$username,$page,$limit);
62     }
63
64     //获取 Steam 用户游戏列表
65     public function getUserGames(Request $request)
66     {
67         $soloUserId = $request->get('solo_userid');
68         if(empty($soloUserId)){
69             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
70         }
71         $page = $request->input('page',1);
72         $limit = $request->input('limit',8);
73         $order = $request->input('order','id:desc');
74         $user_id = $request->get('user_id');
75         $lang = CommonFnc::lang();
76         return Steam::getUserGames($soloUserId,$user_id,$lang,$page,$limit,$order);
77     }
78
79     //同步steam账号信息
80     public function syncUserInfo(Request $request)
81     {
82         $userId = $request->post('user_id');
83         if(empty($userId)){
84             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
85         }
86         $syncKey = $request->post('sync_key');
87         if(empty($syncKey)){
88             $syncKey = Str::uuid();
89         }
90         return Steam::syncUserInfo($userId,$syncKey);
91     }
92
93     public function getUserSyncProgress(Request $request)
94     {
95         $userId = $request->post('user_id');
96         if(empty($userId)){

```

```

43         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
44     }
45     $syncKey = $request->post('sync_key');
46     return Steam::getUserSyncProgress($userId,$syncKey);
47 }
48 //获取Steam用户隐私设置
49 public function getUserPrivacy(Request $request)
50 {


---


1     $userId = $request->get('user_id');
2     if(empty($userId)){
3         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
4     }
5
6     return Steam::getPrivacySettings($userId);
7 }
8 //设置Steam用户隐私设置
9 public function setUserPrivacy(Request $request)
10 {
11     $userId = $request->get('user_id');
12     if(empty($userId)){
13         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
14     }
15     $privacy = $request->post('privacy');
16     if(empty($privacy)){
17         return ApiReturn::error('请选择隐私设置');
18     }
19     return Steam::setPrivacySettings($userId,$privacy);
20 }
21
22 //添加到库
23 public function addToLibrary(Request $request)
24 {
25     $userId = $request->post('user_id');
26     if(empty($userId)){
27         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
28     }
29     $game_id = $request->post('game_id');
30     if(empty($game_id)){
31         return ApiReturn::error('请选择要添加的Steam游戏');
32     }
33     return Steam::addToLibrary($userId,$game_id);
34 }
35
36 }
37

```

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Service\v1\ApiCode;
6 use App\Service\v1\ApiReturn;
7 use App\Service\v1\User;
8 use Illuminate\Http\Request;

```



```

9  use Illuminate\Support\Facades\Redis;
10
11  class UserAttentionController extends Controller
12  {
13      /**
14       * 关注作者
15       * @param Request $request
16       * @return Object
17       * @throws \Throwable
18       */
19      public function addAttentionUser(Request $request): object
20      {
21          try {
22              $authorId = (int)$request->post('author_id',0);
23              if(empty($authorId)){
24                  return ApiReturn::rdata(ApiCode::PARAM_ERROR);
25              }
26              $userId = $request->post('user_id');
27              // if(Redis::exists('user:attention:user:'.$userId)){
28              //     return ApiReturn::error('操作太频繁');
29              // }
30              // Redis::set('user:attention:user:'.$userId,1,'PX',300,'NX');
31              return User::addAttentionUser($userId,$authorId);
32          } catch (\Exception $e) {
33              return ApiReturn::error();
34          }
35      }
36  }
37  }
38

```

```

1  <?php
2
3
4  namespace App\Http\Controllers\api;
5
6  use App\Events\JytFirstPublishTaskEvent;
7  use App\Http\Controllers\Controller;
8  use App\Models\v1\Article;
9  use App\Models\v1\Circle;
10 use App\Models\v1\CollectPlatformUrl;
11 use App\Models\v1\FanBase;
12 use App\Models\v1\Game;
13 use App\Models\v1\LongArticle;
14 use App\Models\v1\User;
15 use App\Models\v1\UserAttention;
16 use App\Models\v1\UserDiamondLog;
17 use App\Models\v1\UserFanBase;
18 use App\Models\v1\UserLike;
19 use App\Models\v1\UserRelevance;
20 use App\Service\v1\AliContentAudit;
21 use App\Service\v1\ApiCode;
22 use App\Service\v1\ApiReturn;
23 use App\Service\v1\BaiDuSdk;

```

```
24 use App\Service\v1\Common;
25 use App\Service\v1\CommonFnc;
26 use App\Service\v1\ElasticsearchService;
27 use App\Service\v1\RedisKey;
28 use App\Service\v1\SoloCode;
29 use App\Service\v1\WechatPayService;
30 use Illuminate\Http\Request;
31 use Illuminate\Support\Facades\DB;
32 use Illuminate\Support\Facades\Log;
33 use stdClass;
34 use Swoole\Coroutine\Redis;
35
36 class IndexController extends Controller
37 {
38
39     public function index(Request $request)
40     {
41         try {
42             $re = LongArticle::where(['status' => 1])->chunk(300,function ($articles) {
43                 foreach ($articles as $article) {
44                     if($article->type == 3) {
45                         $arr = json_decode($article->content,true);
46                         if(empty($arr)) continue;
47                         $str = '';
48                         foreach ($arr as $v) {
49                             if($v['type'] == 'text') $str .= strip_tags($v['character_string']);
50                         }
51
52                         $article->text_content = $str;
53                     }else{
54                         $article->text_content = $article->content;
55                     }
56                     $article->save();
57                 }
58             });
59             return ApiReturn::success();
60         }catch (\Throwable $throwable) {
61             return ApiReturn::error($throwable->getMessage());
62         }
63
64         // $content = $request->get('content');
65         // $result = AliContentAudit::textAudit([$content,'草泥马']);
66         // if($result) return ApiReturn::success();
67         // return ApiReturn::error(ApiCode::ALI_CONTENT_AUDIT['msg'],ApiCode::ALI_CONTENT_AUDIT['code']);
68         // $re = \Illuminate\Support\Facades\Redis::get('banner:list');dd($re);
69         // $redis = new Redis();
70         // $redis->connect(config('redis'))
71         // $userId = Redis::get('amway:walls');print_r($userId);
72     }
73
74     public function test(Request $request)
75     {
76         $ip = CommonFnc::getRealIP();
```

```

1 //      ->groupBy('parent_user_id')->get();
2 //      foreach ($userRelevances as $relevance) {
3 //          if(isset($relevance->mcn_config->id)) {
4 //              User\UserMcnConfig::query()->where(['id' => $relevance->mcn_config->id])->update(['num' =>
5 //                  $relevance->num]);
6 //          }
7 //      }
8
9      //老用户默认分配一个二维码
10     $userRelevances = UserRelevance::query()->with(['mcn_config','user'])
11     //      ->where(['check_status' => 1])
12     //      ->where('user_mcn_config_id','>',0)
13     //      ->get();
14     foreach ($userRelevances as $relevance) {
15     //      if(isset($relevance->mcn_config->id) && isset($relevance->user->id) && $relevance->user->user_mcn_config_id
16     //      == 0) {
17     //          $relevance->user()->update(['user_mcn_config_id' => $relevance->mcn_config->id]);
18     //          $relevance->user_mcn_config_id = $relevance->mcn_config->id;
19     //          $relevance->save();
20     //      }
21
22     //      $userRelevances = UserRelevance::query()->with(['user'])
23     //      ->where(['check_status' => 1])
24     //      ->where('user_mcn_config_id','>',0)
25     //      ->get();
26     foreach ($userRelevances as $relevance) {

```

```

27 //      $relevance->user()->update(['user_mcn_config_id' => $relevance->parent_user_id]);
28 //      $relevance->user_mcn_config_id = $relevance->parent_user_id;
29 //      $relevance->save();
30 //  }
31 //  $list = DB::table('user_mcn_configs_copy1')->get();
32 //  $data = [];
33 //  foreach ($list as $v) {
34 //      $data[] = [
35 //          'id' => $v->user_id,
36 //          'user_id' => $v->user_id,
37 //          'functional_type' => $v->functional_type,
38 //          'commission' => $v->commission,
39 //          'first_publish_commission' => $v->first_publish_commission,
40 //          'qq_group' => $v->qq_group,
41 //          'desc' => $v->desc,
42 //          'tips' => $v->tips,
43 //          'num' => $v->num,
44 //          'name' => $v->name,
45 //          'created_at' => $v->created_at,
46 //          'updated_at' => $v->updated_at,
47 //      ];
48 //  }
49 //  User\UserMcNConfig::query()->insert($data);
50 //  $userMcNConfigs = User\UserMcNConfig::query()->get();

```

```

1 //  foreach ($userMcNConfigs as $userMcNConfig) {
2 //      $userMcNConfig->id = $userMcNConfig->user_id;
3 //      $userMcNConfig->save();
4 //  }
5
6 //  $images = $request->post('images');
7 //  $result = AliContentAudit::imageAudit($images);
8 //  switch ($result) {
9 //      case SoloCode::ALI_AUDIT_PASS :
10 //          $msg = '正常';
11 //          break;
12 //      case SoloCode::ALI_AUDIT_REVIEW :
13 //          $msg = '需要人工审核';
14 //          break;
15 //      case SoloCode::ALI_AUDIT_BLOCK :
16 //          $msg = '违规';
17 //          break;
18 //      case SoloCode::ALI_AUDIT_ERROR :
19 //          $msg = '错误';
20 //          break;
21 //  }
22 //  return ApiReturn::error($msg);
23 //  $circles = Circle::where('id','<',20)->select('id','circle_name','avatar')->get()->toArray();
24 //  $es = (new ElasticsearchService('circles'))->paginate(1,20);
25 //  $es->source(['id','circle_name','avatar','cover','articles_num','attention_num']);
26 //  $circleList = app('es')->search($es->getParams());
27 //  $circles = array_column($circleList['hits']['_source']);
28 //  $param = [
29 //      'body' => []

```

```

30 // ];
31 // foreach ($circles as $circle) {
32 //     array_push($param['body'], ['index' => 'articles'],
33 //     [
34 //         'query' => [
35 //             'term' => ['circle_id' => $circle['id']]
36 //         ],
37 //         '_source' => ['id', 'title', 'circle_id'],
38 //         'from' => 0,
39 //         'size' => 2,
40 //     ]);
41 // }
42 // $response = app('es')->msearch($param);
43 // foreach ($circles as &$circle) {
44 //     $circle['avatar'] = empty($circle['avatar']) ? '' : Common::concatUrlStr($circle['avatar']);
45 //     $circle['cover'] = empty($circle['cover']) ? '' : Common::concatUrlStr($circle['cover']);
46 //     $circle['articles_num'] = Common::disposeAmount($circle['articles_num']);
47 //     $circle['attention_num'] = Common::disposeAmount($circle['attention_num']);
48 //     $circle['article'] = [];
49 //     foreach ($response['responses'] as $v) {
50 //         foreach ($v['hits']['hits'] as $h) {


---


1 //         if($circle['id'] == $h['_source']['circle_id']) {
2 //             $circle['article'][] = $h['_source'];
3 //         }
4 //     }
5 // }
6 // }
7 // return ApiReturn::success($circles);
8 // $jia = UserDiamondLog::where(['type' => 7])->selectRaw("user_id,SUM(diamond) as jia")->groupBy('user_id')->get()->toArray();
9 // $jian = UserDiamondLog::whereIn('type',[5,6])->selectRaw("user_id,SUM(diamond) as jian")->groupBy('user_id')->get()->toArray();
10 // try {
11 //     DB::beginTransaction();
12 //     foreach ($jia as $v) {
13 //         User::where(['user_id' => $v['user_id']])->increment($v['jia']);
14 //     }
15 //     foreach ($jian as $v) {
16 //         User::where(['user_id' => $v['user_id']])->decrement($v['jian']);
17 //     }
18 //     UserDiamondLog::whereIn('type',[5,6,7])->delete();
19 //     DB::commit();
20 // }catch (\Throwable $throwable) {
21 //     DB::rollBack();
22 //     return ApiReturn::error($throwable->getMessage());
23 // }
24
25
26 //整理签到reids数据
27 // $redisDayKeys = \Illuminate\Support\Facades\Redis::keys(RedisKey::SYSTEM_TASK_USER_SIGN_DAY.*');
28 // $redisTimeKeys = \Illuminate\Support\Facades\Redis::keys(RedisKey::SYSTEM_TASK_USER_SIGN_TIME.*');
29 // $tomorrow = strtotime('tomorrow');
30 // $afterTomorrow = strtotime('+ 2 days midnight');

```

```

31 // \Illuminate\Support\Facades\Redis::pipeline(function ($pipe)
    use($redisTimeKeys,$redisDayKeys,$tomorrow,$afterTomorrow) {
32 //     foreach ($redisTimeKeys as $redisTimeKey) {
33 //         $arr = explode(':', $redisTimeKey);
34 //         $userId = array_pop($arr);
35 //         $pipe->expireat(RedisKey::SYSTEM_TASK_USER_SIGN_TIME.$userId,$tomorrow);
36 //         $pipe->incr(RedisKey::SYSTEM_TASK_USER_SIGN_DAY.$userId);
37 //     }
38 //
39 //     foreach ($redisDayKeys as $redisDayKey) {
40 //         $arr = explode(':', $redisDayKey);
41 //         $userId = array_pop($arr);
42 //         $pipe->expireat(RedisKey::SYSTEM_TASK_USER_SIGN_DAY.$userId,$afterTomorrow);
43 //     }
44 // });
45
46 //达人开粉丝群
47 // $users = User::where(['attestation_type' => 2])->select('id','username','avatar')->get();
48 // $fanBaseUserId = FanBase::pluck('user_id')->toArray();
49 // $fanBaseData = $userIds = [];
50 // $time = date('Y-m-d H:i:s');

```

```

1 // $userFanBaseTime = time();
2 // foreach ($users as $user) {
3 //     if(!in_array($user->id,$fanBaseUserId)) {
4 //         $fanBaseData[] = [
5 //             'name' => $user->username."粉丝群",
6 //             'icon' => $user->avatar,
7 //             'user_id' => $user->id,
8 //             'user_count' => 1,
9 //             'created_at' => $time,
10 //             'updated_at' => $time
11 //         ];
12 //         $userIds[] = $user->id;
13 //     }
14 // }
15 // if(!empty($fanBaseData)) {
16 //     FanBase::insert($fanBaseData);
17 //     $userFanBaseData = FanBase::whereIn('user_id',$userIds)->select('user_id','id as fan_base_id')->get()->toArray();
18 //     foreach ($userFanBaseData as &$u) {
19 //         $u['is_master'] = 1;
20 //         $u['created_at'] = $userFanBaseTime;
21 //         $u['updated_at'] = $userFanBaseTime;
22 //     }
23 //     UserFanBase::insert($userFanBaseData);
24 // }
25
26 //点赞数据整理
27 // $userLikes = UserLike::where(['status' => 1])->groupBy('relevance_id')->selectRaw("relevance_id,COUNT(*) as
    total")->get();
28 // foreach($userLikes as $userLike) {
29 //     Article::where(['id' => $userLike->relevance_id])->update(['like_num' => $userLike->total]);
30 // }
31 //dump(11);

```

```

32 // $likes = Article::groupBy('user_id')->selectRaw("user_id,COUNT(*) as total")->get();
33 // foreach($likes as $like) {
34 //     User::where(['id' => $like->user_id])->update(['like_num' => $like->total]);
35 // }
36 // for($i = 0; $i < 100000; $i++) {
37 //     $s1 = '真实体验：萌新开路';
38 //     $s2 = '真实体验：萌新开';
39 //     $re = similar_text($s1,$s2,$end);
40 // }
41
42 // dump($re);
43 // dump($end);
44 // dump(levenshtein($s1,$s2));
45
46 // $re = UserAttention::where(['is_attention' =>
47 //     1])->groupBy('be_focused_user_id')->selectRaw("be_focused_user_id,count(*) as num")->get()->toArray();
48 // foreach ($re as $v) {
49 //     User::where(['id' => $v['be_focused_user_id']])->update(['fans_num' => $v['num']]);
50 // }
51 // return ApiReturn::success($re);

```

```

1 // $articleId = (int)$request->get('id',0);
2 // $num = (int)$request->get('num',1);
3 // $re = \App\Service\v1\Article::editArticleVariation($articleId,RedisKey::LIKE_NUM,$num);
4 // $re = \App\Service\v1\Article::articleVariations($articleId);
5 // return ApiReturn::success($re);
6 //游戏名称添加到搜索词库
7 // $re = Game::pluck('game_name')->toArray();
8 // $data = [];
9 // $time = time();
10 // foreach ($re as $v) {
11 //     if(!empty($v)) {
12 //         $data[] = [
13 //             'keyword' => $v,
14 //             'created_at' => $time,
15 //             'updated_at' => $time,
16 //         ];
17 //     }
18 // }
19 // DB::table('search_keywords')->insert($data);
20 // return ApiReturn::success($data);
21 }
22
23 /**
24  * 内部测试
25  */
26 public function ctest(Request $request)
27 {
28     $type = $request->get('type');
29     if ($type == 'pay_test') {
30         $data = new stdClass();
31         $data->order_num = 'test01';
32         $data->total_price = 0.01;
33         $rs = WechatPayService::appCreateOrder($data);

```

```

34         dd($rs);
35     }
36     if ($type == 'test') {
37         return ApiReturn::rdata('paramError');
38     }
39     if ($type == 'paytest') {
40
41         $option = [
42             "appid" => "wx9b84cce916bbca85",
43             "out_batch_no" => "ts01",
44             "batch_name" => "测试",
45             "batch_remark" => "测试",
46             "total_amount" => 1,
47             "total_num" => 1,
48             "transfer_detail_list" => [
49                 [
50                     "out_detail_no" => "ts01",
51
52                     "transfer_amount" => 1,
53                     "transfer_remark" => "测试",
54                     "openid" => "oaObN6m9GI7nshnyqJkGeelAaskU",
55                     // "user_name" => "小那"
56                 ]
57             ];
58         $rs = WechatPayService::payToBalace($option);
59         Log::info('paytest', $rs);
60     } else if ($type == 'jftp') {
61         JytFirstPublishTaskEvent::dispatch(41435);
62     }
63 }
64

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\v1\ServiceConfig;
7  use App\Models\v1\TaskTab;
8  use App\Models\v1\User;
9  use App\Service\v1\ActivityCenter;
10 use App\Service\v1\ApiReturn;
11 use App\Service\v1\Code;
12 use App\Service\v1\Common;
13 use App\Service\v1\Task;
14 use Illuminate\Http\Request;
15
16 class ActivityCenterController extends Controller
17 {
18
19     /**
20      * 活动中心筛选页面
21      * @return Object
22      */

```



```
23     public function filtrate()
24     {
25         try {
26             $data = ActivityCenter::filtrate();
27             return ApiReturn::success($data);
28         } catch (\Throwable $throwable) {
29             Common::log('活动中心筛选页面错误',$throwable);
30             return ApiReturn::error();
31         }
32     }
33
34     /**
35      * 活动中心任务列表
36      * @param Request $request
37      * @return Object
38      */
39     public function getTaskList(Request $request)
40     {
41         $userId = $request->get('user_id');
42         // $taskTabId = $request->get('task_tab_id','');
43         $taskTabId = 16;
44         $name = (string)$request->get('name','');
45         $limit = (int)$request->get('limit',10);
46         $page = (int)$request->get('page',1);
47         $taskStatus = (int)$request->get('task_status',0);
48         $platformId = $request->get('platform_id','');
49         $gameTypeId = $request->get('game_type_id','');
50         $sort = (int)$request->get('sort',0);
51
52         // 用户角色调取
53         $userRoles = [Code::ROLE_TYPE_MAP['default']]; // 普通权限
54         $page = ($page - 1) * $limit;
55         try {
56             $time = $request->server('REQUEST_TIME');
57             $model = \App\Models\v1\Task::query();
58             // if($taskTabId > 0) $model->where(['task_tab_id' => $taskTabId]);
59             // if(!empty($articleType)) $model->whereIn('type',explode(',',$articleType));
60             $model->where('task_status', 1)->where('participation_type', '<>', 1); // 状态为显示 去除写手活动
61             $model->orderBy('task_expired', 'asc');
62             if($sort > 0) {
63                 $sort = $sort == 1 ? 'asc' : 'desc';
64                 $model->orderBy('end_time',$sort);
65             }
66             // 写手活动筛选
67             if ($userId > 0) {
68                 $userInfo = User::find($userId);
69                 $roles = $userInfo->user_role()->pluck('user_role_id')->toArray();
70                 if ($roles) $userRoles = $roles;
71                 $model->where(function ($q) use ($userId) {
72                     $q->where('writer_id', $userId)->orWhere('writer_id', 0);
73                 });
74                 if($taskStatus < 3) {
75                     $createTime = strtotime($userInfo->create_time);
76                     $model->where('start_exp','<=', $userInfo->experience)
```

```
26         ->where('end_exp','>='.$userInfo->experience)
27         ->where('register_start_time','<='.$createTime)
28         ->where('register_end_time','>='.$createTime);
29     }
30 }else{
31     $model->where(['is_visitor' => 1,'writer_id' => 0]);
32 }
33 if(!empty($taskTabId)) {
34     $model->whereIn('task_tab_id', explode(',',$taskTabId));
35 }
36 if(!empty($platformId)) {
37     $model->whereHas('platform',function ($query) use ($platformId) {
38         $query->whereIn('activity_center_platform_id',explode(',',$platformId));
39     });
40 }
41 if(!empty($gameTypeId)) {
42     $model->whereHas('game_type',function ($query) use ($gameTypeId) {
43         $query->whereIn('activity_center_game_type_id',explode(',',$gameTypeId));
44     });
45 }
46 if(!empty($name)) $model->where('name','like','%{$name}%');
47 $model->with('platform','game');
48 switch ($taskStatus) {
49     case 0:
50         $model->where('start_time','<',$time);
51
52 1         $taskEndTime = (int)ServiceConfig::where(['field_name' => 'task_end_time']->value('value');
53 2         if($taskEndTime > 0) $model->where('end_time','>',$time - $taskEndTime * 24 * 3600);
54 3         $model->where(function ($model) use ($userRoles,$userId) {
55 4             $model->where(function ($model) use ($userRoles) {
56 5                 $model->where(['role_type' => 1])>whereHas('user_role', function ($q) use ($userRoles) {
57 6                     $q->whereIn('user_role_id', $userRoles);
58 7                 });
59 8             })>orWhere(function ($model) use ($userRoles) {
60 9                 $model->where(['role_type' => 2])>whereDoesntHave('not_user_role', function ($q) use ($userRoles) {
61 10                     $q->whereIn('user_role_id', $userRoles);
62 11                 });
63 12             })>orWhere(['role_type' => 0])>orWhere(function ($model) use ($userId) {
64 13                 $model->where(['role_type' => 3])>whereHas('task_user', function ($q) use ($userId) {
65 14                     $q->where(['user_id' => $userId]);
66 15                 });
67 16             });
68 17         });
69 18         $model->orderBy('is_top','desc')
70 19         ->orderBy('is_hot','desc')
71 20         ->orderBy('end_time','desc')
72 21         ->orderBy('id','desc');
73 22
74 23         $list = Task::getTaskList($model,$userId,$page,$limit);
75 24         break;
76 25     case 1:
77 26         //         $model->whereHas('user_role', function ($q) use ($userRoles) {
78 27             //             $q->whereIn('user_role_id', $userRoles);
79 28             //         });
```

```

29     $model->where(function ($model) use ($userRoles,$userId) {
30         $model->where(function ($model) use ($userRoles) {
31             $model->where(['role_type' => 1])->whereHas('user_role', function ($q) use ($userRoles) {
32                 $q->whereIn('user_role_id', $userRoles);
33             });
34         })->orWhere(function ($model) use ($userRoles) {
35             $model->where(['role_type' => 2])->whereDoesntHave('not_user_role', function ($q) use ($userRoles) {
36                 $q->whereIn('user_role_id', $userRoles);
37             });
38         })->orWhere(['role_type' => 0])->orWhere(function ($model) use ($userId) {
39             $model->where(['role_type' => 3])->whereHas('task_user', function ($q) use ($userId) {
40                 $q->where(['user_id' => $userId]);
41             });
42         });
43     });
44     $model->orderBy('is_top','desc')
45     ->orderBy('is_hot','desc')
46     ->orderBy('id','desc');
47     $model->where('start_time','<',$time)
48     ->where('end_time','>',$time)
49     ->whereColumn("astrict_participation_num",">","participation_num");
50     $list = Task::getTaskList($model,$userId,$page,$limit);

```

```

1         break;
2     case 2:
3         //     $model->whereHas('user_role', function ($q) use ($userRoles) {
4         //         $q->whereIn('user_role_id', $userRoles);
5         //     });
6         $model->where(function ($model) use ($userRoles,$userId) {
7             $model->where(function ($model) use ($userRoles) {
8                 $model->where(['role_type' => 1])->whereHas('user_role', function ($q) use ($userRoles) {
9                     $q->whereIn('user_role_id', $userRoles);
10                });
11            })->orWhere(function ($model) use ($userRoles) {
12                $model->where(['role_type' => 2])->whereDoesntHave('not_user_role', function ($q) use ($userRoles) {
13                    $q->whereIn('user_role_id', $userRoles);
14                });
15            })->orWhere(['role_type' => 0])->orWhere(function ($model) use ($userId) {
16                $model->where(['role_type' => 3])->whereHas('task_user', function ($q) use ($userId) {
17                    $q->where(['user_id' => $userId]);
18                });
19            });
20        });
21        $model->orderBy('is_top','desc')
22        ->orderBy('is_hot','desc')
23        ->orderBy('id','desc');
24        $list = Task::getTaskListEnd($model,$userId,$page,$limit);
25        break;
26    case 3:
27        if ($userId == 0) return ApiReturn::success([]); // 已参与
28        $model->where('start_time','<',$time);
29        $model->orderBy('is_top','desc')
30        ->orderBy('is_hot','desc');
31

```

```

32         $list = Task::getTaskListParticipated($model,$userId,$page,$limit);
33         break;
34     }
35     // 详情 h5 链接
36     $h5 = env('WEB_H5_URL');
37     foreach ($list as &$v) {
38         $v['h5_url'] = rtrim($h5, '/') . '/taskdetail/' . $v['id'];
39     }
40     return ApiReturn::success($list);
41 } catch (\Throwable $throwable) {
42     Common::log('活动中心任务列表错误',$throwable);
43     return ApiReturn::error();
44 }
45 }
46
47
48 /**
49  * 活动中心tab列表
50  * @return Object

```

```

1  */
2  public function getTaskTabList()
3  {
4      try {
5          $taskTabs = TaskTab::select('id','name')->whereIn('type', [0,5])->orderBy('sort')->get()->toArray();
6          array_unshift($taskTabs,['id' => 0,'name' => '全部']);
7          return ApiReturn::success($taskTabs);
8      } catch (\Throwable $throwable) {
9          Common::log('活动中心tab列表错误',$throwable);
10         return ApiReturn::error();
11     }
12
13 }
14
15
16 /**
17  * 邀请页活动展示
18  * @param Request $request
19  * @return Object
20  */
21 public function getInviteTaskList(Request $request)
22 {
23     $tab = (int)$request->get('tab',0); // 0 默认 返利活动 1 鉴游团活动
24     $limit = (int)$request->get('limit',50);
25     $page = (int)$request->get('page',1);
26     $page = ($page - 1) * $limit;
27     // 用户角色调取
28     try {
29         $time = $request->server('REQUEST_TIME'); // 脚本开始执行时间
30         $model = \App\Models\v1\Task::query();
31         $model->where('task_status', 1); // 状态为显示
32         $model->orderBy('end_time', 'desc');
33         // 写手活动筛选
34         $model->where(['writer_id' => 0]);

```

```

35     $model->with('platform', 'game');
36     if ($tab == 1) {
37         $model->where('invite_reward_switch', 0);
38         $model->where(['role_type' => 1]);
39         $model->whereHas('jyt_user_role');
40     } else $model->where('invite_reward_switch', 1);
41     $model->where('start_time','<',$time)
42         ->where('end_time','>',$time);
43     $model->orderBy('is_top','desc')
44         ->orderBy('is_hot','desc')
45         ->orderBy('id','desc');
46     $model->where('start_time','<',$time)
47         ->where('end_time','>',$time)
48         ->whereColumn("astrict_participation_num",">","participation_num");
49     $list = Task::getTaskList($model,0,$page,$limit,'invite_reward_switch,invite_reward_diamond');
50

```

```

1     // 详情 h5 链接
2     $h5 = env('WEB_H5_URL');
3     foreach ($list as &$v) {
4         $v['h5_url'] = rtrim($h5, '/').'/taskdetail/'. $v['id'];
5     }
6     return ApiReturn::success($list);
7 }catch (\Throwable $throwable) {
8     Common::log('邀请页活动展示列表错误',$throwable);
9     return ApiReturn::error();
10 }
11 }
12 }
13

```

```

1 <?php
2
3 namespace App\Http\Controllers\api\v1;
4 use App\Http\Controllers\Controller;
5 use App\Service\v1\Ad;
6 use App\Service\v1\ApiReturn;
7 use Illuminate\Http\Request;
8
9
10 class AdController extends Controller
11 {
12     //获取多个广告
13     public function getByIds(Request $request): object
14     {
15         $ids = $request->get('ids',[1]);
16         return Ad::getByIds($ids);
17     }
18     //获取单个广告
19     public function getById(Request $request): object
20     {
21         $id = $request->get('id',1);
22         return Ad::getAdById($id);
23     }
24     //上传广告用户行为数据

```

```
25 public function addUserAction(Request $request): object
26 {
27     $param = $request->post();
28     if(empty($param['ad_id'])){
29         return ApiReturn::error("请上传广告 ID");
30     }
31     if(empty($param['source_type'])){
32         return ApiReturn::error("请上传用户源类型");
33     }
34     if(empty($param['source_type'])){
35         return ApiReturn::error("请上传用户行为源类型");
36     }
37     return Ad::addUserAction($param);
38 }
39 //首页广告
40 public function getHomeAd(): object
41 {
42     return Ad::getHomeAd();
43 }
44 //个人中心广告
45 public function getUserCenterAd(): object
46 {
47     return Ad::getUserCenterAd();
48 }
49 }
50
```

```
1 <?php
2
3 namespace App\Http\Controllers\api\v1;
4
5 use App\Http\Controllers\Controller;
6 use App\Service\v1\AddressService;
7 use App\Service\v1\ApiReturn;
8 use App\Service\v1\Common;
9 use Illuminate\Http\Request;
10
11 class AddressController extends Controller
12 {
13     /**
14      * 全球地址列表
15      * @param Request $request
16      * @return Object
17      */
18     public function getAddress(Request $request)
19     {
20         try {
21             $pid = $request->get('pid',0);
22             $list = AddressService::getAddress((int)$pid);
23             return ApiReturn::success($list);
24         } catch (\Throwable $throwable) {
25             Common::log('获取全球地址',$throwable);
26             return ApiReturn::error();
27         }
28     }
29 }
```

```

28     }
29
30
31     /**
32     * 积分商城地址（国内地址）
33     * @param Request $request
34     * @return Object
35     */
36     public function cnAddress(Request $request)
37     {
38         try {
39             $pid = intval($request->get('pid',0));
40             $list = $pid > 0 ? AddressService::getSetParentIdToAddress($pid) : AddressService::getSetProvinceAddress();
41             return ApiReturn::success($list);
42         } catch (\Throwable $throwable) {
43             Common::log('获取国内地址',$throwable);
44             return ApiReturn::error();
45         }
46     }
47 }
48

```

```

1  <?php
2  declare(strict_types=1);
3
4  namespace App\Http\Controllers\api\v1;
5
6  use App\Events\BrowseArticleTaskEvent;
7  use App\Events\FirstPublishArticleEvent;
8  use App\Events\FirstPublishTaskEvent;
9  use App\Events\JytFirstPublishTaskEvent;
10 use App\Events\ShareArticleTaskEvent;
11 use App\Http\Controllers\Controller;
12 use App\Http\Requests\ArticleRequest;
13 use App\Http\Requests\GameRequest;
14 use App\Jobs\ErrorLog;
15 use App\Jobs\ReadArticle;
16 use App\Models\CircleAttention;
17 use App\Models\v1\Article;
18 use App\Models\v1\Circle;
19 use App\Models\v1\Game;
20 use App\Models\v1\Task as V1Task;
21 use App\Models\v1\User as V1User;
22 use App\Service\v1\AliContentAudit;
23 use App\Service\v1\ApiCode;
24 use App\Service\v1\ApiReturn;
25 use App\Service\v1\Article as V1Article;
26 use App\Service\v1\ArticleService;
27 use App\Service\v1\Code;
28 use App\Service\v1\Common;
29 use App\Service\v1\CommonFnc;
30 use App\Service\v1\LongArticleTask;
31 use App\Service\v1\Search;
32 use App\Service\v1\task\PublishDynamicTask;

```

```

33 use App\Service\v1\task\PublishVideoTask;
34 use App\Service\v1\SoloCode;
35 use App\Service\v1\User;
36 use Hhxs5\LaravelS\Swoole\Task\Task;
37 use Illuminate\Auth\Access\AuthorizationException;
38 use Illuminate\Http\Request;
39 use Illuminate\Support\Facades\Log;
40 use Illuminate\Support\Facades\Route;
41 use Illuminate\Validation\ValidationException;
42
43 class ArticleController extends Controller
44 {
45
46     /**
47      * 发布长文
48      * @param Request $request
49      * @return Object
50      */
51
52     public function publishLongArticle(Request $request)
53     {
54         try {
55             $isSource = intval($request->post('is_source',1));
56             $param = $request->post();
57             if($isSource == 3 && empty($param['circle_id'])) {
58                 return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
59             }else if($isSource == 2 && (empty($param['task_id']) || empty($param['circle_id']))) {
60                 return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
61             }
62             $userInfo = V1User::find($param['user_id']);
63             // $userInfo = (new App\Models\v1\User())->getUserInfo($param['user_id'],['id','mobile','attestation_type','is_writer']);
64             if (empty($userInfo->mobile)) return ApiReturn::error(ApiCode::NOT_BOUND_ERROR['msg'],
65                 ApiCode::NOT_BOUND_ERROR['code']);
66             if (isset($param['task_id']) && $param['task_id'] > 0) {
67                 $articleCount = User::isParticipation($param['user_id'],$param['task_id']);
68                 if($articleCount > 0) return ApiReturn::error(ApiCode::PARTICIPATION_ERROR['msg'],
69                     ApiCode::PARTICIPATION_ERROR['code']);
70                 if (!User::isTaskEnroll($param['user_id'], $param['task_id'])) return
71                     ApiReturn::error(ApiCode::TASK_ENROLL_ERROR['msg'], ApiCode::TASK_ENROLL_ERROR['code']); // 报名检测
72
73                 // 权限检测
74                 $task = V1Task::find($param['task_id']);
75                 $param['game_id'] = $task['game_id'];
76                 $circle = $task->circle;
77                 $param['circle_id'] = $circle ? $circle['id'] : 0;
78                 $userRole = $userInfo->user_role()->pluck('user_role_id')->toArray();
79                 if (!$userRole) $userRole = [Code::ROLE_TYPE_MAP['default']]; // 给个默认防止bug
80                 $taskRole = $task->user_role()->pluck('user_role_id')->toArray();
81                 if ($taskRole) {
82                     if ($task['participation_type'] == 1) { // 写手标记
83                         if (!in_array(Code::ROLE_TYPE_MAP['writer'], $userRole)) return
84                             ApiReturn::rdata(ApiCode::TASK_PERMISSION_DENIED_ERROR);
85                         return ApiReturn::rdata(ApiCode::TASK_WRITER_PLATFORM_ERROR);
86                     }
87                 }
88             }
89         }
90     }

```



```

32         if (!array_intersect($taskRole, $userRole)) return ApiReturn::rdata(ApiCode::TASK_PERMISSION_DENIED_ERROR);
// 无交集无权限
33     } else return ApiReturn::rdata(ApiCode::TASK_PERMISSION_DENIED_ERROR);
34 } else if($param['circle_id'] > 0) {
35     $param['game_id'] = Circle::where(['id' => $param['circle_id']])>->value('game_id') ? 0;
36 }else{
37     $param['game_id'] = 0;
38 }
39 $param['type'] = 3;
40 $param['resource'] = is_string($param['long_text_content']) ? json_decode($param['long_text_content'],true) :
$param['long_text_content'];
41 unset($param['long_text_content']);
42 $param['text_content'] = $param['video'] = '';
43 $param['images'] = $param['video'] = $auditImages = [];
44 $param['check_status'] = 2;
45 $param['cover'] = empty($param['cover']) ? '' : parse_url($param['cover'],PHP_URL_PATH);
46 foreach ($param['resource'] as &$v) {
47     switch ($v['type']) {
48         case 'text':
49             $param['text_content'] .= empty($v['character_string']) ? '' : strip_tags($v['character_string']);
50             break;

1         case 'image':
2             array_push($auditImages,$v['image']);
3             array_push($param['images'],parse_url($v['image'],PHP_URL_PATH));
4             break;
5         case 'video':
6             $param['video'] = parse_url($v['url'],PHP_URL_PATH);
7             $param['check_status'] = 1;
8             break;
9         case 'game':
10            $gameInfo = Game::with('game_labels')>->where(['id' => $v['game_id']])>->first();
11            if(empty($gameInfo)) {
12                $v['game_id'] = 0;
13                $v['game_name'] = '';
14                $v['game_icon'] = '';
15                $v['game_label'] = [];
16                $v['game_score'] = '';
17            }else{
18                $gameInfo = $gameInfo->toArray();
19                $v['game_id'] = $gameInfo['id'];
20                $v['game_name'] = $gameInfo['game_name'];
21                $v['game_icon'] = $gameInfo['game_icon'];
22                Search::scoreCount($gameInfo);
23                $v['game_score'] = $gameInfo['score'];
24                $gameLabels = array_slice($gameInfo['game_labels'],0,3);
25                foreach ($gameLabels as $gl) {
26                    $v['game_label'][] = [
27                        'label_id' => $gl['id'],
28                        'label_name' => $gl['label_name']
29                    ];
30                }
31            }
32            break;

```

```

33         case 'link':
34             $v['app_url'] = ArticleService::linkToArticleUrl($v['url']);
35             break;
36         }
37     }
38     if(mb_strlen($param['text_content']) > 10000) return
    ApiReturn::error(ApiCode::LONG_ARTICLE_TEXT['msg'],ApiCode::LONG_ARTICLE_TEXT['code']);
39     if(count($param['images']) > 50) return
    ApiReturn::error(ApiCode::LONG_ARTICLE_IMAGE['msg'],ApiCode::LONG_ARTICLE_IMAGE['code']);
40     // 内容截取
41     $param['abstract'] = mb_substr($param['text_content'], 0, Code::ARTICLE_LIST_CONTENT_LIMIT);
42     $contentStatus = AliContentAudit::textAudit([$param['title'],$param['text_content']]);
43     if($contentStatus === SoloCode::ALI_AUDIT_BLOCK) {
44         return ApiReturn::error(ApiCode::ALI_CONTENT_AUDIT['msg'],ApiCode::ALI_CONTENT_AUDIT['code']);
45     }
46     if($param['check_status'] == 2){
47         $param['check_status'] = $contentStatus == 1 ? 2 : 1;
48     }
49     // $auditImages = $param['images'];
50     if(isset($param['cover']) && !in_array($param['cover'],$auditImages)) array_push($auditImages,$param['cover']);

1     foreach ($auditImages as $image) {
2         $imageStatus = AliContentAudit::imageAudit($image);
3         if($imageStatus === SoloCode::ALI_AUDIT_BLOCK) return
    ApiReturn::error(ApiCode::ALI_IMAGE_AUDIT['msg'],ApiCode::ALI_IMAGE_AUDIT['code']);
4         if($param['check_status'] == 2){
5             $param['check_status'] = $imageStatus == 1 ? 2 : 1;
6         }
7     }
8     $task = new LongArticleTask($param,$userInfo->attestation_type);
9     $result = Task::deliver($task);
10    if($result) {
11        FirstPublishTaskEvent::dispatch($param['user_id'],SoloCode::SYSTEM_TASK_FIRST_PUBLISH);
12        FirstPublishArticleEvent::dispatch($param['user_id']);
13        if(isset($param['task_id']) && $param['task_id'] > 0) JytFirstPublishTaskEvent::dispatch($param['user_id']);
14        return ApiReturn::success((object)[]);
15    }
16    return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
17 }catch (\Throwable $throwable) {
18     Common::log('发布帖子',$throwable);
19     return ApiReturn::error();
20 }
21 }
22
23
24 /**
25  * 发布视频
26  * @param Request $request
27  * @return Object
28  */
29 public function publishVideoOld(Request $request)
30 {
31     try {
32         $isSource = intval($request->post('is_source', 1));

```

```

33     $title = (string)$request->post('title', '');
34     $cover = $request->post('cover', '');
35     $video = $request->post('video', '');
36     $circleId = (int)$request->post('circle_id', 0);
37     $taskId = intval($request->post('task_id', 0));
38     $checkStatus = (empty($cover) || empty($video));
39     if (($isSource == 1 || $isSource == 3) && $checkStatus) {
40         return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
41     } else if ($checkStatus || ($isSource == 2 && $taskId <= 0)) {
42         return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
43     }
44     if($taskId > 0 && $circleId <= 0) {
45         return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
46     }
47     $userId = intval($request->post('user_id'));
48     $userInfo = (new
\App\Models\v1\User())->getUserInfo($userId,['id','mobile','attestation_type','is_writer','create_time','experience']);
49     if($taskId > 0) {
50         $taskInfo = \App\Models\v1\Task::where(['id' =>
$taskId])->select('register_start_time','register_end_time','start_exp','end_exp')->first();


---


1         if(empty($taskInfo)) ApiReturn::error(ApiCode::TASK_ERROR['msg'], ApiCode::TASK_ERROR['code']);
2         if($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) {
3             return ApiReturn::error(ApiCode::ROLE_ERROR['msg'], ApiCode::ROLE_ERROR['code']);
4         }
5     }
6     $param = $request->post();
7     $param['circle_id'] = $circleId;
8     $param['label_list'] = $param['label_list'] ?? [];
9     $param['task_id'] = $taskId;
10    $param['type'] = 1;
11    $param['title'] = $title;
12    $param['cover'] = empty($cover) ? '' : parse_url($cover,PHP_URL_PATH);
13    $param['video'] = empty($video) ? '' : parse_url($video,PHP_URL_PATH);
14    if (empty($userInfo->mobile)) return ApiReturn::error(ApiCode::NOT_BOUND_ERROR['msg'],
ApiCode::NOT_BOUND_ERROR['code']);
15    if ($taskId > 0) {
16        $articleCount = User::isParticipation($userId,$taskId);
17        if($articleCount > 0) return ApiReturn::error(ApiCode::PARTICIPATION_ERROR['msg'],
ApiCode::PARTICIPATION_ERROR['code']);
18        if (!User::isTaskEnroll($param['user_id'], $param['task_id'])) return
ApiReturn::error(ApiCode::TASK_ENROLL_ERROR['msg'], ApiCode::TASK_ENROLL_ERROR['code']); // 报名检测
19        if (in_array($userInfo['is_writer'], [0, 5])) $param['is_writer_task'] = $userInfo['is_writer'];
20    }
21    $content = $title;
22    if(isset($param['video_content']) && !empty($param['video_content'])) {
23        $content .= $param['video_content'];
24        $param['content'] = $param['video_content'];
25        $param['content_len'] = mb_strlen($param['content']);
26    } else {
27        $param['content'] = '';
28    }

```

```
29     unset($param['video_content']);
30     if(mb_strlen($param['content']) > 10000) return
    ApiReturn::error(ApiCode::LONG_ARTICLE_TEXT['msg'],ApiCode::LONG_ARTICLE_TEXT['code']);
31     $contentStatus = AliContentAudit::textAudit([$content]);
32     if($contentStatus === SoloCode::ALI_AUDIT_BLOCK) {
33         return ApiReturn::error(ApiCode::ALI_CONTENT_AUDIT['msg'],ApiCode::ALI_CONTENT_AUDIT['code']);
34     }
35     $imageStatus = AliContentAudit::imageAudit($cover);
36     if($imageStatus === SoloCode::ALI_AUDIT_BLOCK) return
    ApiReturn::error(ApiCode::ALI_IMAGE_AUDIT['msg'],ApiCode::ALI_IMAGE_AUDIT['code']);
37     $task = new PublishVideoTask($param,$userInfo->attestation_type);
38     $result = Task::deliver($task);
39     if($result) {
40         FirstPublishTaskEvent::dispatch($param['user_id'],SoloCode::SYSTEM_TASK_FIRST_PUBLISH);
41         return ApiReturn::success((object)[]);
42     }
43     return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
44 }catch (\Throwable $throwable) {
45     Common::log('发布视频',$throwable);
46     return ApiReturn::error();
47 }
48 }
49
50
```

```
1  /**
2   * showdoc
3   * @catalog 发布|修改 动态
4   * @title 修改动态
5   * @description 修改动态的接口
6   * @method POST
7   * @url /api/publishDynamic
8   * @param GameRequest $request
9   * @param int $action 选填 动作 1(默认) 添加 2 修改
10  * @param int $article_id 选填 文章id action 2 必填
11  * @param string $dynamic_content 选填 内容
12  * @return Object {"code": "200", "msg": "ok", "data": []}
13  */
14 public function publishDynamic(Request $request)
15 {
16     try {
17         $param = $request->post();
18         $param['action'] = $request->post('action', Code::ACTION_ADD);
19         $param['type'] = Code::ARTICLE_TYPE_MAP['dynamic'];
20         // 验证 设置
21         Log::info('发布动态入口参数-----',$param);
22         $err = (array)V1Article::checkArticleParam($param);
23         if($err) return ApiReturn::rdata($err);
24         // 插入到 article 表的数据
25         $param['data'] = [
26             'cover' => $param['cover'],
27             'check_status' => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
28             'content' => $param['content'],
29             // article 标题 截取
```

```

30         'title'      => $param['title'] ?? '',
31         'text_content' => $param['text_content'],
32         'content_len' => $param['content_len'],
33         'home_page'   => $param['home_page'],
34         'label_list'  => $param['label_list'],
35         'abstract'    => $param['abstract'],
36         'user_id'     => $param['user_id'],
37     ];
38     if(!empty($data['title'])){
39         if(!V1Article::checkContentBySensitiveWord($data['title'])){
40             $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
41         }
42     }
43     // 敏感词检测-内容
44     if(!empty($data['content'])){
45         if(!V1Article::checkContentBySensitiveWord($data['content'])){
46             $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
47         }
48     }
49     if(!empty($data['text_content'])){
50         if(!V1Article::checkContentBySensitiveWord($data['text_content'])){


---


1         $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
2     }
3 }
4 if ($param['action'] == Code::ACTION_ADD) {
5     // 新增需要的数据 (修改的时候不可更改的数据)
6     $param['data'] = array_merge($param['data'],[
7         'game_id'      => $param['game_id'],
8         'task_id'      => $param['task_id'],
9         'circle_id'    => $param['circle_id'],
10        'type'         => $param['type'],
11        'is_writer_task'=> $param['is_writer_task'],
12        'ip_location' => $param['ip_location'] ?? '',
13    ]);
14    if(isset($param['created_at'])){
15        $param['data']['create_time'] = strtotime($param['created_at']);
16        $param['data']['update_time'] = strtotime($param['created_at']);
17    }
18 }
19 $res = V1Article::addArticle($param);
20 return $res;
21 }catch (\Throwable $throwable) {
22     Common::log('发布动态',$throwable);
23     return ApiReturn::error();
24 }
25 }
26
27 /**
28  * showdoc
29  * @catalog 发布|修改 动态
30  * @title 修改动态
31  * @description 修改动态的接口
32  * @method POST

```

```

33  * @url /api/publishVideo
34  * @param GameRequest $request
35  * @param int $action 选填 动作 1(默认) 添加 2 修改
36  * @param int $article_id 选填 文章id action 2 必填
37  * @param string $video_content 选填 内容
38  * @return Object {"code": "200", "msg": "ok", "data": []}
39  */
40  public function publishVideo(Request $request)
41  {
42      try {
43          $param = $request->post();
44          Log::info('发布视频入口参数', $param);
45          $param['action'] = $request->post('action', Code::ACTION_ADD);
46          $param['type'] = Code::ARTICLE_TYPE_MAP['video'];
47          // 验证 设置
48          $err = (array)V1Article::checkArticleParam($param);
49          if($err) return ApiReturn::rdata($err);
50          // 插入到 article 表的数据

```

```

1      $param['data'] = [
2          'cover'      => $param['cover'],
3          'check_status' => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
4          'content'    => $param['content'],
5          // article 标题 截取
6          'title'      => $param['title'] ?? '',
7          'text_content' => $param['text_content'],
8          'content_len' => $param['content_len'],
9          'home_page'  => $param['home_page'],
10         'label_list' => $param['label_list'],
11         'abstract'   => $param['abstract'],
12         'user_id'    => $param['user_id'],
13     ];
14     if(!empty($data['title'])){
15         if(!V1Article::checkContentBySensitiveWord($data['title'])){
16             $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
17         }
18     }
19     // 敏感词检测-内容
20     if(!empty($data['content'])){
21         if(!V1Article::checkContentBySensitiveWord($data['content'])){
22             $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
23         }
24     }
25     if(!empty($data['text_content'])){
26         if(!V1Article::checkContentBySensitiveWord($data['text_content'])){
27             $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
28         }
29     }
30     if ($param['action'] == Code::ACTION_ADD) {
31         // 新增需要的数据 (修改的时候不可更改的数据)
32         $param['data'] = array_merge($param['data'], [
33             'game_id'    => $param['game_id'],
34             'task_id'    => $param['task_id'],
35             'circle_id'  => $param['circle_id'],

```

```
36         'type'      => $param['type'],
37         'is_writer_task'=> $param['is_writer_task'],
38         'ip_location' => $param['ip_location'] ?? ''
39     });
40 }
41 return V1Article::addArticle($param);
42 }catch (\Throwable $throwable) {
43     Common::log('发布视频',$throwable);
44     return ApiReturn::error();
45 }
46 }
47
48 /**
49  * 分享文章
50  * @param Request $request
51
52  * @return Object
53  */
54 public function shareArticle(Request $request)
55 {
56     try {
57         $articleId = intval($request->post('article_id',0));
58         if($articleId >= 0) {
59             // $articleType = intval(Article::where(['id' => $articleId,'check_status' => 2])->value('type'));
60             // if(in_array($articleType,[1,3])) {
61                 $userId = $request->post('user_id');
62                 ShareArticleTaskEvent::dispatch($userId,$articleId);
63             // }
64         }
65         return ApiReturn::success();
66     }catch (\Throwable $throwable) {
67         Common::log('任务中心分享文章记录',$throwable);
68         return ApiReturn::error();
69     }
70 }
71
72 /**
73  * 社区推荐文章列表
74  * @param Request $request
75  * @return Object
76  */
77 public function getCommunityRecommendArticleList(Request $request)
78 {
79     try {
80         $userId = $request->get('user_id');
81         $platform = (int)$request->get('platform');
82         $page = (int)$request->get('page',1);
83         $limit = (int)$request->get('limit',10);
84         $device_id = $request->get('device_num');
85         $response = V1Article::getCommunityRecommendArticleList($userId,$page,$limit,$platform,$device_id);
86         return response()->json([
87             'code' => 200,
88             'msg' => "OK",
89             'data' => $response['articles'],
90         ])
```

```

39         'total_count' => $response['total_count']
40     });
41 }catch (\Throwable $throwable) {
42     Common::log('社区推荐文章列表',$throwable);
43     return ApiReturn::error();
44 }
45 }
46
47 /**
48  * showdoc
49  * @catalog 发布|修改 游戏评价
50  * @title 发布|修改评价

```

```

1  * @description 发布|修改评价的接口
2  * @method POST
3  * @url /api/article/publishGameEvaluate
4  * @param GameRequest $request
5  * @return Object
6  * @throws AuthorizationException
7  * @throws ValidationException
8  */
9  public function publishGameEvaluate(GameRequest $request): object
10 {
11     $request->validate('publishGameEvaluate');
12     try {
13         $param = $request->post();
14         $param['action'] = $request->post('action', Code::ACTION_ADD);
15         $param['type'] = Code::ARTICLE_TYPE_MAP['evaluate'];
16         // 验证 设置
17         $err = (array)V1Article::checkGameArticleParam($param);
18         if($err) return ApiReturn::rdata($err);
19
20         // 插入到 article 表的数据
21         $param['data'] = [
22             'cover' => $param['cover'],
23             'game_score' => $param['game_score'],
24             'check_status' => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
25             'content' => $param['content'],
26             // article 标题 截取
27             'title' => $param['title'] ?? '',
28             'text_content' => $param['text_content'],
29             'content_len' => $param['content_len'],
30             'home_page' => $param['home_page'],
31             'label_list' => $param['label_list'],
32             'abstract' => $param['abstract'],
33             'user_id' => $param['user_id'],
34         ];
35         if(!empty($data['title'])){
36             if(!v1Article::checkContentBySensitiveWord($data['title'])){
37                 $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
38             }
39         }
40         // 敏感词检测-内容
41         if(!empty($data['content'])){

```



```

42         if(!V1Article::checkContentBySensitiveWord($data['content'])){
43             $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
44         }
45     }
46     if(!empty($data['text_content'])){
47         if(!V1Article::checkContentBySensitiveWord($data['text_content'])){
48             $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
49         }
50     }

```

```

1     if ($param['action'] == Code::ACTION_ADD) {
2         // 新增需要的数据 (修改的时候不可更改的数据)
3         $param['data'] = array_merge($param['data'],[
4             'game_id'    => $param['game_id'],
5             'task_id'    => $param['task_id'],
6             'circle_id'  => $param['circle_id'],
7             'type'       => $param['type'],
8             'is_writer_task'=> $param['is_writer_task'],
9         ]);
10    }
11    return V1Article::addArticle($param);
12 }catch (\Throwable $throwable) {
13     Common::log('发布动态',$throwable);
14     return ApiReturn::error();
15 }
16 }
17
18 /**
19  * 发布游戏评测
20  * @param GameRequest $request
21  * @param int $action 1添加 2 修改
22  * @return Object
23  * @throws \Illuminate\Auth\Access\AuthorizationException
24  * @throws \Illuminate\Validation\ValidationException
25  * @throws \Throwable
26  */
27 public function publishGameEvaluating(GameRequest $request)
28 {
29     $request->validate('publishGameEvaluating');
30     $action = $request->post('action',1);
31     $param = $request->post();
32     $param['type'] = 5;
33     $param['action'] = $action;
34     $param['resource'] = is_string($param['resource']) ? json_decode($param['resource'],true) : $param['resource']; // ios 传过
string
35
36     // 验证 设置
37     $err = V1Article::checkGameArticleParam($param);
38     if($err) return ApiReturn::error($err);
39     // 插入到 article 表的数据
40     $param['data'] = [
41         'cover'        => $param['cover'],
42         'game_score'    => $param['game_score'],
43         'image_score'   => $param['image_score'],

```

```

44     'experience_score' => $param['experience_score'],
45     'playing_score'   => $param['playing_score'],
46     'check_status'    => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
47     'content'         => $param['content'],
48     // article 标题 截取
49     'title'           => "",
50     'text_content'     => $param['text_content'],

```

```

1     'content_len'     => $param['content_len'],
2     'home_page'       => $param['home_page'],
3     'label_list'      => $param['label_list'],
4     'abstract'        => $param['abstract'],
5     'spend'           => $param['spend'],
6     'user_id'         => $param['user_id'],
7 ];
8 if(!empty($data['title'])){
9     if(!V1Article::checkContentBySensitiveWord($data['title'])){
10         $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
11     }
12 }
13 // 敏感词检测-内容
14 if(!empty($data['content'])){
15     if(!V1Article::checkContentBySensitiveWord($data['content'])){
16         $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
17     }
18 }
19 if(!empty($data['text_content'])){
20     if(!V1Article::checkContentBySensitiveWord($data['text_content'])){
21         $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
22     }
23 }
24 if ($param['action'] == Code::ACTION_ADD) {
25     // 新增需要的数据 (修改的时候不可更改的数据)
26     $param['data'] = array_merge($param['data'],[
27         'game_id'     => $param['game_id'],
28         'task_id'     => $param['task_id'],
29         'circle_id'   => $param['circle_id'],
30         'type'        => $param['type'],
31         'is_writer_task'=> $param['is_writer_task'],
32     ]);
33 }
34
35 return V1Article::addArticle($param);
36 }
37
38
39 /**
40  * showdoc
41  * @catalog 个人稿件管理
42  * @title 个人稿件管理
43  * @description 个人稿件管理的接口
44  * @method POST
45  * @url /api/article/publishGameEvaluate
46  * @param ArticleRequest $request

```

```

47  * @return Object
48  * @throws AuthorizationException
49  * @throws ValidationException
50  */

1  public function personalArticleList(ArticleRequest $request): object
2  {
3      $request->validate('personalArticleList');
4      try {
5          $limit      = $request->get('limit',10);
6          $checkStatus = $request->get('check_status',0);
7          $uid        = $request->get('user_id');
8          $where       = ['user_id' => $uid];
9          $keyword     = $request->get('keyword');
10         if($keyword){
11             array_push($where,['text_content','like','%{$keyword}%']);
12             \App\Service\v2\UserSearchLog::addUserSearchLog($keyword, $uid);
13         }
14         if ($checkStatus == 4) {
15             $checkStatus = [1,3];
16         } else if ($checkStatus != 0){
17             $checkStatus = [$checkStatus];
18         }
19         $list = (new Article())->getUserAllArticleLists($where, $checkStatus,$limit);
20         Article::handleResorce($list);
21         foreach (Code::ARTICLE_TYPE as $v) {
22             Comment::disposeLike($uid,$list['data'],$v);
23         }
24         return ApiReturn::success($list);
25     }catch (\Exception $e) {
26         ErrorLog::dispatch(\Route::current()->getActionName(),CommonFnc::getRealIP(),json_encode($e),['file'=>$e->getFile(),'line'=>$e->getLine(),'code'=>$e->getCode(),'message'=>$e->getMessage()],time()->onQueue('error_log');
27         return ApiReturn::error();
28     }
29 }

30
31 /**
32  * 圈子文章列
33  * @param ArticleRequest $request
34  * @return Object
35  */
36 public function getCircleArticleList(ArticleRequest $request): object
37 {
38     Log::info('圈子文章列表入参',$request->all());
39     try {
40         $limit = $request->get('limit',10);
41         $order = $request->get('order',1);
42         $circle_id = $request->get('circle_id');
43         if(empty($circle_id)){
44             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
45         }
46         $uid = $request->get('user_id');
47         $article_type = $request->get('article_type') ? $request->get('article_type') : 1;

```

```

48     $elite = 0;
49     $type = [];
50     switch ($article_type){
1
        case 1:
2            break;
3        case 2:
4            $elite = 1;
5            break;
6        case 3:
7            $type = [2,3];
8            break;
9        case 4:
10           $type = [1];
11           break;
12        default:
13           return ApiReturn::rdata(ApiCode::PARAM_ERROR);
14    }
15    $device_id = $request->get('device_num');
16    $list = V1Article::getCircleArticleLists($circle_id,$elite,$limit,$type,$uid,$order,$device_id);
17    if($uid){
18        $rs = CircleAttention::query()->where(['user_id' => $uid, 'circle_id' => $circle_id])->first();
19        $role_type = $rs ? $rs['role_type'] : 1;
20        foreach ($list['data'] as &$v) {
21            $v['role_type'] = $role_type;
22        }
23    }
24    return ApiReturn::success($list);
25 }catch (\Exception $e) {
26
    ErrorLog::dispatch(Route::current()->getActionName(),CommonFnc::getRealIP(),json_encode($e),['file'=>$e->getFile(),'line'=>
    $e->getLine(),'code'=>$e->getCode(),'message'=>$e->getMessage(),time())->onQueue('error_log');
27     return ApiReturn::error($e->getMessage());
28 }
29 }
30 /**
31  * 文章详情
32  * @param Request $request
33  * @return Object
34  */
35 public function getArticleInfo(Request $request): object
36 {
37     try {
38         $uid = $request->get('user_id');
39         $deviceNum = $request->get('device_num');
40         $id = $request->get('article_id',0);
41         $type = $request->get('type',1);
42         if(empty($id)){
43             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
44         }
45         if(empty($uid)) $uid = 0;
46         switch ($type) {
47             case Code::ARTICLE_TYPE_MAP['video']:
48                 $info = V1Article::getVideoInfo($id,$uid);

```

```

49         break;
50         case Code::ARTICLE_TYPE_MAP['dynamic']:
1         $info = V1Article::getDynamicInfo($id,$uid);
2         break;
3         case Code::ARTICLE_TYPE_MAP['long_text']:
4         $info = V1Article::getLongTextInfo($id,$uid);
5         break;
6         case Code::ARTICLE_TYPE_MAP['evaluate']:
7         $info = V1Article::getEvaluateInfo($id,$uid);
8         break;
9         case Code::ARTICLE_TYPE_MAP['evaluating']:
10        $info = V1Article::getEvaluatingInfo($id,$uid);
11        break;
12        default:
13        return ApiReturn::rdata(ApiCode::PARAM_ERROR);
14    }
15    if(!empty($info)){
16        // 获奖判断
17        $info['is_awarded'] = (bool)$info['user_award'];
18        unset($info['user_award']);
19        $info['is_attention'] = User::isAttention($uid,$info['user_id']);
20        dispatch(new ReadArticle($uid,$info,$deviceNum))->onQueue('read_article');
21        $info['create_time'] = Common::disposeDateTime(strtotime($info['create_time']));
22        $info['like_num'] = Common::disposeAmount($info['like_num']);
23        return ApiReturn::success($info);
24    }else return ApiReturn::error('该内容已被删除');
25    }catch (\Exception $e) {
26
ErrorLog::dispatch(Route::current()->getActionName(),CommonFnc::getRealIP(),json_encode($e),['file'=>$e->getFile(),'line'=>
    $e->getLine(),'code'=>$e->getCode(),'message'=>$e->getMessage(),time())->onQueue('error_log');
27        return ApiReturn::error($e->getMessage());
28    }
29 }
30 /**
31  * showdoc
32  * @catalog 社区关注信息流
33  * @title 社区关注信息流
34  * @description 社区关注信息流的接口
35  * @method POST
36  * @url /api/article/publishGameEvaluate
37  * @param ArticleRequest $request
38  * @return Object
39  */
40 public function getAttentionArticleList(ArticleRequest $request): object
41 {
42     try {
43         $userId = $request->get('user_id');
44         $platform = (int)$request->get('platform');
45         $page = (int)$request->get('page',1);
46         $limit = (int)$request->get('limit',10);
47         $device_id = $request->get('device_num');
48         $response = V1Article::getAttentionArticleList($userId,$page,$limit,$platform,$device_id);
49         return response()->json([

```

```
50         'code' => 200,
1         'msg' => "OK",
2         'data' => $response['articles'],
3         'total_count' => $response['total_count']
4     });
5 }catch (\Throwable $throwable) {
6     Common::log('社区关注信息流',$throwable);
7     return ApiReturn::error();
8 }
9 }
10 //模版
11 public function templates()
12 {
13
14     return V1Article::templates();
15 }
16 /**
17  * showdoc
18  * @catalog 社区关注信息流
19  * @title 社区关注信息流
20  * @description 社区关注信息流的接口
21  * @method POST
22  * @url /api/article/publishGameEvaluate
23  * @param ArticleRequest $request
24  * @return Object
25  */
26 public function getAttentionArticleListNew(ArticleRequest $request): object
27 {
28     try {
29         $userId = $request->get('user_id');
30         $platform = (int)$request->get('platform');
31         $page = (int)$request->get('page',1);
32         $limit = (int)$request->get('limit',10);
33         $device_id = $request->get('device_num');
34         return V1Article::getAttentionArticleListNew($userId,$page,$limit,$platform,$device_id);
35     }catch (\Throwable $throwable) {
36         Common::log('社区关注信息流',$throwable);
37         return ApiReturn::error($throwable->getMessage());
38     }
39 }
40 //推荐文章
41 public function getRecommend(ArticleRequest $request)
42 {
43     $limit = $request->get('limit', 10);
44     $user_id = $request->get('user_id');
45     $game_id = $request->get('game_id');
46     $elite = $request->get('elite');
47     $page = $request->get('page',1);
48     $device_id = $request->get('device_num');
49     return V1Article::getRecommendEs($user_id, $page, $limit, $game_id, $elite,0,$device_id);
50 }
1 //文章详情
2 public function detail(ArticleRequest $request)
```

```

3      {
4          $article_id = $request->get('article_id');
5          if (!$article_id) {
6              return ApiReturn::error('请选择要查看的文章');
7          }
8          $user_id = $request->get('user_id');
9          return V1Article::detail($article_id,$user_id);
10     }
11     //评论列表
12     public function commentList(ArticleRequest $request)
13     {
14         $article_id = $request->get('article_id');
15         if(empty($article_id)){
16             return ApiReturn::error('参数错误');
17         }
18         $limit = $request->get('limit', 10);
19         $user_id = $request->get('user_id');
20         return V1Article::commentList($article_id, $limit, $user_id);
21     }
22     //增加浏览记录
23     public function addReadRecord(ArticleRequest $request)
24     {
25         $article_id = $request->get('article_id');
26         $user_id = $request->get('user_id');
27         $deviceNum = $request->post('device_num','');
28         if(empty($article_id)){
29             return ApiReturn::error('参数错误');
30         }
31         return V1Article::addReadRecord($article_id,$user_id,$deviceNum);
32     }
33     /**
34      * 文章评论回复列表
35      * @param Request $request
36      * @return Object
37      */
38     public function getCommentReplyList(Request $request): object
39     {
40         try {
41             $commentId = intval($request->get('comment_id',0));
42             $limit = intval($request->get('limit',10));
43             $userId = intval($request->get('user_id'));
44             if(empty($commentId)){
45                 return ApiReturn::rdata(ApiCode::PARAM_ERROR);
46             }
47             $exclude_ids = $request->get('exclude_ids','');
48             return V1Article::getCommentReplyList($commentId,$userId,$limit,$exclude_ids);
49         }catch (\Exception $e) {
50             return ApiReturn::error();
51         }
52     }
53     //游戏文章列表
54     public function gameGameArticleList(ArticleRequest $request)
55     {

```

```
6     $game_id = $request->get('game_id');
7     $circle_id = $request->get('circle_id');
8     if(empty($game_id) && empty($circle_id)){
9         return ApiReturn::error('参数错误');
10    }
11    $limit = $request->get('limit', 10);
12    $user_id = $request->get('user_id');
13    $elite = $request->get('elite');
14    $is_review = $request->get('is_review');
15    Log::info('获取游戏文章列表入参', $request->all());
16    $device_id = $request->get('device_num');
17    return V1Article::getRecommend($user_id, $limit, $game_id, $elite,$is_review,$circle_id,$device_id,true);
18 }
19 //同步来源于SoloGame的文章
20 public function syncArticleBySoloGame(Request $request)
21 {
22     $param = $request->all();
23     if(empty($param['title'])){
24         return ApiReturn::error('请输入文章标题');
25     }
26     if(empty($param['content'])){
27         return ApiReturn::error('请输入文章内容');
28     }
29     if(empty($param['rss_guid'])){
30         return ApiReturn::error('请输入rss_guid');
31     }
32     if(empty($param['link_url'])){
33         return ApiReturn::error('请输入文章url');
34     }
35     if(empty($param['lang'])){
36         return ApiReturn::error('请输入文章语言');
37     }
38     return V1Article::syncArticleBySoloGame($param);
39 }
40 public function syncArticleBySoloPublisher(Request $request)
41 {
42     $params = $request->all();
43     return V1Article::syncArticleBySoloPublisher($params);
44 }
45 //文章相关推荐
46 public function getRelatedArticle(ArticleRequest $request)
47 {
48     $user_id = $request->get('user_id');
49     $article_id = $request->get('article_id');
50     if(empty($article_id)){


---


1         return ApiReturn::success();
2     }
3     //社区ID
4     $circle_id = $request->get('circle_id');
5     //游戏ID
6     $game_id = $request->get('game_id');
7     $page = $request->get('page',1);
8     $limit = $request->get('limit', 10);
```



```

9      //是否测评
10     $type = $request->get('type',0);
11     if($type == '(null)'){
12         $type = 0;
13     }
14     if($circle_id == '(null)'){
15         $circle_id = 0;
16     }
17     $device_id = $request->get('device_num');
18     return V1Article::getRelatedArticle($user_id,$article_id, $circle_id, $game_id, $page, $limit,$type,$device_id);
19 }
20 }
21

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1\BindingGame;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\v1\BindingGame\GameGameLabel;
7  use App\Models\v1\BindingGame\GameImage;
8  use App\Models\v1\BindingGame\GameInfo;
9  use App\Models\v1\BindingGame\GameLabel;
10 use App\Models\v1\BindingGame\GameManufacturer;
11 use App\Models\v1\Circle;
12 use App\Models\v1\Game;
13 use App\Models\v1\GameDefaultPlatformGame;
14 use App\Service\v1\ApiCode;
15 use App\Service\v1\ApiReturn;
16 use App\Service\v1\check_url\Tap;
17 use App\Service\v1\Common;
18 use Illuminate\Http\Request;
19 use Illuminate\Support\Facades\DB;
20 use function Stringy\create;
21
22 class BindingGameController extends Controller
23 {
24     /**
25      * 三方平台游戏绑定solo平台游戏
26      * @param Request $request
27      * @return Object
28      */
29     public function editBindingGame(Request $request)
30     {
31         $platformId = (int)$request->post('platform_id',0);
32         $platformGameId = (int)$request->post('platform_game_id',0);
33
34         if($platformId <= 0 || $platformGameId <= 0) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
35         $params = $request->post('game_info',[]);
36         if(!isset($params['game_name']) || empty($params['game_name'])) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
37         try {
38             DB::beginTransaction();

```

```

39         $gameInfo = Game::query()->withoutGlobalScope('status')->where(['game_name' =>
(string)$params['game_name'])->first();
40         $labelIds = [];
41         if(empty($params['game_label'])) {
42             $params['game_label'] = '';
43         }else{
44             foreach ($params['game_label'] as $label) {
45                 $labelInfo = GameLabel::query()->where(['label_name' => $label])->first();
46                 if(empty($labelInfo)) {
47                     $labelInfo = GameLabel::query()->create(['label_name' => $label]);
48                 }
49                 $labelIds[] = $labelInfo->id;
50             }

```

```

1         $params['game_label'] = GameLabel::query()->select('id as
label_id','label_name')->whereIn('id',$labelIds)->get()->toJson();
2     }
3     if(empty($gameInfo)) {
4         $manufacturerName = (string)$params['manufacturer_name'];
5         if(!empty($manufacturerName)) {
6             $manufacturerInfo = GameManufacturer::query()->where(['name' => $manufacturerName])->first();
7             if(empty($manufacturerInfo)) {
8                 $manufacturerInfo = GameManufacturer::query()->create(['name' => $manufacturerName]);
9             }
10            $params['manufacturer_id'] = $manufacturerInfo->id;
11        }
12        $params['status'] = 0;
13        $gameInfo = Game::query()->create($params);
14        Circle::create([
15            'game_id' => $gameInfo->id,
16            'circle_name' => $gameInfo->game_name,
17            'avatar' => $gameInfo->game_icon,
18            'cover' => $gameInfo->game_cover,
19        ]);
20    }else{
21        $gameInfo->game_cover = $params['game_cover'];
22        $gameInfo->score = $params['score'];
23        $gameInfo->game_icon = $params['game_icon'];
24        $gameInfo->game_label = $params['game_label'];
25        $gameInfo->save();
26    }
27    if (count($labelIds)) $gameInfo->game_labels()->attach($labelIds);
28    GameInfo::query()->updateOrCreate([
29        'game_id' => $gameInfo->id
30    ],[
31        'content' => $this->deleteBr($params['content']),
32        'version' => $params['version'] ? ': ',
33        'size' => $params['size'] ? ': 0',
34        'service' => $params['service'] ? ': ',
35        'game_update_time' => $params['game_update_time'],
36        'update_log' => $params['update_log'],
37    ]);
38    if(!empty($params['images'])) {
39        GameImage::query()->where(['game_id' => $gameInfo->id,'type' => 1])->delete();

```

```
40     $images = [];
41     foreach ($params['images'] as $key => $image) {
42         $images[] = [
43             'game_id' => $gameInfo->id,
44             'url' => $image,
45             'type' => 1,
46             'sort' => $key+1,
47             'status' => 2
48         ];
49         GameImage::query()->insert($images);
50     }

```

```
1     }
2     if(!empty($params['videos'])) {
3         GameImage::query()->where(['game_id' => $gameInfo->id,'type' => 2])->delete();
4         $videos = [];
5         foreach ($params['videos'] as $key => $video) {
6             $videos[] = [
7                 'game_id' => $gameInfo->id,
8                 'url' => $video,
9                 'type' => 2,
10                'sort' => $key+1,
11                'status' => 2
12            ];
13            GameImage::query()->insert($videos);
14        }
15    }
16    GameDefaultPlatformGame::query()->updateOrCreate([
17        'game_id' => $gameInfo->id,
18        'platform_type' => $platformId
19    ],['platform_game_id' => $platformGameId]);
20    DB::commit();
21    return ApiReturn::success();
22 }catch (\Throwable $throwable) {
23     DB::rollBack();
24     Common::log('绑定游戏错误',$throwable);
25     return ApiReturn::error();
26 }
27 }
28
29 /**
30  * 根据第三方平台id和游戏名称搜索游戏
31  * @param Request $request
32  * @return Object
33  */
34 public function games(Request $request)
35 {
36     $platformId = (int)$request->get('platform_id',0);
37     $gameName = (string)$request->get('game_name','');
38     if($platformId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
39     $page = (int)$request->get('page',1);
40     $limit = (int)$request->get('limit',20);
41     $page = ($page - 1) * $limit;
42     try {
```

```

43     $games = Game::query()
44     ->join('game_default_platform_games as pg','games.id','=', 'pg.game_id')
45     ->where('games.game_name','like','%{$gameName}%')
46     ->where(['pg.platform_type' => $platformId])
47     ->withoutGlobalScope('status')
48     ->offset($page)->limit($limit)->get(['games.id','games.game_name','pg.platform_game_id']);
49 //     $games = GameDefaultPlatformGame::query()->with(['game' => function($query) use($gameName) {
50 //         $query->withoutGlobalScope('status')->where('game_name','like','%{$gameName}%')->select('id','game_name');

1 //     }])->where(['platform_type' => $platformId])->offset($page)->limit($limit)->get();
2 //     $list = [];
3 //     foreach ($games as $game) {
4 //         if(!empty($game->game)) $list[] = [
5 //             'id' => $game->game->id,
6 //             'game_name' => $game->game->game_name,
7 //             'platform_game_id' => $game->platform_game_id
8 //         ];
9 //     }
10     return ApiReturn::success(['games' => $games]);
11 }catch (\Throwable $throwable) {
12     DB::rollBack();
13     Common::log('三方平台游戏列表错误',$throwable);
14     return ApiReturn::error();
15 }
16 }
17
18 /**
19  * 内容过滤br标签
20  * @param string $content
21  * @return string
22  */
23 public function deleteBr(string $content) :string
24 {
25     if(empty($content)) return "";
26     $preg = "/<br.*?\\>/";
27     $content = preg_replace($preg,"\\n",$content);
28     return $content;
29 }
30 }
31

```

```

1 <?php
2
3 namespace App\Http\Controllers\api\v1;
4
5 use App\Http\Controllers\Controller;
6 use App\Service\v1\ApiReturn;
7
8 class CaptchaController extends Controller
9 {
10     public function getCaptcha()
11     {
12         return ApiReturn::success(app('captcha')->create('default', true));
13     }
14 }

```

15

```
1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Service\CodeMessage;
7  use App\Service\v1\ApiCode;
8  use App\Service\v1\ApiReturn;
9  use App\Service\v1\Checkin;
10 use Illuminate\Http\Request;
11 use Illuminate\Support\Facades\Log;
12 use Illuminate\Support\Facades\Validator;
13
14 class CheckinController extends Controller
15 {
16     //创建打卡
17     public function add(Request $request)
18     {
19         $param = $request->all();
20         if (empty($param['user_id'])) {
21             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
22         }
23         // 定义校验规则
24         $rules = [
25             'content' => 'required',
26             'type' => 'required|numeric|min:1|max:3',
27             'start_time' => ['required', 'date', 'after_or_equal:'.date("Y-m-d H:i"), 'before_or_equal:end_time'],
28             'end_time' => ['required', 'date', 'after:start_time'],
29             'community_id' => 'required_if:type,3',
30             // 添加其他字段的校验规则
31         ];
32         $messages = [
33             'content.required' => '请输入内容',
34             'type.required' => '缺少参数',
35             'type.numeric' => '参数格式不正确',
36             'type.min' => '参数格式不正确',
37             'type.max' => '参数格式不正确',
38             'start_time.required' => '请输入开始时间',
39             'start_time.date' => '开始时间格式不正确',
40             'start_time.after_or_equal' => '开始时间不能早于当前时间',
41             'start_time.before_or_equal' => '开始时间不能晚于结束时间',
42             'end_time.required' => '请输入结束时间',
43             'end_time.date' => '结束时间格式不正确',
44             'end_time.after' => '结束时间必须晚于开始时间',
45             'community_id.required_if' => '请选择社群'
46         ];
47         // 执行校验并获取校验后的数据
48         $validator = Validator::make($param, $rules, $messages);
49         if ($validator->fails()) {
50             return ApiReturn::error($validator->errors()->first());
51         }
52         return Checkin::add($param);
53     }
54 }
```

```
3     }
4
5
6     //打卡创建详情
7     public function info(Request $request)
8     {
9         $user_id = $request->get('user_id');
10        if (empty($user_id)) {
11            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
12        }
13        //打卡创建ID
14        $checkin_creation_id = $request->get('checkin_creation_id');
15        if (empty($checkin_creation_id)) {
16            return ApiReturn::rdata(CodeMessage::common(30001));
17        }
18        //第三方群类型
19        $external_group_type = $request->get('external_group_type','wechat');
20        //第三方群ID
21        $external_group_id = $request->get('external_group_id');
22        //加权因子
23        $iv = $request->get('iv');
24        //token
25        $token = $request->bearerToken();
26        //社群邀请用户ID
27        $community_inviter_id = $request->get('community_inviter_id',0);
28        return Checkin::info($user_id,
29        $checkin_creation_id,$external_group_type,$external_group_id,$iv,$token,$community_inviter_id);
30    }
31
32    //点击打卡
33    public function click(Request $request)
34    {
35        $user_id = $request->post('user_id');
36        if (empty($user_id)) {
37            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
38        }
39        //打卡创建ID
40        $checkin_creation_id = $request->post('checkin_creation_id');
41        if (empty($checkin_creation_id)) {
42            return ApiReturn::rdata(CodeMessage::common(30001));
43        }
44        //第三方群类型
45        $external_group_type = $request->post('external_group_type','wechat');
46        //第三方群ID
47        $external_group_id = $request->post('external_group_id');
48        //加密参数
49        $iv = $request->post('iv');
50        //token
51        $token = $request->bearerToken();
52
53        return Checkin::click($user_id, $checkin_creation_id, $external_group_type, $external_group_id,$iv,$token);
54    }
55
56    //已参与人员列表
57    public function participants(Request $request){
```

```

5      //打卡创建ID
6      $checkin_creation_id = $request->get('checkin_creation_id');
7      if (empty($checkin_creation_id)) {
8          return ApiReturn::rdata(CodeMessage::common(30001));
9      }
10     $limit = $request->get('limit',10);
11     return Checkin::participants($checkin_creation_id,$limit);
12 }
13 //我参与的打卡
14 public function myCheckinsParticipated(Request $request){
15     $limit = $request->get('limit',10);
16     $user_id = $request->post('user_id');
17     if (empty($user_id)) {
18         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
19     }
20     return Checkin::myCheckinsParticipated($user_id,$limit);
21 }
22 //结束打卡
23 public function stop(Request $request) {
24     $checkin_creation_id = $request->get('checkin_creation_id');
25     if (empty($checkin_creation_id)) {
26         return ApiReturn::rdata(CodeMessage::common(30001));
27     }
28     $user_id = $request->post('user_id');
29     if (empty($user_id)) {
30         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
31     }
32     return Checkin::stop($user_id,$checkin_creation_id);
33 }
34 //获取不受限制小程序分享海报
35 public function getWechatMiniProgramSharePoster(Request $request)
36 {
37     $param = $request->all();
38     if (empty($param['user_id'])) {
39         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
40     }
41     if (empty($param['checkin_creation_id'])) {
42         return ApiReturn::rdata(CodeMessage::common(30001));
43     }
44     if (empty($param['page'])) {
45         return ApiReturn::rdata(CodeMessage::common(30001));
46     }
47     return Checkin::getWechatMiniProgramSharePoster($param);
48 }
49 }
50

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Events\SearchEvent;
6  use App\Http\Controllers\Controller;
7  use App\Models\UserBrowseLog;

```

```

8  use App\Models\v1\BindingGame\GameGameLabel;
9  use App\Models\v1\Circle;
10 use App\Service\v1\ApiCode;
11 use App\Service\v1\ApiReturn;
12 use App\Service\v1\Article as V1Article;
13 use App\Service\v1\CircleService;
14 use App\Service\v1\Common;
15 use App\Service\v1\CommonFnc;
16 use App\Service\v1\Search;
17 use App\Service\v1\UserNotifications;
18 use Illuminate\Http\Request;
19
20 class CircleController extends Controller
21 {
22     /**
23      * 搜索圈子
24      * @param Request $request
25      * @return Object
26      */
27     public function circleSearch(Request $request)
28     {
29         try {
30             $name = strval($request->get('circle_name',''));
31             $userId = $request->get('user_id',1);
32             $deviceNum = $request->get('device_num','');
33             $page = $request->get('page',1);
34             $limit = $request->get('limit',10);
35             $field = ["id","circle_name","avatar","game_id","articles_num","attention_num"];
36             $circles = Search::circles($name,$page,$limit,$field);
37             if(!empty($name)) SearchEvent::dispatch($name,(int)$userId,(string)$deviceNum);
38             return ApiReturn::success($circles);
39         }catch (\Throwable $throwable) {
40             Common::log('搜索圈子',$throwable);
41             return ApiReturn::error();
42         }
43     }
44
45
46     /**
47      * 热门圈子
48      * @param Request $request
49      * @return Object
50      */

```

```

1  public function getHotCircleList(Request $request)
2  {
3      try {
4          $userId = $request->get('user_id');
5          $page = (int)$request->get('page',1);
6          $limit = (int)$request->get('limit',15);
7          $type = (int)$request->get('type',1);
8          $list = CircleService::getHotCircleList($userId,$page,$limit,$type);
9          $rdata = [
10             'code' => 200,

```



```
11         'msg' => 'OK',
12         'data' => $list['circles'],
13         'total_count'=>$list['total_count']
14     ];
15     return response()->json($rdata);
16 } catch (\Throwable $throwable) {
17     Common::log('热门圈子',$throwable);
18     return ApiReturn::error();
19 }
20 }
21 //圈子列表
22 public function getCircleList(Request $request)
23 {
24     $userId = $request->get('user_id');
25     $page = (int)$request->get('page',1);
26     $limit = (int)$request->get('limit',15);
27     $type = (int)$request->get('type',1);
28     return CircleService::getCircleList($userId,$page,$limit,$type);
29 }
30 /**
31  * 最近预览的圈子
32  * @param Request $request
33  * @return Object
34  */
35 public function getUserLastBrowseList(Request $request): object
36 {
37     try {
38         $uid = $request->get('user_id');
39         $device_num = $request->get('device_num');
40         $limit = $request->get('limit', 20);
41         $list = (new UserBrowseLog())->getCircleBrowseList($uid,$device_num ? ':' : '',$limit);
42         foreach($list as &$v){
43             $label = GameGameLabel::query()
44                 ->where('game_id',$v['game_id'])
45                 ->whereHas('game_labels')
46                 ->with('game_labels',function ($query){
47                     $query->select('id','label_name');
48                 })
49                 ->limit(3)
50                 ->get();
51
52             ->toArray();
53             $label = array_column($label,'game_labels');
54             $label = array_column($label,'label_name');
55             $v['game_label_txt'] = '';
56             if(!empty($label)){
57                 $v['game_label_txt'] = implode(',',$label);
58             }
59             $v['ut'] = date('Y.m.d',$v['ut']);
60             $v['avatar'] = Common::concatUrlStr($v['avatar']);
61             $v['cover'] = Common::concatUrlStr($v['cover']);
62             $v['is_attention'] = CircleService::isAttention($uid,$v['id']);
63             $v['articles_num'] = Common::disposeAmount($v['articles_num']);
64             $v['attention_num'] = Common::disposeAmount($v['attention_num']);
65         }
66     } catch (\Exception $e) {
67         return ApiReturn::error($e->getMessage());
68     }
69 }
```

```

14         $v['circle_id'] = $v['id'];
15     }
16     $res = [
17         'total_count' => (int)$limit,
18         'circles' => $list
19     ];
20     return ApiReturn::success($res);
21 } catch (\Exception $e) {
22     return ApiReturn::error($e->getMessage());
23 }
24 }
25 public function getRecommendCircleList(Request $request)
26 {
27     $game_label_ids = $request->get('game_label_ids','');
28     $label_ids = [];
29     if(!empty($game_label_ids)){
30         $label_ids = explode(',',$game_label_ids);
31         //去除空
32         $label_ids = array_filter($label_ids);
33         //去重
34         $label_ids = array_unique($label_ids);
35     }
36     return CircleService::getRecommendCircleList($label_ids);
37 }
38 /**
39  * 圈子详情
40  * @param Request $request
41  * @return Object
42  */
43 public function getCircleInfo(Request $request): object
44 {
45     try {
46         $userId = $request->get('user_id');
47         $circleId = intval($request->get('circle_id',0));
48         if(empty($circleId)){
49             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
50         }
51     } catch (\Exception $e) {
52         return ApiReturn::error($e->getMessage());
53     }
54 }
55
56 //圈子文章列表
57 public function articleList(Request $request)
58 {
59     $limit = $request->get('limit', 10);
60     $type = $request->get('type', 'all');
61     $orderBy = $request->get('order_by', 'publish_time');
62     $circleId = $request->get('circle_id');
63     if (empty($circleId)) {

```

```

17     $list = [];
18 }else{
19     $lang = CommonFnc::lang();
20     $device_id = $request->get('device_num');
21     $userId = $request->get('user_id');
22     $list = CircleService::articleList($limit,$type,$orderBy,$circleId,$userId,$device_id,$lang);
23 }
24 return ApiReturn::success($list);
25 }
26 }
27

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5
6  use App\Http\Controllers\Controller;
7  use App\Service\CodeMessage;
8  use App\Service\v1\ApiCode;
9  use App\Service\v1\ApiReturn;
10 use App\Service\v1\Community;
11 use App\Service\v1\WechatMiniProgram;
12 use Illuminate\Http\JsonResponse;
13 use Illuminate\Http\Request;
14
15 class CommunityController extends Controller
16 {
17     /**
18      * 创建社群
19      * @param Request $request
20      * @return JsonResponse|Object
21      */
22     public function add(Request $request)
23     {
24         $user_id = $request->post('user_id');
25         if (empty($user_id)) {
26             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
27         }
28         $name = $request->post('name');
29         if (empty($name)) {
30             return ApiReturn::rdata(CodeMessage::common(30001));
31         }
32         return Community::add($user_id, $name);
33     }
34
35     /**
36      * 修改社群
37      * @param Request $request
38      * @return array|JsonResponse|Object
39      */
40     public function edit(Request $request)
41     {
42         $user_id = $request->post('user_id');

```

```

43     if (empty($user_id)) {
44         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
45     }
46     $community_id = $request->post('community_id');
47     if (empty($community_id)) {
48         return ApiReturn::rdata(CodeMessage::common(30001));
49     }
50     $type = $request->post('type');

```

```

1     if (empty($type)) {
2         return ApiReturn::rdata(CodeMessage::common(30001));
3     }
4     $data = $request->post('data');
5     if (!isset($data) || $data === '') {
6         return ApiReturn::rdata(CodeMessage::common(30001));
7     }
8     return Community::edit($user_id, $community_id, $type, $data);
9 }
10
11 /**
12  * 加入社群
13  * @param Request $request
14  * @return JsonResponse|Object
15  */
16 public function join(Request $request)
17 {
18     $user_id = $request->post('user_id');
19     if (empty($user_id)) {
20         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
21     }
22     $community_id = $request->post('community_id');
23     if (empty($community_id)) {
24         return ApiReturn::rdata(CodeMessage::common(30001));
25     }
26     $nick_name = $request->post('nick_name');
27     if (empty($nick_name)) {
28         return ApiReturn::rdata(CodeMessage::common(30001));
29     }
30     return Community::join($user_id, $community_id, $nick_name);
31 }
32
33 //社群详情
34 public function info(Request $request)
35 {
36     $user_id = $request->post('user_id');
37     if (empty($user_id)) {
38         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
39     }
40     $community_id = $request->get('community_id');
41     if (empty($community_id)) {
42         return ApiReturn::rdata(CodeMessage::common(30001));
43     }
44     return Community::info($user_id, $community_id);
45 }

```

```

46
47 //社群成员
48 public function memberships(Request $request)
49 {
50     $user_id = $request->get('user_id');
51
52     if (empty($user_id)) {
53         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
54     }
55     $community_id = $request->get('community_id');
56     if (empty($community_id)) {
57         return ApiReturn::rdata(CodeMessage::common(30001));
58     }
59     $limit = $request->get('limit', 10);
60     return Community::memberships($user_id, $community_id, $limit);
61 }
62
63 //我参与的社群
64 public function myCommunities(Request $request)
65 {
66     $user_id = $request->get('user_id');
67     if (empty($user_id)) {
68         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
69     }
70     $limit = $request->get('limit', 10);
71     return Community::myCommunities($user_id, $limit);
72 }
73
74 //我参与的全部社群
75 public function myCommunitiesAll(Request $request)
76 {
77     $user_id = $request->get('user_id');
78     if (empty($user_id)) {
79         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
80     }
81     return Community::myCommunitiesAll($user_id);
82 }
83
84 //退出社群
85 public function quit(Request $request)
86 {
87     $user_id = $request->post('user_id');
88     if (empty($user_id)) {
89         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
90     }
91     $community_id = $request->post('community_id');
92     if (empty($community_id)) {
93         return ApiReturn::rdata(CodeMessage::common(30001));
94     }
95     return Community::quit($user_id, $community_id);
96 }
97
98 //解散群聊
99 public function dissolve(Request $request)

```

```
49 {
50     $user_id = $request->post('user_id');
1   if (empty($user_id)) {
2       return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
3   }
4   $community_id = $request->post('community_id');
5   if (empty($community_id)) {
6       return ApiReturn::rdata(CodeMessage::common(30001));
7   }
8   return Community::dissolve($user_id, $community_id);
9 }
10
11 //获取不受限制小程序分享海报
12 public function getWechatMiniProgramSharePoster(Request $request)
13 {
14     $param = $request->all();
15     if (empty($param['user_id'])) {
16         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
17     }
18     if (empty($param['community_id'])) {
19         return ApiReturn::rdata(CodeMessage::common(30001));
20     }
21     if (empty($param['page'])) {
22         return ApiReturn::rdata(CodeMessage::common(30001));
23     }
24     return Community::getWechatMiniProgramSharePoster($param);
25 }
26 //踢出群员
27 public function kickMember(Request $request) {
28     //群主Id
29     $user_id = $request->post('user_id');
30     if (empty($user_id)) {
31         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
32     }
33     //社群ID
34     $community_id = $request->post('community_id');
35     if (empty($community_id)) {
36         return ApiReturn::rdata(CodeMessage::common(30001));
37     }
38     //成员ID
39     $member_id = $request->post('member_id');
40     if (empty($member_id)) {
41         return ApiReturn::rdata(CodeMessage::common(30001));
42     }
43     //检测要被踢出成员是否为当前账号
44     if($user_id == $member_id){
45         return ApiReturn::error('您不能将自己退出群聊');
46     }
47     return Community::kickMember($user_id,$community_id,$member_id);
48 }
49 }
50
```

```

2
3 namespace App\Http\Controllers\api\v1;
4
5 use App\Http\Controllers\Controller;
6 use App\Service\v1\ApiReturn;
7 use App\Service\v1\Common;
8 use App\Service\v1\RedisKey;
9 use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Redis;
11 use Predis\Client;
12
13 class ConversionExplainController extends Controller
14 {
15     /**
16      * 积分商城兑换说明
17      * @return Object
18      */
19     public function conversionExplain()
20     {
21         try {
22             $redisConf = config('database.redis.default');
23             $client = new Client($redisConf);
24             $info = $client->get(env('SOLO_ADMIN_REDIS_PREFIX').RedisKey::CONVERSION_EXPLAIN);
25             return ApiReturn::success(['explain' => $info]);
26         } catch (\Throwable $throwable) {
27             Common::log('积分商城兑换说明',$throwable);
28             return ApiReturn::error();
29         }
30     }
31 }
32

```

```

1 <?php
2
3
4 namespace App\Http\Controllers\api\v1;
5
6 use App\Events\ShareArticleTaskEvent;
7 use App\Http\Controllers\Controller;
8 use App\Models\v1\ChatLog;
9 use App\Models\v1\User;
10 use App\Models\v1\UserFanBase;
11 use App\Service\v1\ApiCode;
12 use App\Service\v1\ApiReturn;
13 use App\Service\v1\Common;
14 use App\Service\v1\FanBase;
15 use App\Service\v1\RedisKey;
16 use App\Service\v1\task\JoinFanBaseTask;
17 use App\Service\v1\task\LogoutFanBaseTask;
18 use App\Service\v1\task\ShareArticleTask;
19 use App\Service\v1\task\ShareImageTask;
20 use Hhxsv5\LaravelS\Swoole\Task\Task;
21 use Illuminate\Http\Request;
22 use Illuminate\Support\Facades\Redis;

```

```

23
24 class FanBaseController extends Controller
25 {
26     /**
27      * 聊天记录
28      * @param Request $request
29      * @return Object
30      */
31     public function getChatLogs(Request $request)
32     {
33         $fanBaseId = intval($request->get('fan_base_id',0));
34         $chatLogId = intval($request->get('chat_log_id',0));
35         if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
36         $userId = $request->get('user_id');
37         try {
38             $startTime = UserFanBase::where(['user_id' => $userId,'fan_base_id' => $fanBaseId,'status' =>
11)->value('updated_at');
39             if(empty($startTime)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
40             $page = $request->get('page',1);
41             $limit = $request->get('limit',20);
42             $page = ($page - 1) * $limit;
43             $where = [
44                 ['fan_base_id','<=', $fanBaseId],
45                 ['created_at','>=', $startTime]
46             ];
47             if($chatLogId > 0) $where[] = ['id','<', $chatLogId];
48             $chatLogs = (new ChatLog())->getChatLogs($where,$page,$limit);
49             return ApiReturn::success($chatLogs);
50         }catch (\Throwable $throwable) {

1         Common::log('获取聊天记录',$throwable);
2         return ApiReturn::error();
3     }
4 }
5
6     /**
7      * 群成员
8      * @param Request $request
9      * @return Object
10     */
11     public function getFanBaseMembers(Request $request)
12     {
13         $fanBaseId = intval($request->get('fan_base_id',0));
14         if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
15         $page = intval($request->get('page',1));
16         $limit = intval($request->get('limit',20));
17         $page = ($page - 1) * $limit;
18         try {
19             $users = (new UserFanBase())->getFanBaseUsers($fanBaseId,$page,$limit);
20             return ApiReturn::success($users);
21         }catch (\Throwable $throwable) {
22             Common::log('获取聊天记录',$throwable);
23             return ApiReturn::error();

```



```

24     }
25 }
26
27
28 /**
29  * 退群
30  * @param Request $request
31  * @return Object
32  */
33 public function logoutFanBase(Request $request)
34 {
35     $fanBaseId = intval($request->get('fan_base_id',0));
36     if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
37     $userId = $request->get('user_id');
38     try {
39         // $result = (new UserFanBase())->logoutFanBase($fanBaseId,$userId);
40         // if($result == 0) return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
41         $task = new LogoutFanBaseTask($userId,$fanBaseId);
42         $result = Task::deliver($task);
43         if($result) return ApiReturn::success();
44         return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
45     }catch (\Throwable $throwable) {
46         Common::log('退群',$throwable);
47         return ApiReturn::error();
48     }
49 }
50

```

```

1  /**
2  * 获取我的群列表
3  * @param Request $request
4  * @return Object
5  */
6  public function fanBases(Request $request)
7  {
8      try {
9          $userId = $request->get('user_id');
10         $page = $request->get('page',1);
11         $limit = $request->get('limit',10);
12         $page = ($page - 1) * $limit;
13         $fanBases = FanBase::fanBases($userId,$page,$limit);
14         return ApiReturn::success($fanBases);
15     }catch (\Throwable $throwable) {
16         Common::log('获取我的群列表',$throwable);
17         return ApiReturn::error();
18     }
19 }
20
21 /**
22 * 分享到群到群列表
23 * @param Request $request
24 * @return Object
25 */
26 public function shareFanBases(Request $request)

```

```

27     {
28         try {
29             $userId = $request->get('user_id');
30             $page = $request->get('page',1);
31             $limit = $request->get('limit',20);
32             $page = ($page - 1) * $limit;
33             $fanBases = FanBase::shareFanBases($userId,$page,$limit);
34             return ApiReturn::success($fanBases);
35         }catch (\Throwable $throwable) {
36             Common::log('分享到群到群列表',$throwable);
37             return ApiReturn::error();
38         }
39     }
40
41     /**
42     * 加入群聊
43     * @param Request $request
44     * @return Object
45     */
46     public function joinFanBase(Request $request)
47     {
48         try {
49             $fanBaseId = intval($request->get('fan_base_id',0));
50             if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
51
52             $userId = $request->get('user_id');
53             $redisNx = boolval(Redis::set(RedisKey::USER_JOIN_FAN_BASE_STATUS.$userId,1,'EX',2,'NX'));
54             if($redisNx) {
55                 $status = FanBase::isFans($fanBaseId,$userId);
56                 if(empty($status)) {
57                     $task = new JoinFanBaseTask($userId,$fanBaseId);
58                     $result = Task::deliver($task);
59                     if($result) return ApiReturn::success();
60                 }else{
61                     return ApiReturn::success();
62                 }
63             }
64             return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
65         }catch (\Throwable $throwable) {
66             Common::log('加入群聊',$throwable);
67             return ApiReturn::error();
68         }
69     }
70
71     /**
72     * 分享群页面
73     * @param Request $request
74     * @return Object
75     */
76     public function sharePage(Request $request)
77     {
78         try {
79             $fanBaseId = intval($request->get('fan_base_id',0));
80             if($fanBaseId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);

```

```
30     $userId = $request->get('user_id');
31     $re = FanBase::sharePage($userId,$fanBaseId);
32     return ApiReturn::success($re);
33 }catch (\Throwable $throwable) {
34     Common::log('分享群页面',$throwable);
35     return ApiReturn::error();
36 }
37 }
38
39 /**
40  * pc分享群页面
41  * @param Request $request
42  * @return Object
43  */
44 public function pcSharePage(Request $request)
45 {
46     try {
47         $fanBaseId = intval($request->get('fan_base_id',0));
48         $userId = intval($request->get('share_user_id',0));
49         if($fanBaseId <= 0 || $userId <= 0) return
50             ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
51         $re = FanBase::sharePage($userId,$fanBaseId);
52
53         return ApiReturn::success($re);
54     }catch (\Throwable $throwable) {
55         Common::log('pc分享群页面',$throwable);
56         return ApiReturn::error();
57     }
58 }
59
60 /**
61  * 分享文章到群
62  * @param Request $request
63  * @return Object
64  */
65 public function shareArticleToFanBase(Request $request)
66 {
67     try {
68         $fanBaseId = (int)$request->post('fan_base_id',0);
69         $articleId = (int)$request->post('article_id',0);
70         if($fanBaseId <= 0 || $articleId <= 0) return
71             ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
72         $userId = $request->post('user_id');
73         $content = (string)$request->post('content','');
74         $task = new ShareArticleTask($userId,$fanBaseId,$articleId,$content);
75         $result = Task::deliver($task);
76         if($result) {
77             ShareArticleTaskEvent::dispatch($userId,$articleId);
78             return ApiReturn::success();
79         }
80         return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
81     }catch (\Throwable $throwable) {
82         Common::log('分享文章到群错误',$throwable);
83         return ApiReturn::error();
84     }
```

```

31     }
32 }
33
34 /**
35  * 分享图片到群聊
36  * @param Request $request
37  * @return Object
38  */
39 public function shareImageToFanBase(Request $request)
40 {
41     try {
42         $fanBaseId = (int)$request->post('fan_base_id',0);
43         $image = (string)$request->post('image','');
44         if($fanBaseId <= 0 || empty($image)) return
45         ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
46         $userId = $request->post('user_id');
47         $task = new ShareImageTask($userId,$fanBaseId,$image);
48         $result = Task::deliver($task);
49         if($result) {
50             return ApiReturn::success();
51         }
52     } catch (\Throwable $throwable) {
53         Common::log('分享文章到群错误',$throwable);
54         return ApiReturn::error();
55     }
56 }
57
58 /**
59  * 设置下线时间
60  * @param Request $request
61  * @return Object
62  */
63 public function setOffLineTime(Request $request)
64 {
65     $userId = $request->get('user_id');
66     try {
67         User::where(['id' => $userId])>->update(['off_line_time' => $request->server('REQUEST_TIME')]);
68         return ApiReturn::success();
69     } catch (\Throwable $throwable) {
70         Common::log('设置下线时间错误',$throwable);
71         return ApiReturn::error();
72     }
73 }
74
75 /**
76  * 任务中心分享群聊
77  * @param Request $request
78  * @return Object
79  */
80 public function shareFanBase(Request $request)

```

```

33     {
34         try {
35             $fanBaseId = intval($request->post('fan_base_id',0));
36             if($fanBaseId >= 0) {
37                 $userId = $request->post('user_id');
38                 $count = \App\Models\v1\FanBase::where(['id' => $fanBaseId])->count();
39                 if($count > 0) {
40                     Redis::eval(<<<'LUA'
41                         redis.call("INCR",KEYS[1])
42                         if redis.call("TTL",KEYS[1]) < 0 then
43                             redis.call("EXPIREAT",KEYS[1],ARGV[1])
44                         end
45                         LUA, 1, RedisKey::SYSTEM_TASK Everyday_SHARE_GROUP.$userId,strtotime('tomorrow'));
46                     }
47                 }
48                 return ApiReturn::success();
49             }catch (\Throwable $throwable) {
50                 Common::log('任务中心分享群聊',$throwable);
51             }
52         }
53     }
54 }
55
56 /**
57  * 群主创建的群列表
58  * @param Request $request
59  * @return Object
60 */
61 public function groupOwnerFanBases(Request $request)
62 {
63     $userId = $request->get('user_id');
64     $groupOwnerId = (int)$request->get('group_owner_id',0);
65     if($groupOwnerId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
66     try {
67         $fanBases = FanBase::groupOwnerFanBases($userId,$groupOwnerId);
68         return ApiReturn::success($fanBases);
69     }catch (\Throwable $throwable) {
70         Common::log('群主创建的群列表',$throwable);
71         return ApiReturn::error();
72     }
73 }
74
75 }
76
77 }
78
79 }
80
81 }
82
83 }
84
85 }
86
87 }
88
89 }
90
91 }
92
93 }
94
95 }
96
97 }
98
99 }
100
101 }
102
103 }
104
105 }
106
107 }
108
109 }
110
111 }
112
113 }
114
115 }
116
117 }
118
119 }
120
121 }
122
123 }
124
125 }
126
127 }
128
129 }
130
131 }
132
133 }
134
135 }
136
137 }
138
139 }
140
141 }
142
143 }
144
145 }
146
147 }
148
149 }
150
151 }
152
153 }
154
155 }
156
157 }
158
159 }
160
161 }
162
163 }
164
165 }
166
167 }
168
169 }
170
171 }
172
173 }
174
175 }
176
177 }
178
179 }
180
181 }
182
183 }
184
185 }
186
187 }
188
189 }
190
191 }
192
193 }
194
195 }
196
197 }
198
199 }
200
201 }
202
203 }
204
205 }
206
207 }
208
209 }
210
211 }
212
213 }
214
215 }
216
217 }
218
219 }
220
221 }
222
223 }
224
225 }
226
227 }
228
229 }
230
231 }
232
233 }
234
235 }
236
237 }
238
239 }
240
241 }
242
243 }
244
245 }
246
247 }
248
249 }
250
251 }
252
253 }
254
255 }
256
257 }
258
259 }
260
261 }
262
263 }
264
265 }
266
267 }
268
269 }
270
271 }
272
273 }
274
275 }
276
277 }
278
279 }
280
281 }
282
283 }
284
285 }
286
287 }
288
289 }
290
291 }
292
293 }
294
295 }
296
297 }
298
299 }
300
301 }
302
303 }
304
305 }
306
307 }
308
309 }
310
311 }
312
313 }
314
315 }
316
317 }
318
319 }
320
321 }
322
323 }
324
325 }
326
327 }
328
329 }
330
331 }
332
333 }
334
335 }
336
337 }
338
339 }
340
341 }
342
343 }
344
345 }
346
347 }
348
349 }
350
351 }
352
353 }
354
355 }
356
357 }
358
359 }
360
361 }
362
363 }
364
365 }
366
367 }
368
369 }
370
371 }
372
373 }
374
375 }
376
377 }
378
379 }
380
381 }
382
383 }
384
385 }
386
387 }
388
389 }
390
391 }
392
393 }
394
395 }
396
397 }
398
399 }
400
401 }
402
403 }
404
405 }
406
407 }
408
409 }
410
411 }
412
413 }
414
415 }
416
417 }
418
419 }
420
421 }
422
423 }
424
425 }
426
427 }
428
429 }
430
431 }
432
433 }
434
435 }
436
437 }
438
439 }
440
441 }
442
443 }
444
445 }
446
447 }
448
449 }
450
451 }
452
453 }
454
455 }
456
457 }
458
459 }
460
461 }
462
463 }
464
465 }
466
467 }
468
469 }
470
471 }
472
473 }
474
475 }
476
477 }
478
479 }
480
481 }
482
483 }
484
485 }
486
487 }
488
489 }
490
491 }
492
493 }
494
495 }
496
497 }
498
499 }
500
501 }
502
503 }
504
505 }
506
507 }
508
509 }
510
511 }
512
513 }
514
515 }
516
517 }
518
519 }
520
521 }
522
523 }
524
525 }
526
527 }
528
529 }
530
531 }
532
533 }
534
535 }
536
537 }
538
539 }
540
541 }
542
543 }
544
545 }
546
547 }
548
549 }
550
551 }
552
553 }
554
555 }
556
557 }
558
559 }
560
561 }
562
563 }
564
565 }
566
567 }
568
569 }
570
571 }
572
573 }
574
575 }
576
577 }
578
579 }
580
581 }
582
583 }
584
585 }
586
587 }
588
589 }
590
591 }
592
593 }
594
595 }
596
597 }
598
599 }
600
601 }
602
603 }
604
605 }
606
607 }
608
609 }
610
611 }
612
613 }
614
615 }
616
617 }
618
619 }
620
621 }
622
623 }
624
625 }
626
627 }
628
629 }
630
631 }
632
633 }
634
635 }
636
637 }
638
639 }
640
641 }
642
643 }
644
645 }
646
647 }
648
649 }
650
651 }
652
653 }
654
655 }
656
657 }
658
659 }
660
661 }
662
663 }
664
665 }
666
667 }
668
669 }
670
671 }
672
673 }
674
675 }
676
677 }
678
679 }
680
681 }
682
683 }
684
685 }
686
687 }
688
689 }
690
691 }
692
693 }
694
695 }
696
697 }
698
699 }
699
699 }

```

```

10 use App\Service\v1\ApiCode;
11 use App\Service\v1\ApiReturn;
12 use App\Service\v1\Common;
13 use App\Service\v1\CommonFnc;
14 use App\Service\v1\Game;
15 use App\Service\v1\ProductService;
16 use Illuminate\Http\Request;
17 use Illuminate\Support\Facades\Log;
18
19 class GameController extends Controller
20 {
21
22     /**
23      * 我的游戏页/ios发布动态页--搜索游戏
24      * @param Request $request
25      * @return Object
26      */
27     public function searchGames(Request $request)
28     {
29         try {
30             $gameName = $request->get('game_name');
31             if (empty($gameName)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
32             $userId = $request->get('user_id');
33             $page = $request->get('page', 1);
34             $limit = $request->get('limit', 20);
35             $games = Game::searchGames($userId, $gameName, $page, $limit);
36             GameSearchEvent::dispatch($gameName, $userId, $request->get('device_num', '') ?? '');
37             return ApiReturn::success($games);
38         } catch (\Throwable $throwable) {
39             Common::log('搜索游戏错误', $throwable);
40             return ApiReturn::error();
41         }
42     }
43
44     /**
45      * 我的游戏
46      * @param Request $request
47      * @return Object
48      */
49     public function myGames(Request $request)
50     {
51         try {
52             $userId = $request->get('user_id');
53             $page = $request->get('page', 1);
54             $limit = $request->get('limit', 10);
55             $page = ($page - 1) * $limit;
56             $games = Game::myGames($userId, $page, $limit);
57             return ApiReturn::success($games);
58         } catch (\Throwable $throwable) {
59             Common::log('我的游戏', $throwable);
60             return ApiReturn::error();
61         }
62     }

```

```

13
14  /**
15   * 加入我的游戏
16   * @param Request $request
17   * @return Object
18   */
19   public function addMyGame(Request $request)
20   {
21       try {
22           $userId = $request->post('user_id');
23           $gameIds = $request->post('game_ids', []);
24           if (empty($gameIds) || !is_array($gameIds)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
25           foreach ($gameIds as $gameId) {
26               UserGame::updateOrCreate([
27                   'user_id' => $userId,
28                   'game_id' => $gameId
29               ], ['status' => 1]);
30           }
31           return ApiReturn::success();
32       } catch (\Throwable $throwable) {
33           Common::log('加入我的游戏', $throwable);
34           return ApiReturn::error();
35       }
36   }
37
38
39  /**
40   * 删除我的游戏
41   * @param Request $request
42   * @return Object
43   */
44   public function delMyGame(Request $request)
45   {
46       try {
47           $userId = $request->post('user_id');
48           $gameIds = $request->post('game_ids', []);
49           if (empty($gameIds) || !is_array($gameIds)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
ApiCode::PARAM_ERROR['code']);
50           UserGame::where(['user_id' => $userId, 'status' => 1])->whereIn('game_id', $gameIds)->update(['status' => 0]);
51
52           return ApiReturn::success();
53       } catch (\Throwable $throwable) {
54           Common::log('删除我的游戏', $throwable);
55           return ApiReturn::error();
56       }
57   }
58
59  /**
60   * 玩过的游戏（临时解决方案）
61   * @param Request $request
62   * @return Object
63   */
64   public function playedGames(Request $request)

```

```

14     {
15         try {
16             $userId = $request->post('user_id');
17             $games = Game::playedGames($userId);
18             return ApiReturn::success($games);
19         } catch (\Throwable $throwable) {
20             Common::log('玩过的游戏', $throwable);
21             return ApiReturn::error();
22         }
23     }
24
25     //pc首页排行榜
26     public function pcHomeRanking(): object
27     {
28         return Game::pcHomeRanking();
29     }
30
31     //相关游戏
32     public function related(Request $request)
33     {
34         $game_id = $request->get('game_id', 0);
35         return Game::related($game_id);
36     }
37
38     /**
39      * 获取下载链接并新增下载记录
40      * @param Request $request
41      * @return object
42      */
43     public function getDownLoadUrl(Request $request): object
44     {
45         $game_id = $request->get('game_id');
46         if (empty($game_id)) {
47             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
48         }
49         $user_id = $request->get('user_id');
50         if (empty($user_id)) {
51
52             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
53         }
54
55         //设备信息
56         $platform = $request->header('Platform', 1);
57         return Game::getDownloadUrl($user_id, $game_id, $platform);
58     }
59
60     //游戏推荐列表
61     public function recommend(Request $request): object
62     {
63         $user_id = $request->get('user_id');
64         $page = $request->get('page', 1);
65         $limit = $request->get('limit', 10);
66         return Game::recommendList($user_id, $page, $limit);
67     }
68

```



```

17 //游戏详情
18 public function detail(Request $request): object
19 {
20     $lang = CommonFnc::lang();
21     $game_id = $request->get('game_id');
22     if (empty($game_id)) {
23         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
24     }
25     $user_id = $request->get('user_id');
26     return Game::detail($game_id, $user_id,$lang);
27 }
28
29 //排行榜
30 public function ranking(Request $request)
31 {
32     Log::error('排行榜', $request->all());
33     $id = $request->get('id', 1);
34     $page = $request->get('page', 1);
35     $limit = $request->get('limit', 10);
36     if ($page * $limit > 150) {
37         return ApiReturn::success();
38     }
39     $leardboard = Leaderboard::query()->select(['id', 'name', 'cover', 'lang'])->find($id);
40     if (!$leardboard) {
41         return ApiReturn::success();
42     }
43     $leardboard->cover = CommonFnc::concatUrlStr($leardboard->cover);
44     $leaderboardLang = $leardboard->lang;
45     $list = $leardboard->games()->with([
46         'circle' => function ($query) {
47             $query->select(['id', 'game_id', 'circle_name', 'cover']);
48         },
49         'game_translation' => function ($query) use ($leaderboardLang) {
50             $query->where('lang', $leaderboardLang);
51
52             },
53         'game_labels' => function ($query) use ($leaderboardLang) {
54             $query->where('lang', $leaderboardLang);
55         }
56     ])->paginate($limit);
57     $list = isset($list) ? $list->toArray() : [];
58     $list['leardboard'] = $leardboard;
59     CommonFnc::paginateReturnUnset($list);
60     foreach ($list['data'] as $k => $v) {
61         $game = [];
62         $game['position'] = ($page - 1) * $limit + $k + 1;
63         $game['id'] = $v['id'];
64         $game['score'] = $v['score'];
65         $game['name'] = empty($v['game_translation']['game_name']) ? $v['game_name'] :
66             $v['game_translation']['game_name'];
67         $game['icon'] = CommonFnc::concatUrlStrAdmin($v['game_icon']);
68         $game['circle_id'] = $v['circle']['id'] ?? 0;
69         $game['circle'] = $v['circle'];
70         $game['label_txt'] = implode(',', array_slice(array_column($v['game_labels'], 'label_name'), 0, 3));

```

```

19         $game['rec_text'] = $v['game_translation']['rec_text'] ?? '';
20         $list['data'][$k] = $game;
21     }
22     return ApiReturn::success($list);
23 }
24
25 //榜单设置
26 public function leaderboard(Request $request)
27 {
28     $user_id = $request->get('user_id');
29     return Game::leaderboard($user_id);
30 }
31
32 //今日游戏
33 public function calendar(Request $request)
34 {
35     $page = $request->get('page', 0);
36     $user_id = $request->get('user_id');
37     return Game::calendar($page, $user_id);
38 }
39
40 //游戏日历更多游戏
41 public function calendarMore(Request $request)
42 {
43     $page = $request->get('page', 0);
44     $user_id = $request->get('user_id');
45     return Game::calendarMore($page, $user_id);
46 }
47
48 /**
49  * 热门游戏
50  * @return Object

```

```

1  */
2  public function getHotGame(): object
3  {
4      try {
5          $hotGame = ServiceConfig::query()->where(['field_name' => 'hot_recommend'])->value('value');
6          $hotGame = explode('$$', $hotGame);
7          $arr = [];
8          foreach ($hotGame as $k => $v) {
9              $arr[] = [
10                 'sort' => $k + 1,
11                 'game_name' => $v
12             ];
13         }
14         return ApiReturn::success($arr);
15     } catch (\Exception $e) {
16         return ApiReturn::error();
17     }
18 }
19
20 //游戏列表
21 public function list(Request $request)

```

```

22  {
23      $limit = $request->get('limit', 10);
24      $label = $request->get('label');
25      $user_id = $request->get('user_id');
26      $category = $request->get('category');
27      $order = $request->get('order', 'time');
28      $lang = CommonFnc::lang();
29      if($request->get('version_code_num') >= 339){
30          return Game::newList($limit, $label, $user_id, $category, $order, $lang);
31      }else{
32          return Game::list($limit, $label, $user_id, $category, $order, $lang);
33      }
34  }
35
36  //关注游戏
37  public function attention(Request $request)
38  {
39      $user_id = $request->get('user_id');
40      if (empty($user_id)) {
41          return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
42      }
43      $game_id = $request->get('game_id');
44      if (empty($game_id)) {
45          return ApiReturn::error('请选择要关注的游戏');
46      }
47      return Game::attention($user_id, $game_id);
48  }
49
50  //游戏筛选选择集

```

```

1  public function filterSelects()
2  {
3      return Game::filterSelects();
4  }
5
6  //预约
7  public function reserve(Request $request)
8  {
9      $user_id = $request->get('user_id');
10     if (empty($user_id)) {
11         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
12     }
13     $game_id = $request->get('game_id');
14     if (empty($game_id)) {
15         return ApiReturn::error('请选择要预约的游戏');
16     }
17     $status = $request->get('status', 1);
18     if (!in_array($status, [1, 2])) {
19         return ApiReturn::error('预约状态错误');
20     }
21     return Game::reserve($user_id, $game_id, $status);
22 }
23
24 //校验游戏是否存在

```

```
25     public function check(Request $request)
26     {
27         $game_id = $request->get('game_id');
28         if (empty($game_id)) {
29             return ApiReturn::error('游戏不存在');
30         }
31         return Game::check($game_id);
32     }
33
34     //游戏商品详情
35     public function productList(Request $request)
36     {
37         //游戏ID
38         $gameId = $request->get('game_id',0);
39         if(empty($gameId)){
40             return ApiReturn::error('请选择要查看的游戏商品');
41         }
42         $list = Game::productList($gameId);
43         return ApiReturn::success($list);
44     }
45     //热门免费游戏列表
46     public function hotFreeList(Request $request)
47     {
48         $limit = $request->get('limit', 10);
49         $lang = CommonFnc::lang();
50         $list = Game::hotFreeList($limit,$lang);
51
52         return ApiReturn::success($list);
53     }
54
55     //特惠栏
56     public function specialOffer(Request $request)
57     {
58         $limit = $request->get('limit', 10);
59         $type = $request->get('type', 'all');
60         $lang = CommonFnc::lang();
61         $version = $request->get('version_code_num');
62         $list = Game::specialOffer($limit,$lang,$type,$version);
63         return ApiReturn::success($list);
64     }
65
66     //特惠栏筛选标签集
67     public function specialOfferFilterSelects()
68     {
69         $list = Game::specialOfferFilterSelects();
70         return ApiReturn::success($list);
71     }
72
73     //限时免费游戏
74     public function getFreeTemporary(Request $request)
75     {
76         $limit = $request->get('limit', 10);
77         $lang = CommonFnc::lang();
78         $user_id = $request->get('user_id');
79         $list = Game::getFreeTemporary($limit,$user_id,$lang);
```

```

28     return ApiReturn::success($list);
29 }
30 //添加心愿单
31 public function addWishlist(Request $request)
32 {
33     $user_id = $request->get('user_id');
34     if (empty($user_id)) {
35         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
36     }
37     $game_id = $request->get('game_id');
38     if (empty($game_id)) {
39         return ApiReturn::error('请选择要添加的游戏');
40     }
41     $type = $request->get('type', 'steam');
42     return Game::addWishlist($user_id,$game_id,$type);
43 }
44 //移除心愿单
45 public function removeWishlist(Request $request)
46 {
47     $user_id = $request->get('user_id');
48     if (empty($user_id)) {
49         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
50     }
51 }
52
53 $game_id = $request->get('game_id');
54 if (empty($game_id)) {
55     return ApiReturn::error('请选择要移除的心愿单游戏');
56 }
57 $type = $request->get('type', 'steam');
58 return Game::removeWishlist($user_id,$game_id,$type);
59 }
60 //心愿单列表
61 public function wishlist(Request $request)
62 {
63     $user_id = $request->get('user_id');
64     if (empty($user_id)) {
65         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
66     }
67     $solo_userid = $request->get('solo_userid');
68
69     $limit = $request->get('limit', 10);
70     $lang = CommonFnc::lang();
71     $type = $request->get('type', 'all');
72     $list = Game::wishlist($user_id,$lang,$limit,$type,$solo_userid);
73     return ApiReturn::success($list);
74 }
75 }
76
77 <?php
78
79 namespace App\Http\Controllers\api\v1;
80
81 use App\Http\Controllers\Controller;
82 use App\Models\v1\GameLabel;

```

```
7 use App\Models\v1\UserGameLabel;
8 use App\Service\v1\ApiCode;
9 use App\Service\v1\ApiReturn;
10 use App\Service\v1\Common;
11 use App\Service\v1\CommonFnc;
12 use App\Service\v1\Game;
13 use Illuminate\Http\Request;
14 use Illuminate\Support\Facades\Log;
15
16 class GameController extends Controller
17 {
18     /**
19      * 游戏推荐标签
20      * @return Object
21      */
22     public function recommendLabels(Request $request)
23     {
24         $lang = CommonFnc::lang();
25         try {
26             $type = $request->get('type',1);
27             $recommendLabels = GameLabel::where(['is_recommend' => 1])
28                 ->where('lang',$lang)
29                 ->select('id as label_id','label_name')
30                 ->get()
31                 ->toArray();
32             if($type == 2){
33                 $all = '全部';
34                 if($lang == 2){
35                     $all = 'All';
36                 }elseif($lang == 3){
37                     //日文
38                     $all = 'すべて';
39                 }
40                 //在数组增加 全部
41                 array_unshift($recommendLabels,['label_id' => 0,'label_name' => $all]);
42             }
43             if($request->header('platform') == 2){
44                 return ApiReturn::success(['data' => $recommendLabels]);
45             }else{
46                 return ApiReturn::success( $recommendLabels);
47             }
48         }catch (\Throwable $throwable) {
49             Common::log('游戏推荐标签',$throwable);
50             return ApiReturn::error();
51         }
52     }
53
54     /**
55      * 用户喜欢的游戏标签
56      * @param Request $request
57      * @return Object
58      */
59     public function likeLabels(Request $request)
```

```

10     {
11         try {
12             $userId = $request->get('user_id');
13             $labels = Game::likeLabels($userId);
14             return ApiReturn::success($labels);
15         } catch (\Throwable $throwable) {
16             Common::log('用户喜欢的游戏标签',$throwable);
17             return ApiReturn::error();
18         }
19     }
20
21     /**
22      * 我的游戏--用户标签编辑
23      * @param Request $request
24      * @return Object
25      */
26     public function addLikeLabel(Request $request)
27     {
28         try {
29             $userId = $request->post('user_id');
30             $gameLabelIds = $request->post('game_label_ids',[]);
31             // if(empty($gameLabelIds) || !is_array($gameLabelIds)) return
32             ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
33             $oldGameLabelIds = UserGameLabel::where(['user_id' => $userId,'status' => 1])->pluck('game_label_id')->toArray();
34             $del = array_diff($oldGameLabelIds,$gameLabelIds);
35             $add = array_diff($gameLabelIds,$oldGameLabelIds);
36             if(!empty($del)) {
37                 UserGameLabel::whereIn('game_label_id',$del)->where(['user_id' => $userId])->update(['status' => 0]);
38             }
39             if(!empty($add)) {
40                 foreach ($add as $v) {
41                     if(empty($v)){
42                         continue;
43                     }
44                     UserGameLabel::updateOrCreate([
45                         'user_id' => $userId,
46                         'game_label_id' => $v
47                     ],['status' => 1]);
48                 }
49             }
50             return ApiReturn::success();
51         } catch (\Throwable $throwable) {
52
53             Common::log('我的游戏--用户标签编辑',$throwable);
54             return ApiReturn::error();
55         }
56     }
57 }
58
59 }
60
61 }
62
63 }
64
65 }
66
67 }
68
69 }
70
71 }
72
73 }
74
75 }
76
77 }
78
79 }
80
81 }
82
83 }
84
85 }
86
87 }
88
89 }
90
91 }
92
93 }
94
95 }
96
97 }
98
99 }
100

```

```

1 <?php
2
3 namespace App\Http\Controllers\api\v1;
4

```

```

5  use App\Http\Controllers\Controller;
6  use App\Service\v1\ApiCode;
7  use App\Service\v1\ApiReturn;
8  use App\Service\v1\Information;
9  use Illuminate\Http\Request;
10
11  class InformationController extends Controller
12  {
13      //头条
14      public function headlines(Request $request): object
15      {
16          $id = $request->get('id',0);
17          return Information::headlines($id);
18      }
19      //推荐
20      public function recommended(Request $request): object
21      {
22          $catId = $request->get('cat_id',0);
23          $id = $request->get('id',0);
24          return Information::recommended($catId,$id);
25      }
26      //列表
27      public function list(Request $request): object
28      {
29          $page = $request->get('page',1);
30          $limit = $request->get('limit',2);
31          $catId = $request->get('cat_id',0);
32          $use_default_img = $request->get('use_default_img',1);
33          $id = $request->get('id',0);
34          return Information::list($limit,$catId,$use_default_img,$id);
35      }
36      //详情
37      public function info(Request $request): object
38      {
39          $id = $request->get('id');
40          if(empty($id)){
41              return ApiReturn::rdata(ApiCode::PARAM_ERROR);
42          }
43          $user_id = $request->get('user_id');
44          return Information::info($id,$user_id);
45      }
46      public function catList(): object
47      {
48          return Information::catList();
49      }
50      //评论

```

```

1  public function addComment(Request $request){
2      $commentable_id = $request->post('commentable_id');
3      if(empty($commentable_id)){
4          return ApiReturn::rdata(ApiCode::PARAM_ERROR);
5      }
6      $content = $request->post('content');
7      if(empty($content)){

```



```

8         return ApiReturn::error('请输入内容');
9     }
10    $user_id = $request->post('user_id');
11    return Information::addComment($user_id,$commentable_id,$content);
12 }
13 //回复
14 public function addReply(Request $request){
15     $commentable_id = $request->post('commentable_id');
16     if(empty($commentable_id)){
17         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
18     }
19     $content = $request->post('content');
20     if(empty($content)){
21         return ApiReturn::error('请输入内容');
22     }
23     $user_id = $request->post('user_id');
24     return Information::addReply($user_id,$commentable_id,$content);
25 }
26 //评论列表
27 public function commentList(Request $request): object
28 {
29     $information_id = $request->get('information_id');
30     if(empty($information_id)){
31         return ApiReturn::success();
32     }
33     $limit = $request->get('limit',10);
34     return Information::commentList($information_id,$limit);
35 }
36 }
37 //点赞
38 public function like(Request $request){
39     $user_id = $request->post('user_id');
40     if(empty($user_id)){
41         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
42     }
43     $information_id = $request->post('information_id');
44     if(empty($information_id)){
45         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
46     }
47     return Information::like($user_id,$information_id);
48 }
49 //收藏
50 public function collect(Request $request){
51     $user_id = $request->post('user_id');
52     if(empty($user_id)){
53         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
54     }
55     $information_id = $request->post('information_id');
56     if(empty($information_id)){
57         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
58     }
59     return Information::collect($user_id,$information_id);
60 }

```

```

11 //分享
12 public function share(Request $request){
13     $user_id = $request->post('user_id');
14     if(empty($user_id)){
15         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
16     }
17     $information_id = $request->post('information_id');
18     if(empty($information_id)){
19         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
20     }
21     return Information::share($user_id,$information_id);
22 }
23 //回复列表
24 public function replyList(Request $request): object
25 {
26     $comment_id = $request->get('comment_id');
27     if(empty($comment_id)){
28         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
29     }
30     $limit = $request->get('limit',10);
31     return Information::replyList($comment_id,$limit);
32 }
33 //今日要闻
34 public function getTodayInformation(): object
35 {
36     return Information::getTodayInformation();
37 }
38 //新游预约
39 public function newGameReservation(): object
40 {
41     return Information::newGameReservation();
42 }
43 //行业资讯
44 public function getIndustry(): object
45 {
46     return Information::getIndustry();
47 }
48 }
49

```

```

1 <?php
2
3 namespace App\Http\Controllers\api\v1;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\v1\UserDiamondLog;
7 use App\Models\v1\UserInviteLog;
8 use App\Service\v1\ApiCode;
9 use App\Service\v1\ApiReturn;
10 use App\Service\v1\Common;
11 use App\Service\v1\CommonFnc;
12 use App\Service\v1\Invite;
13 use App\Service\v1\RedisKey;
14 use Illuminate\Http\Request;

```

```

15 use Illuminate\Support\Facades\Redis;
16
17 class InviteController extends Controller
18 {
19     /**
20      * 邀请页面接口
21      * @param Request $request
22      * @return Object
23      */
24     public function invitePage(Request $request)
25     {
26         $userId = $request->get('user_id');
27         try {
28             $invitePage = Invite::invitePage($userId);
29             return ApiReturn::success($invitePage);
30         } catch (\Throwable $throwable) {
31             Common::log('邀请页面接口', $throwable);
32             return ApiReturn::error();
33         }
34     }
35
36     /**
37      * 邀请页面接口
38      * @param Request $request
39      * @return Object
40      */
41     public function bindUser(Request $request)
42     {
43         $userId = $request->post('user_id');
44         $inviteCode = $request->post('invite_code');
45         if (!$inviteCode) return ApiReturn::rdata(ApiCode::PARAM_ERROR);
46         try {
47             $rs = Invite::bindUser($userId, $inviteCode);
48             if (!$rs['status']) return ApiReturn::rdata($rs['data']);
49             else return ApiReturn::success();
50         } catch (\Throwable $throwable) {
51
52             Common::log('绑定用户', $throwable);
53             return ApiReturn::error();
54         }
55     }
56
57     /**
58      * 好友贡献
59      * @param Request $request
60      * @return Object
61      */
62     public function inviteContributionInfo(Request $request)
63     {
64         $userId = $request->get('user_id');
65         try {
66             $data = [
67                 'people_total' => UserInviteLog::query()->where(['pid' => $userId])->count(),
68                 'diamond_total' => UserDiamondLog::query()->where(['user_id' => $userId, 'type' => 22])->sum('diamond')
69             ];
70         } catch (\Throwable $throwable) {
71
72             Common::log('好友贡献', $throwable);
73             return ApiReturn::error();
74         }
75     }
76 }

```

```
18     ];
19     return ApiReturn::success($data);
20 }catch (\Throwable $throwable) {
21     Common::log('好友贡献错误',$throwable);
22     return ApiReturn::error();
23 }
24 }
25
26
27 /**
28  * 佣金汇总
29  * @param Request $request
30  * @return Object
31  */
32 public function commissionLogs(Request $request)
33 {
34     $page = (int)$request->get('page',1);
35     $limit = (int)$request->get('limit',10);
36     $userId = $request->get('user_id');
37     try {
38         $page = ($page - 1) * $limit;
39         $commissionLogs = UserDiamondLog::query()->with(['user' => function($query) {
40             $query->select('id','username');
41             }])->where(['user_id' => $userId,'type' => 30])
42             ->select('diamond','describe','source_id')
43             ->limit($limit)->offset($page)->get();
44         return ApiReturn::success($commissionLogs);
45     }catch (\Throwable $throwable) {
46         Common::log('佣金汇总错误',$throwable);
47         return ApiReturn::error();
48     }
49 }
50
51
52 /**
53  * 邀请规则
54  * @return Object
55  */
56 public function inviteRule()
57 {
58     $awards = CommonFnc::getServiceConfigData(RedisKey::INVITE_CONFIG);
59     $rule = empty($awards) ? '' : json_decode($awards,true)['rule'];
60     return ApiReturn::success(['rule' => $rule]);
61 }
62
63 public function inviteBanner()
64 {
65     $arr = [
66         ['name' => "我***尚 提现",'money' => "707.59"],
67         ['name' => "安***远 提现",'money' => "505.03"],
68         ['name' => "曹***哥 提现",'money' => "323.38"],
69         ['name' => "浪***会 提现",'money' => "403.86"],
70         ['name' => "毋***山 提现",'money' => "350.71"],
71         ['name' => "心***了 提现",'money' => "508.53"],
```

21 ['name' => "彡***族 提现",'money' => "601.82"],
 22 ['name' => "你***唯 提现",'money' => "1101.9"],
 23 ['name' => "老***6 提现",'money' => "450.28"],
 24 ['name' => "去***卡 提现",'money' => "600.79"],
 25 ['name' => "勿***心 提现",'money' => "252.05"],
 26 ['name' => "剑***子 提现",'money' => "805.72"],
 27 ['name' => "无***美 提现",'money' => "414.34"],
 28 ['name' => "奋***小 提现",'money' => "453.31"],
 29 ['name' => "拾***睡 提现",'money' => "1061.4"],
 30 ['name' => "不***菜 提现",'money' => "303.59"],
 31 ['name' => "云***会 提现",'money' => "300.52"],
 32 ['name' => "清***清 提现",'money' => "506.18"],
 33 ['name' => "青***的 提现",'money' => "300.88"],
 34 ['name' => "我***字 提现",'money' => "351.01"],
 35 ['name' => "翔***作 提现",'money' => "404.17"],
 36 ['name' => "姓***兄 提现",'money' => "300.61"],
 37 ['name' => "挺***初 提现",'money' => "502.26"],
 38 ['name' => "睿***欢 提现",'money' => "402.83"],
 39 ['name' => "无***何 提现",'money' => "302.88"],
 40 ['name' => "肚***了 提现",'money' => "222.19"],
 41 ['name' => "我***逼 提现",'money' => "808.83"],
 42 ['name' => "江***月 提现",'money' => "702.41"],
 43 ['name' => "张***爸 提现",'money' => "504.42"],
 44 ['name' => "可***加 提现",'money' => "505.82"],
 45 ['name' => "春***了 提现",'money' => "703.07"],
 46 ['name' => "小***老 提现",'money' => "250.88"],
 47 ['name' => "伟***克 提现",'money' => "301.67"],
 48 ['name' => "初***猫 提现",'money' => "350.98"],
 49 ['name' => "琼***伯 提现",'money' => "252.42"],
 50 ['name' => "戴***特 提现",'money' => "402.23"],

1 ['name' => "家***钱 提现",'money' => "524.36"],
 2 ['name' => "你***坏 提现",'money' => "502.93"],
 3 ['name' => "大***江 提现",'money' => "363.73"],
 4 ['name' => "老***口 提现",'money' => "754.99"],
 5 ['name' => "小***1 提现",'money' => "553.92"],
 6 ['name' => "张***g 提现",'money' => "554.72"],
 7 ['name' => "禾***æ 提现",'money' => "454.03"],
 8 ['name' => "北***王 提现",'money' => "909.58"],
 9 ['name' => "心***白 提现",'money' => "252.78"],
 10 ['name' => "冷***会 提现",'money' => "853.75"],
 11 ['name' => "上***魔 提现",'money' => "500.04"],
 12 ['name' => "自***全 提现",'money' => "303.52"],
 13 ['name' => "小***★ 提现",'money' => "261.55"],
 14 ['name' => "丑***罗 提现",'money' => "504.37"],
 15 ['name' => "十***髅 提现",'money' => "301.08"],
 16 ['name' => "小***d 提现",'money' => "230.47"],
 17 ['name' => "远***英 提现",'money' => "301.37"],
 18 ['name' => "星***入 提现",'money' => "551.56"],
 19 ['name' => "仔***无 提现",'money' => "601.31"],
 20 ['name' => "一***闯 提现",'money' => "504.06"],
 21 ['name' => "火***木 提现",'money' => "404.59"],
 22 ['name' => "幻***的 提现",'money' => "210.71"],
 23 ['name' => "长***这 提现",'money' => "300.14"],

```

24      ['name' => "修***灵 提现",'money' => "510.37"],
25      ['name' => "张***的 提现",'money' => "503.91"],
26      ['name' => "陈***他 提现",'money' => "500.58"],
27      ['name' => "神***不 提现",'money' => "362.02"],
28      ['name' => "刘***解 提现",'money' => "650.61"],
29      ['name' => "暑***巅 提现",'money' => "700.17"],
30      ['name' => "一***唯 提现",'money' => "300.13"],
31      ['name' => "浅***倾 提现",'money' => "260.19"],
32      ['name' => "他***好 提现",'money' => "1007.3"],
33      ['name' => "飞***甲 提现",'money' => "273.19"],
34      ['name' => "王***耀 提现",'money' => "322.22"],
35      ['name' => "猜***是 提现",'money' => "805.15"],
36      ['name' => "瑶***静 提现",'money' => "252.33"],
37      ['name' => "小***加 提现",'money' => "301.12"],
38      ['name' => "陈***老 提现",'money' => "403.43"],
39      ['name' => "你***系 提现",'money' => "804.22"],
40      ['name' => "调***儿 提现",'money' => "301.04"],
41      ['name' => "了***红 提现",'money' => "450.84"],
42      ['name' => "鸣***永 提现",'money' => "504.13"],
43      ['name' => "小***一 提现",'money' => "302.55"],
44      ['name' => "咯***日 提现",'money' => "252.19"],
45      ['name' => "宇***带 提现",'money' => "302.34"],
46      ['name' => "今***开 提现",'money' => "500.25"],
47      ['name' => "闫***的 提现",'money' => "250.91"],
48      ['name' => "疯***里 提现",'money' => "252.78"],
49      ['name' => "实***3 提现",'money' => "312.02"],
50      ['name' => "哥***下 提现",'money' => "402.02"],

```

```

1      ['name' => "快***诞 提现",'money' => "302.04"],
2      ['name' => "为***出 提现",'money' => "273.32"],
3      ['name' => "机***女 提现",'money' => "354.23"],
4      ['name' => "宝***油 提现",'money' => "403.47"],
5      ['name' => "段***段 提现",'money' => "501.91"],
6      ['name' => "外***世 提现",'money' => "303.05"],
7      ['name' => "李***凤 提现",'money' => "251.99"],
8      ['name' => "全***阿 提现",'money' => "453.03"],

```

```

9    ];
10    return ApiReturn::success($arr);
11  }
12
13
14  }
15

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Http\Requests\NewActivityCenterRequest;
7  use App\Models\v1\ServiceConfig;
8  use App\Models\v1\UserUserRole;
9  use App\Service\v1\ApiCode;
10 use App\Service\v1\ApiReturn;
11 use App\Service\v1\Article;

```

```
12 use App\Service\v1\Code;
13 use App\Service\v1\Common;
14 use App\Service\v1\CommonFnc;
15 use App\Service\v1\NewTask;
16 use Illuminate\Http\Request;
17 use Illuminate\Support\Facades\Log;
18 use Throwable;
19
20 /**
21  * 新任务中心
22  */
23 class NewActivityCenterController extends Controller
24 {
25     /**
26      * 获取Tab分类设置的奖励
27      * @param Request $request
28      * @return object
29      */
30     public function getTaskTabList(Request $request): object
31     {
32         //获取userId
33         $userId = $request->get('user_id');
34         return NewTask::getTaskTabList($userId);
35     }
36
37     /**
38      * 活动列表
39      * @param Request $request
40      * @return object
41      */
42     public function getTaskList(Request $request): object
43     {
44         //获取userId
45         $userId = $request->get('user_id');
46         //获取tab分类
47         $taskTabId = $request->get('task_tab_id', 0);
48         $limit = (int)$request->get('limit', 10);
49         $page = (int)$request->get('page', 1);
50         $page = ($page - 1) * $limit;
51
52         $taskStatus = (int)$request->get('task_status', 0);
53         $time = $request->server('REQUEST_TIME');
54         return NewTask::getTaskList($userId, $taskTabId, $time, $page, $limit, $taskStatus);
55     }
56
57     /**
58      * 任务详情
59      * @param Request $request
60      * @return Object
61      */
62     public function taskInfo(Request $request): object
63     {
64         $taskId = intval($request->get('task_id', 0));
65         $userId = $request->get('user_id', 0);
```

```
15 // if(empty($userId)){
16 //     return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
17 // }
18 if ($taskId <= 0) return ApiReturn::rdata(ApiCode::PARAM_ERROR);
19 $platform = $request->header('Platform');
20 return NewTask::taskInfo($taskId, $userId,$platform);
21 }
22
23 //参与任务
24 public function participatingTask(NewActivityCenterRequest $request): object
25 {
26     Log::info('用户参与任务入参', $request->all());
27 //     $is_creative_camp = $request->post('is_creative_camp', 0);
28     $is_creative_camp = 0;
29     $param = $request->post();
30     if (empty($param['type'])) {
31         return ApiReturn::error('任务参与类型错误');
32     }
33     $param['action'] = $request->post('action', Code::ACTION_ADD);
34     if ($param['type'] == Code::ARTICLE_TYPE_MAP['evaluating']) {
35         $param['type'] = Code::ARTICLE_TYPE_MAP['evaluate'];
36     } else {
37         $param['type'] = intval($param['type']);
38     }
39     if ($param['type'] == Code::ARTICLE_TYPE_MAP['evaluate'] || $param['type'] == Code::ARTICLE_TYPE_MAP['evaluating']) {
40         if ($is_creative_camp <= 0) {
41             $request->validate('participatingTask');
42         }
43         if(empty($param['game_score'])){
44             return ApiReturn::error('请输入游戏评分');
45         }
46     } else {
47         if ($is_creative_camp > 0) {
48             return ApiReturn::error('此任务类型暂时不支持创作营');
49         }
50     }
51 }
52
53 $ip = CommonFnc::getRealIP();
54 try {
55     $param['action'] = $request->post('action', Code::ACTION_ADD);
56     $param['status'] = $request->post('status', 0);
57     Log::info('参与任务==入口参数====', $param);
58     // 验证 设置
59     $err = NewTask::checkParticipatingTask($param, $is_creative_camp,$ip);
60     if ($err) {
61         Log::info('参与任务==校验失败====', $err);
62         return ApiReturn::rdata($err);
63     }
64     // 插入到 article 表的数据
65     $param['data'] = [
66         'cover' => $param['cover'],
67         'game_score' => empty($param['game_score']) ? 0 : $param['game_score'],
68         'check_status' => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
69         'content' => $param['content'],
70     ]
71 }
```



```

18         // article 标题 截取
19         'title' => "",
20         'text_content' => $param['text_content'],
21         'content_len' => $param['content_len'],
22         'home_page' => $param['home_page'],
23         'label_list' => $param['label_list'],
24         'abstract' => $param['abstract'],
25         'user_id' => $param['user_id'],
26         'creator_source_id' => $param['creator_source_id'],
27         'task_article_status' => $param['task_article_status'],
28         'state' => 1,
29         'task_tab_id' => $param['task_tab_id']
30     ];
31     if ($param['action'] == Code::ACTION_ADD) {
32         // 新增需要的数据 (修改的时候不可更改的数据)
33         $param['data'] = array_merge($param['data'], [
34             'parent_id' => empty($param['parent_id']) ? 0 : $param['parent_id'],
35             'game_id' => $param['game_id'],
36             'task_id' => $param['task_id'],
37             'circle_id' => $param['circle_id'],
38             'type' => $param['type'],
39             'task_tab_id' => $param['task_tab_id'],
40             'task_article_status' => $param['task_article_status'],
41             'status' => $param['status'],
42             'is_writer_task' => $param['is_writer_task'],
43             'creator_source_id' => $param['creator_source_id']
44         ]);
45     }
46     Log::info('参与任务==进入数据库参数====', $param);
47     return Article::addNewArticle($param,$ip);
48 } catch (Throwable $throwable) {
49     Common::log('发布动态', $throwable);
50     return ApiReturn::error();
}

1     }
2 }
3
4 /**
5  * 新增/编辑收集地址
6  * @param NewActivityCenterRequest $request
7  * @return Object
8  */
9 public function editCollectPlatformUrl(NewActivityCenterRequest $request): object
10 {
11     $params = $request->post('activity_center_platform_url', []);
12     if (is_string($params)) $params = json_decode($params, true); // ios 传参问题
13     $articleId = (int)$request->get('article_id', 0);
14     if (empty($params)) {
15         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
16     }
17     $type = $request->get('type', 0);
18     $userId = $request->post('user_id');
19     return NewTask::editCollectPlatformUrl($userId, $articleId, $params, $type);
20 }

```

```

21
22 //放弃任务
23 public function abortMission(NewActivityCenterRequest $request): object
24 {
25     $userId = $request->post('user_id');
26     $articleId = (int)$request->get('article_id', 0);
27     if (empty($articleId)) {
28         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
29     }
30     Log::info('放弃任务==入口参数==article_id=' . $articleId . '==user_id==' . $userId);
31     return NewTask::abortMission($userId, $articleId);
32 }
33
34 /**
35  * 获取创作营内容
36  * @param Request $request
37  * @return Object
38  */
39 public function getChannelContent(Request $request): object
40 {
41     $taskId = $request->get('task_id');
42     if (empty($taskId)) {
43         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
44     }
45     $userId = $request->get('user_id');
46     if (empty($userId)) {
47         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
48     }
49     return NewTask::getChannelContent($taskId, $userId);
50 }

```

```

1
2 /**
3  * 获取用户参与任务信息
4  * @param Request $request
5  * @return Object
6  */
7 public function getUserTaskInfo(Request $request): object
8 {
9     $user_id = $request->get('user_id', 0);
10    if (empty($user_id)) {
11        return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
12    }
13    $task_id = $request->get('task_id', 0);
14    if (empty($task_id)) {
15        return ApiReturn::rdata(ApiCode::PARAM_ERROR);
16    }
17    $platform = $request->header('Platform');
18    return NewTask::getUserTaskInfo($user_id, $task_id,$platform);
19 }
20
21 /**
22  * 用户任务列表
23  * @param Request $request

```

```
24 * @return Object
25 */
26 public function getUserTaskList(Request $request): object
27 {
28     $user_id = $request->get('user_id');
29     if (empty($user_id)) {
30         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
31     }
32     $limit = (int)$request->get('limit', 10);
33     $page = (int)$request->get('page', 1);
34     return NewTask::getUserTaskList($user_id, $page, $limit);
35 }
36
37 /**
38  * 活动中心用户信息
39  * @param Request $request
40  * @return Object
41  */
42 public function getUserInfo(Request $request): object
43 {
44     $user_id = $request->get('user_id');
45     if (empty($user_id)) {
46         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
47     }
48     return NewTask::getUserInfo($user_id);
49 }
50 //检测非鉴游团用户是否显示任务


---


1 public function userShowTask(Request $request): object
2 {
3     $data = [
4         'show' => false
5     ];
6     $user_id = $request->get('user_id');
7     if (empty($user_id)) {
8         //获取游客用户是否显示任务入口配置
9         if (ServiceConfig::query()->where('tab', 4)->where('field_name', 'is_show_task_for_guests')->value('value')){
10             $data['show'] = true;
11         }
12         return ApiReturn::success($data);
13     }
14     //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
15     if (!ServiceConfig::query()->where('tab', 4)->where('field_name', 'show_task_functionality')->value('value')) {
16         if (!UserUserRole::query()->where('user_id', $user_id)->where('user_role_id', 6)->first()) {
17             return ApiReturn::success($data);
18         }
19     }
20     $data['show'] = true;
21     return ApiReturn::success($data);
22 }
23
24 /**
25  * 检测用于用户参与活动权限
26  * @param Request $request
```

```

27     * @return Object
28     */
29     public function checkUserParticipationTask(Request $request): object
30     {
31         $userId = $request->get('user_id');
32         if (empty($userId)) {
33             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
34         }
35         $taskId = $request->get('task_id');
36         if (empty($taskId)) {
37             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
38         }
39         return NewTask::checkUserParticipationTask($userId, $taskId);
40     }
41 }
42

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\Notification;
7  use App\Notifications\UserNotification;
8  use App\Service\v1\ApiReturn;
9  use App\Service\v1\UserNotifications;
10 use Illuminate\Http\Request;
11 use Illuminate\Support\Facades\Log;
12
13 /**
14  * 消息通知
15  */
16 class NotificationController extends Controller
17 {
18     public function categories(Request $request)
19     {
20         $user_id = $request->get('user_id');
21         return UserNotifications::categories($user_id);
22     }
23     //消息列表
24     public function list(Request $request)
25     {
26         $user_id = $request->get('user_id');
27         $limit = $request->get('limit',10);
28         $category = $request->get('category',0);
29         return UserNotifications::list($user_id,$category,$limit);
30     }
31     //读取单条
32     public function readOne(Request $request)
33     {
34         $user_id = $request->post('user_id');
35         $notify_id = $request->post('notify_id',0);
36         return UserNotifications::readOne($user_id,$notify_id);
37     }

```

```

38 //读取所用通知
39 public function readAll(Request $request)
40 {
41     $user_id = $request->post('user_id');
42     $category = $request->get('category',0);
43     if(empty($category)){
44         return ApiReturn::success();
45     }
46     return UserNotifications::readAll($user_id,$category);
47 }
48
49 /**
50  * 获取用户 是否有新消息

```

```

1  * @param Request $request
2  * @return Object
3  */
4  public function getUserNewNotic(Request $request): object
5  {
6      try {
7          $uid = $request->get('user_id');
8          $rdata = ['new' => 0];
9          if($uid > 0){
10             $rdata['new'] = Notification::query()->where('notifiable_id',$uid)->whereNull('read_at')->count();
11         }
12         return ApiReturn::success($rdata);
13     } catch (\Exception $e) {
14         return ApiReturn::success($rdata);
15     }
16 }
17 }
18

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Events\OrderEvent;
6  use App\Http\Controllers\Controller;
7  use App\Models\v1\Cdk;
8  use App\Models\v1\Order;
9  use App\Models\v1\Product;
10 use App\Models\v1\ProductDiscountCoupon;
11 use App\Models\v1\User;
12 use App\Service\v1\ApiCode;
13 use App\Service\v1\ApiReturn;
14 use App\Service\v1\Common;
15 use App\Service\v1\OrderService;
16 use App\Service\v1\SoloCode;
17 use Illuminate\Http\Request;
18 use Illuminate\Support\Facades\DB;
19 use Illuminate\Support\Facades\Log;
20
21 class OrderController extends Controller
22 {

```

```

23  /**
24   * 钻石购买商品(积分支付)
25   * @param Request $request
26   * @return Object
27   */
28   public function diamondBuyProduct(Request $request)
29   {
30       try {
31           $productId = intval($request->post('product_id',0));
32           $userAddressId = intval($request->post('user_address_id',0));
33           $userId = $request->post('user_id');
34           if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
35           $remark = strval($request->post('remark',''));
36           $productInfo = Product::where(['id' => $productId])
37               ->select('id','name','cover','product_type_id','price','reference_price','diamond','type','pay_type',
38                   'quantity_sold','usable_inventory','total_inventor','cdk_status')->first();
39           if(empty($productInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
40           if($productInfo->usable_inventory <= 0) return
41           ApiReturn::error(ApiCode::INVENTORY_ERROR['msg'],ApiCode::INVENTORY_ERROR['code']);
42           if($productInfo->pay_type == 1) return
43           ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
44           if($productInfo->type == 1 && $userAddressId <= 0) return
45           ApiReturn::error(ApiCode::ADDRESS_ERROR['msg'],ApiCode::ADDRESS_ERROR['code']);
46           $userInfo = User::query()->where(['id' => $userId])->first(['id','diamond_num','freezing_diamond']);
47           if(($userInfo->diamond_num - $userInfo->freezing_diamond) < $productInfo->diamond) return
48           ApiReturn::error(ApiCode::DISCOUNT_ERROR['msg'],ApiCode::DISCOUNT_ERROR['code']);
49           DB::beginTransaction();
50           $ckdInfo = OrderService::pushCdk($productInfo->type,$productInfo->id,$userInfo->id,$productInfo->cdk_status);
51           if(empty($ckdInfo['cdk'])) {
52               DB::rollBack();
53               return ApiReturn::error(ApiCode::CDK_ERROR['msg'],ApiCode::CDK_ERROR['code']);
54           }
55       }
56   }
57
58   1   $res = OrderService::diamondBuyVirtualProduct($userInfo,$userAddressId,$productInfo,$remark,$ckdInfo);
59   2   if($res){
60   3       DB::commit();
61   4       return ApiReturn::success();
62   5   }
63   6   DB::rollBack();
64   7   return ApiReturn::error(ApiCode::RESULT_ERROR['msg'],ApiCode::RESULT_ERROR['code']);
65   8   } catch (\Throwable $throwable) {
66   9       DB::rollBack();
67   10      Common::log('钻石购买商品',$throwable);
68   11      return ApiReturn::error();
69   12  }
70   13  }
71
72   14
73   15
74   16  /**
75   17   * 钱+积分支付
76   18   * @param Request $request
77   19   * @return Object
78   20   */
79   21  public function moneyBuyProduct(Request $request)

```

```

22     {
23         try {
24             $productId = intval($request->post('product_id',0));
25             $userAddressId = intval($request->post('user_address_id',0));
26             $productDiscountCouponId = intval($request->post('product_discount_coupon_id',0));
27             $type = intval($request->post('type',0));
28             $userId = $request->post('user_id');
29             if($productId <= 0 || !isset(SoloCode::WECHATPAY_TYPE[$type])) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
30             $productInfo = Product::where(['id' => $productId])
31                 ->select('id','name','cover','product_type_id','price','reference_price','diamond','type','pay_type',
32                     'quantity_sold','usable_inventory','total_inventor','discount_coupon_id','cdk_status')->first();
33             if(empty($productInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
34             if($productInfo->usable_inventory <= 0) return
ApiReturn::error(ApiCode::INVENTORY_ERROR['msg'],ApiCode::INVENTORY_ERROR['code']);
35
36             if($productInfo->pay_type == 0) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
37
38             if($productInfo->type == 1 && $userAddressId <= 0) return
ApiReturn::error(ApiCode::ADDRESS_ERROR['msg'],ApiCode::ADDRESS_ERROR['code']);
39             if($productInfo->type == 0 && empty(Cdk::where(['product_id' => $productId,'user_cdk_id' => 0])->value('cdk')))
return ApiReturn::error(ApiCode::CDK_ERROR['msg'],ApiCode::CDK_ERROR['code']);
40             if($productDiscountCouponId > 0) {
41                 $discountCouponInfo = ProductDiscountCoupon::where([
42                     'id' => $productDiscountCouponId,
43                     'discount_coupon_id' => $productInfo->discount_coupon_id,
44                     'is_use' => 1
45                 ])->select('id','discount_coupon_price','is_use')->first();
46                 if(empty($discountCouponInfo)) return
ApiReturn::error(ApiCode::DISCOUNT_COUPON_ERROR['msg'],ApiCode::DISCOUNT_COUPON_ERROR['code']);
47                 $productInfo->discount_coupon_price = $discountCouponInfo->discount_coupon_price;
48             }else{
49                 $discountCouponInfo = "";
50                 $productInfo->discount_coupon_price = 0;
51
52             }
53             DB::beginTransaction();
54             $remark = strval($request->post('remark',''));
55             $res =
OrderService::moneyBuyVirtualProduct($userId,$userAddressId,$productInfo,$discountCouponInfo,$type,$remark);
56             if(empty($res)) {
57                 DB::rollBack();
58                 return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
59             }else if($res['code'] == 200){
60                 DB::commit();
61                 OrderEvent::dispatch($userId,SoloCode::OBLIGATION_TITLE,SoloCode::OBLIGATION_EXPLAIN);
62                 return ApiReturn::success($res['data']);
63             }
64             DB::rollBack();
65             return ApiReturn::error($res['msg'],$res['code']);
66         } catch (\Throwable $throwable) {
67             DB::rollBack();
68             Common::log('money+积分购买商品',$throwable);

```

```
18         return ApiReturn::error();
19     }
20 }
21
22 /**
23  * 订单列表
24  * @param Request $request
25  * @return Object
26  */
27 public function orders(Request $request)
28 {
29     try {
30         // $provinceType = intval($request->get('province_type',0));
31         $status = intval($request->get('status',0));
32         if(in_array($status,[0,1,2,3,4,5,6,7])) return
33         ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
34         $userId = $request->get('user_id');
35         $page = intval($request->get('page',1));
36         $limit = intval($request->get('limit',10));
37         $page = ($page - 1) * $limit;
38         $where = ['user_id' => $userId];
39         if($status > 0) $where['status'] = $status;
40         $field = [
41             'id','order_num','total_diamond','discounts_total_price','total_price','pay_type',
42             'status','created_at','product_pay_type','refuse_reason'
43         ];
44         $list = OrderService::orders($where,$field,$page,$limit);
45         return ApiReturn::success($list);
46     } catch (\Throwable $throwable) {
47         Common::log('订单列表',$throwable);
48         return ApiReturn::error();
49     }
50 }

```

```
1  /**
2   * 订单详情
3   * @param Request $request
4   * @return Object
5   */
6   public function orderInfo(Request $request)
7   {
8       try {
9           $orderId = intval($request->get('order_id',0));
10          if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
11          $userId = intval($request->get('user_id',0));
12          $info = OrderService::orderInfo($orderId,$userId);
13          return ApiReturn::success($info);
14      } catch (\Throwable $throwable) {
15          Common::log('订单详情',$throwable);
16          return ApiReturn::error($throwable->getMessage());
17      }
18  }
19
```



```

20  /**
21  * 确认完成订单
22  * @param Request $request
23  * @return Object
24  */
25  public function finishOrder(Request $request)
26  {
27      try {
28          $orderId = intval($request->post('order_id',0));
29          if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
30          $userId = $request->post('user_id');
31          $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId,'status' => 3])>first();
32          if(empty($orderInfo)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
33          $orderInfo->status = 4;
34          $orderInfo->completion_time = time();
35          $orderInfo->save();
36          OrderEvent::dispatch($userId,SoloCode::FINISH_TITLE,SoloCode::FINISH_EXPLAIN);
37          return ApiReturn::success();
38      } catch (\Throwable $throwable) {
39          Common::log('确认完成订单',$throwable);
40          return ApiReturn::error();
41      }
42  }
43
44  /**
45  * 删除订单
46  * @param Request $request
47  * @return Object
48  */
49  public function deleteOrder(Request $request)
50  {
51
52      try {
53          $orderId = intval($request->post('order_id',0));
54          if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
55          $userId = intval($request->post('user_id',0));
56          $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])>first();
57          if(empty($orderInfo)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
58          OrderService::deleteOrder($orderInfo);
59          return ApiReturn::success();
60      } catch (\Throwable $throwable) {
61          Common::log('删除订单',$throwable);
62          return ApiReturn::error();
63      }
64  }
65
66  /**
67  * 取消订单
68  * @param Request $request
69  * @return Object
70  */

```

```
21 public function cancelOrder(Request $request)
22 {
23     try {
24         $orderId = intval($request->post('order_id',0));
25         if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
26         $userId = intval($request->post('user_id',0));
27         $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])->first();
28         if(empty($orderInfo)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
29         if($orderInfo->status == 1) {
30             $orderInfo->status = 7;
31             $orderInfo->save();
32             return ApiReturn::success();
33         }
34         return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
35     } catch (\Throwable $throwable) {
36         Common::log('取消订单',$throwable);
37         return ApiReturn::error();
38     }
39 }
40
41 /**
42  * 继续支付
43  * @param Request $request
44  * @return Object
45  */
46 public function continuePay(Request $request)
47 {
48     try {
49         $orderId = intval($request->post('order_id',0));
50         $type = intval($request->post('type',1));


---


1         if($orderId <= 0 || !isset(SoloCode::WECHATPAY_TYPE[$type])) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
2         $userId = intval($request->post('user_id',0));
3         $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])->first();
4         if(empty($orderInfo) || $orderInfo->status != 1) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
5         $response = OrderService::continuePay($orderInfo,$type);
6         return ApiReturn::success($response['data']);
7     } catch (\Throwable $throwable) {
8         Common::log('继续支付',$throwable);
9         return ApiReturn::error();
10    }
11 }
12
13 /**
14  * 微信app支付回调
15  * @param Request $request
16  * @return \Illuminate\Http\JsonResponse
17  */
18 public function wechatOrderNotify(Request $request)
19 {
20     try {
```

```

21     $wechatHeader = [
22         'wechatpay_signature' => $request->header('wechatpay-signature'),
23         'wechatpay_timestamp' => $request->header('wechatpay-timestamp'),
24         'wechatpay_serial' => $request->header('wechatpay-serial'),
25         'wechatpay_nonce' => $request->header('wechatpay-nonce'),
26     ];
27     $inBody = $request->getContent();
28     Log::info('微信App支付回调接受参数',[$wechatHeader,$inBody]);
29     DB::beginTransaction();
30     $resource = OrderService::wechatOrderNotify($wechatHeader,$inBody);
31     if($resource) {
32         DB::commit();
33         return response()->json(['code' => 'SUCCESS','message' => '成功']);
34     }
35     DB::rollBack();
36     return response()->json(['code' => 'ERROR','message' => '失败']);
37 } catch (\Throwable $throwable) {
38     DB::rollBack();
39     Common::log('微信支付回调',$throwable);
40     return response()->json(['code' => 'ERROR','message' => '失败']);
41 }
42 }
43
44 /**
45  * 退款
46  * @param Request $request
47  * @return Object
48  */
49 public function orderRefund(Request $request)
50 {


---


1     try {
2         $orderId = intval($request->post('order_id',0));
3         if($orderId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
4         $userId = intval($request->post('user_id',0));
5         $orderInfo = Order::where(['id' => $orderId,'user_id' => $userId])->first();
6         if(empty($orderInfo) || !in_array($orderInfo->status,[2,3,4]) || !empty($orderInfo->refund_order_num)) return
ApiReturn::error(ApiCode::OPERATION_ERROR['msg'],ApiCode::OPERATION_ERROR['code']);
7         $response = OrderService::orderRefund($orderInfo);
8         if($response['code'] == 200) return ApiReturn::success();
9         return ApiReturn::error($response['msg'],$response['code']);
10    } catch (\Throwable $throwable) {
11        Common::log('退款',$throwable);
12        return ApiReturn::error();
13    }
14 }
15
16 }
17


---


1 <?php
2
3 namespace App\Http\Controllers\api\v1;
4
5 use App\Http\Controllers\Controller;

```

```

6  use App\Models\Contact;
7  use App\Service\v1\ApiReturn;
8  use App\Service\v1\CommonFnc;
9  use App\Service\v1\Feishu;
10 use Illuminate\Http\Request;
11 use Mews\Captcha\Facades\Captcha;
12
13 class PcHomeController extends Controller
14 {
15     //提交联系我们信息
16     public function submitContact(Request $request): object
17     {
18         $captcha = $request->post('captcha');
19         if (empty($captcha)) {
20             return ApiReturn::error('请输入验证码');
21         }
22         $key = $request->post('key');
23         if (empty($key)) {
24             return ApiReturn::error('请输入验证码key');
25         }
26         $name = $request->post('name');
27         if (empty($name)) {
28             return ApiReturn::error('请填写姓名');
29         }
30         $contact = $request->post('contact');
31         if (empty($contact)) {
32             return ApiReturn::error('请填写联系方式');
33         }
34         $user_id = $request->post('user_id', 0);
35         $isCaptchaValid = captcha_api_check($captcha, $key);
36         if ($isCaptchaValid) {
37             try {
38                 // CAPTCHA 验证成功，执行其他逻辑
39                 Contact::query()->create([
40                     'name' => $name,
41                     'contact' => $contact,
42                     'ip' => CommonFnc::getRealIP(),
43                     'user_id' => $user_id
44                 ]);
45                 $content = [
46                     'msg_type' => 'interactive',
47                     'card' => [
48                         'header' => [
49                             'title' =>[
50                                 'content' => '意向客户',
51
52                                 'tag' => 'plain_text'
53                             ],
54                         ],
55                     ],
56                     'elements' => [
57                         [
58                             'tag' => "div",
59                             'text' => [
60                                 'content' => '**姓名：**' . $name . ' **联系方式：**' . $contact,

```

```

9             "tag" => "lark_md"
10        ]
11    ],
12    [
13        'actions' => [
14            [
15                'tag' => 'button',
16                'text' => [
17                    'content' => '点击查看',
18                    'tag' => 'lark_md'
19                ],
20                'url' => env('SOLO_ADMIN_HOST').admin/contacts',
21                'type' => 'default',
22                'value' => []
23            ]
24        ],
25        "tag" => "action"
26    ]
27 ]
28 ]
29 ];
30 Feishu::sendWebhookCrowdRobotMessage($content);
31 return ApiReturn::success('提交成功');
32 } catch
33 (\Exception $e) {
34     return ApiReturn::error('提交失败');
35 }
36 } else {
37     // CAPTCHA 验证失败，返回错误提示
38     return ApiReturn::error('验证码错误');
39 }
40 }
41 }
42

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\ProductType;
7  use App\Models\v1\DiscountCoupon;
8  use App\Models\v1\Product;
9  use App\Models\v1\ProductDiscountCoupon;
10 use App\Models\v1\User;
11 use App\Models\v1\UserDiamondLog;
12 use App\Service\v1\ApiCode;
13 use App\Service\v1\ApiReturn;
14 use App\Service\v1\Common;
15 use App\Service\v1\ProductService;
16 use App\Service\v1\RedisKey;
17 use App\Service\v1\SoloCode;
18 use Illuminate\Http\Request;
19 use Illuminate\Support\Facades\Cache;

```

```
20 use Illuminate\Support\Facades\DB;
21 use Illuminate\Support\Facades\Redis;
22 use Predis\Client;
23
24 class ProductController extends Controller
25 {
26     /**
27      * 商品分类
28      * @return Object
29      */
30     public function productTypes()
31     {
32         try {
33             $list = Cache::store('common_redis')->rememberForever(RedisKey::PRODUCT_TYPE,function (){
34                 $list = ProductType::query()->where('status',1)->select('id','name')->get();
35                 return json_encode($list,JSON_UNESCAPED_UNICODE);
36             });
37             return ApiReturn::success(json_decode($list,true));
38         } catch (\Throwable $throwable) {
39             Common::log('商品分类',$throwable);
40             return ApiReturn::error();
41         }
42     }
43
44     /**
45      * 商品列表
46      * @param Request $request
47      * @return Object
48      */
49     public function products(Request $request)
50     {
```

```
1         try {
2             $productId = intval($request->get('product_type_id',0));
3             $page = intval($request->get('page',1));
4             $limit = intval($request->get('limit',10));
5             $page = ($page - 1) * $limit;
6             $list = ProductService::products($productId,$page,$limit);
7             return ApiReturn::success($list);
8         } catch (\Throwable $throwable) {
9             Common::log('商品列表',$throwable);
10            return ApiReturn::error();
11        }
12    }
13
14
15    /**
16     * 商品详情
17     * @param Request $request
18     * @return Object
19     */
20    public function productInfo(Request $request)
21    {
22        try {
```

```

23     $productId = intval($request->get('product_id',0));
24     if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
25     $list = ProductService::productInfo($productId);
26     return ApiReturn::success($list);
27 } catch (\Throwable $throwable) {
28     Common::log('商品详情',$throwable);
29     return ApiReturn::error();
30 }
31 }
32
33 /**
34  * 兑换优惠券
35  * @param Request $request
36  * @return Object
37  */
38 public function exchangeDiscountCoupon(Request $request)
39 {
40     try {
41         $productId = intval($request->post('product_id',0));
42         if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
43         $userId = $request->post('user_id');
44         DB::beginTransaction();
45         $productInfo = Product::where(['id' => $productId])
46             ->select('diamond','pay_type','discount_coupon_id')->first();
47         $userInfo = User::query()->where(['id' => $userId])->first(['id','diamond_num','freezing_diamond']);
48         if($productInfo->pay_type !== SoloCode::PAY_TYPE_DISCOUNT_MONEY) return
49             ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
50         $discountCoupon = DiscountCoupon::where(['id' =>
51             $productInfo->discount_coupon_id])->select('id','discount_coupon_price','discount_coupon_num')->first();
52         if($discountCoupon->discount_coupon_price > ($userInfo->diamond_num - $userInfo->freezing_diamond)) return
53             ApiReturn::error(ApiCode::DISCOUNT_ERROR['msg'],ApiCode::DISCOUNT_ERROR['code']);
54
55         if($discountCoupon->discount_coupon_num <= 0) return
56             ApiReturn::error(ApiCode::INVENTORY_ERROR['msg'],ApiCode::INVENTORY_ERROR['code']);
57         $discountCoupon->discount_coupon_num -= 1;
58         $discountCoupon->save();
59         $userInfo->diamond_num = $userInfo->diamond_num - $productInfo->diamond;
60         $userInfo->save();
61         ProductDiscountCoupon::create([
62             'discount_coupon_id' => $productInfo->discount_coupon_id,
63             'user_id' => $userId,
64             'discount_coupon_price' => $discountCoupon->discount_coupon_price,
65             'diamond' => $productInfo->diamond
66         ]);
67         $data = [
68             'user_id' => $userId,
69             'source_id' => $productId,
70             'type' => 7,
71             'describe' => '购买优惠券',
72             'diamond' => $productInfo->diamond,
73             'balance' => $userInfo->diamond_num,
74             'action' => 2
75         ];
76         UserDiamondLog::create($data);

```

```

22         DB::commit();
23         User::where('id',$userId)->searchable();
24         return ApiReturn::success();
25     } catch (\Throwable $throwable) {
26         DB::rollBack();
27         Common::log('兑换优惠券',$throwable);
28         return ApiReturn::error();
29     }
30 }
31
32
33 /**
34  * 确认订单页
35  * @param Request $request
36  * @return Object
37  */
38 public function verifyOrder(Request $request)
39 {
40     try {
41         $productId = intval($request->get('product_id',0));
42         if($productId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
43         $userId = $request->get('user_id');
44         $info = ProductService::verifyOrder($productId,$userId);
45         return ApiReturn::success($info);
46     } catch (\Throwable $throwable) {
47         Common::log('确认订单页',$throwable);
48         return ApiReturn::error();
49     }
50 }

```

```

1 }
2

```

```

1 <?php
2
3 namespace App\Http\Controllers\api\v1\Questionnaire;
4
5 use App\Http\Controllers\Controller;
6 use App\Http\Requests\AnswerRequest;
7 use App\Models\v1\Questionnaires\Questionnaire;
8 use App\Models\v1\Questionnaires\QuestionnaireQuestionOption;
9 use App\Models\v1\Questionnaires\QuestionnaireUserAnswer;
10 use App\Models\v1\Questionnaires\QuestionnaireUserAnswerOption;
11 use App\Models\v1\Questionnaires\QuestionnaireUserAnswerQuestion;
12 use App\Models\v1\Questionnaires\QuestionnaireUserRole;
13 use App\Models\v1\User;
14 use App\Models\v1\UserUserRole;
15 use App\Service\v1\ApiCode;
16 use App\Service\v1\ApiReturn;
17 use App\Service\v1\BaiDuSdk;
18 use App\Service\v1\Code;
19 use App\Service\v1\Common;
20 use App\Service\v1\CommonFnc;
21 use App\Service\v1\RedisKey;
22 use App\Service\v1\UserNotifications;

```



```

23 use Illuminate\Http\Request;
24 use Illuminate\Support\Facades\DB;
25 use Illuminate\Support\Facades\Redis;
26
27 class QuestionnaireController extends Controller
28 {
29
30     /**
31      * 问卷页面
32      * @catalog
33      * @title 问卷页面
34      * @description 描述信息
35      * @method get
36      * @url /api/questionnaire/questionnaireInfo
37      * @param questionnaire_id 选填 int 问卷id
38      * @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"}]}}}}
39      * @return_param id int 问卷id
40      * @return_param type int 状态 1:考试 0:问卷
41      * @return_param title string 标题
42      * @return_param sub_title string 副标题
43      * @return_param submit_num int 用户提交次数
44      * @return_param submit_message string 提交成功提示
45      * @return_param description string 描述
46      * @return_param start_time string 开始时间
47      * @return_param end_time string 结束时间
48      * @return_param question.id int 试题id
49      * @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
50      * @return_param question.is_required int 是否必填 : 1 是 , 0 否
51
52      * @return_param question.is_random int 是否随机显示选项(单选, 多选) : 1 是 , 0 否
53      * @return_param question.fraction int 分数 0-100
54      * @return_param question.title string 标题
55      * @return_param question.tips string 提示
56      * @return_param question.upload_num int 上传文件数量
57      * @return_param question.upload_max int 上传文件大小
58      * @return_param question.upload_type int 上传文件类型 : 1 图片 , 2 视频 , 3 压缩文件
59      * @return_param question.options.id int 选项id
60      * @return_param question.options.write_able int 是否可以填空 1:可以 0:不可以
61      * @return_param question.options.correct int 是否正确 1:正确 0:错误
62      * @return_param question.options.sort int 排序
63      * @return_param question.options.title string 标题
64      */
65     public function questionnaireInfo(Request $request)
66     {
67         try {
68             $code = (string)$request->get('code','');
69             if(empty($code)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
70             $questionnaireInfo = Questionnaire::query()->with(['question','question.options'])
71                 ->where(['code' => $code])->first();
72         } catch (Exception $e) {
73             return ApiReturn::error($e->getMessage(),ApiCode::PARAM_ERROR['code']);
74         }
75     }
76 }

```

```

21         if(empty($questionnaireInfo)) return
    ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
22         $questionnaireInfo = $questionnaireInfo->toArray();
23         $submitNum = QuestionnaireUserAnswer::query()
24             ->where(['questionnaire_id' => $questionnaireInfo['id'],'user_id' => $request->get('user_id')])>count();
25         if($submitNum >= $questionnaireInfo['submit_num']) return
    ApiReturn::error(ApiCode::SUBMIT_NUM['msg'],ApiCode::SUBMIT_NUM['code']);
26         $userRoleIds = QuestionnaireUserRole::query()->where(['questionnaire_id' =>
    $questionnaireInfo['id']])>pluck('user_role_id')>toArray();
27         if(!empty($userRoleIds)) {
28             $userId = $request->get('user_id');
29             $count = UserUserRole::query()->where(['user_id' => $userId])>whereIn('user_role_id',$userRoleIds)>count();
30             if($count == 0) return
    ApiReturn::error(ApiCode::TASK_PERMISSION_DENIED_ERROR['msg'],ApiCode::TASK_PERMISSION_DENIED_ERROR['code']);
31         }
32         foreach ($questionnaireInfo['question'] as &$question) {
33             if(in_array($question['question_type'],[1,2]) && $question['is_random'] == 1) {
34                 shuffle($question['options']);
35             }
36         }
37         $time = time();
38         if($time < strtotime($questionnaireInfo['start_time'])) {
39             return ApiReturn::success(['status' => 1,'questionnaire' => new \stdClass()]);
40         }elseif ($time > strtotime($questionnaireInfo['end_time'])) {
41             return ApiReturn::success(['status' => 2,'questionnaire' => new \stdClass()]);
42         }else{
43             return ApiReturn::success(['status' => 3,'questionnaire' => $questionnaireInfo]);
44         }
45     }catch (\Throwable $throwable) {
46         Common::log('调查问卷错误',$throwable);
47         return ApiReturn::error();
48     }
49 }
50

```

```

1  /**
2   * 用户提交答案
3   * @catalog
4   * @title 用户提交答案
5   * @description 描述信息
6   * @method post
7   * @url /api/questionnaire/submitAnswer
8   * @param questionnaire_id 选填 int 问卷id
9   * @param complete_degree 选填 string 完成度 0-100
10  * @param question.questionnaire_question_id 选填 int 题目id
11  * @param question.answer.questionnaire_question_option_id 选填 int 选项id
12  * @param question.answer.answer 选填 string 答案
13  * @param question.answer.default_answer 选填 string 选择填空答案
14  * @return {"code": 200,"msg": "OK","data": {"fraction_total": 11,"certification": true}}
15  * @return_param fraction_total int 总分, certification bool 是否通过认证
16  */
17  public function submitAnswer(AnswerRequest $request)
18  {
19      $send = false;

```

```

20     $request->validate();
21     $answers = $request->validated();
22     try {
23         $userRoleIds = QuestionnaireUserRole::query()->where(['questionnaire_id' =>
24 $answers['questionnaire_id']])->pluck('user_role_id')->toArray();
25         $answers['user_id'] = $request->post('user_id');
26         if(!empty($userRoleIds)) {
27             $count = UserUserRole::query()->where(['user_id' =>
28 $answers['user_id']])->whereIn('user_role_id',$userRoleIds)->count();
29             if($count == 0) return
30 ApiReturn::error(ApiCode::TASK_PERMISSION_DENIED_ERROR['msg'],ApiCode::TASK_PERMISSION_DENIED_ERROR['code']);
31         }
32         $questionnaireInfo = Questionnaire::query()->with(['question','question.options'])
33         ->where(['id' => $answers['questionnaire_id']])->first();
34         if(empty($questionnaireInfo)) return
35 ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
36         $submitNum = QuestionnaireUserAnswer::query()
37         ->where(['questionnaire_id' => $answers['questionnaire_id'],'user_id' => $answers['user_id']])->count();
38         if($submitNum >= $questionnaireInfo->submit_num) return
39 ApiReturn::error(ApiCode::SUBMIT_NUM['msg'],ApiCode::SUBMIT_NUM['code']);
40         $options = $questionTypes = $fractions = $multipleOptions = [];
41         foreach ($questionnaireInfo->question as $question) {
42             if(in_array($question->question_type,[1,2])) {
43                 foreach ($question->options as $option) {
44                     if ($option->correct == 1) {
45                         if ($question->question_type == 1) {
46                             $options[$option->id] = $option->title;
47                         } else {
48                             if (isset($multipleOptions[$question->id])) $multipleOptions[$question->id][$option->id] = $option->title;
49                             else $multipleOptions[$question->id] = [$option->id => $option->title];
50                         }
51                     }
52                 }
53             }
54         }
55         $questionTypes[$question->id] = $question->question_type;
56         $fractions[$question->id] = $question->fraction;
57     }
58     DB::beginTransaction();
59     $time = date('Y-m-d H:i:s');
60     $answers['fraction_total'] = 0;
61     $questionTotal = count($questionnaireInfo->question);
62     $answerTotal = count($answers['question']);
63     $answers['complete_degree'] = floor($answerTotal/$questionTotal*100);
64     foreach ($answers['question'] as &$answer) {
65         $answer['questionnaire_id'] = $answers['questionnaire_id'];
66         $answer['user_id'] = $answers['user_id'];
67         $answer['questionnaire_question_type'] = $questionTypes[$answer['questionnaire_question_id']];
68         if(in_array($answer['questionnaire_question_type'],[1,2])) {
69             if ($answer['questionnaire_question_type'] == 1) {
70                 foreach ($answer['answer'] as $option) {
71                     if(isset($options[$option['questionnaire_question_option_id']])) {
72                         $answer['correct'] = 1;
73                         $answer['fraction'] = $fractions[$answer['questionnaire_question_id']];
74                     }
75                 }
76             }
77         }
78     }

```

```
18         $answers['fraction_total'] += $answer['fraction'];
19     }else{
20         $answer['correct'] = 0;
21         $answer['fraction'] = 0;
22         break;
23     }
24 }
25 } else {
26     // 获取数组key
27     if(isset($multipleOptions[$answer['questionnaire_question_id']])) {
28         // 多选正答列表
29         $keys = array_keys($multipleOptions[$answer['questionnaire_question_id']]);
30         sort($keys);
31         $optionIds = array_column($answer['answer'], 'questionnaire_question_option_id');
32         sort($optionIds);
33         if (json_encode($keys) == json_encode($optionIds)) {
34             $answer['correct'] = 1;
35             $answer['fraction'] = $fractions[$answer['questionnaire_question_id']];
36             $answers['fraction_total'] += $answer['fraction'];
37         } else {
38             $answer['correct'] = 0;
39             $answer['fraction'] = 0;
40         }
41     } else {
42         $answer['correct'] = 0;
43         $answer['fraction'] = 0;
44     }
45 }
46 }
47 //     $answer['answer'] = json_encode($answer['answer']);
48 //     $answer['created_at'] = $time;
49 //     $answer['updated_at'] = $time;
50 }
```

```
1     $ip = CommonFnc::getRealIP();
2     $answers['user_ip'] = $ip;
3     $answers['user_address'] = BaiDuSdk::ipToAddress($ip);
4     $answerInfo = QuestionnaireUserAnswer::query()->create($answers);
5     foreach ($answers['question'] as &$option) {
6         $option['questionnaire_question_answer_id'] = $answerInfo->id;
7         $questionInfo = QuestionnaireUserAnswerQuestion::query()->create($option);
8         foreach ($option['answer'] as &$answer) {
9             $answer['questionnaire_question_answer_id'] = $answerInfo->id;
10            $answer['questionnaire_id'] = $answers['questionnaire_id'];
11            $answer['questionnaire_question_id'] = $option['questionnaire_question_id'];
12            $answer['user_id'] = $answers['user_id'];
13            $answer['questionnaire_user_answer_question_id'] = $questionInfo->id;
14            $answer['created_at'] = $time;
15            $answer['updated_at'] = $time;
16        }
17        QuestionnaireUserAnswerOption::query()->insert($option['answer']);
18    }
19    $rdata = ['fraction_total' => $answers['fraction_total']];
20    // 自动认证逻辑
```

```

21      $rdata['certification'] = false;
22      // 考试类型 && 设置了自动认证 && 通过率达到要求
23      if ($questionnaireInfo->type == 1) {
24          $rdata['questions'] = $answers['question'];
25          if ($questionnaireInfo->certification_role && $answers['fraction_total'] >= $questionnaireInfo->certification_score)
26          {
27              // $can = true;
28              // // 鉴游团黑名单
29              // if ($questionnaireInfo->certification_role_id == Code::ROLE_TYPE_MAP['jianyoutuan']) {
30              //     $blackList = Redis::lrange(RedisKey::JYT_BLACK_LIST,0,-1);
31              //     if ($blackList && in_array($answers['user_id'],$blackList)) $can = false;
32              // }
33              // if ($can) {
34              //     $rdata['certification'] = true;
35              //     UserUserRole::firstOrCreate(['user_id' => $answers['user_id'], 'user_role_id' =>
36              // $questionnaireInfo->certification_role_id]);
37              //     if ($questionnaireInfo->certification_role_id == Code::ROLE_TYPE_MAP['jianyoutuan']) {
38              //         User::mcnAward($answers['user_id'], 'jyt_award', $answers['user_id'], false);
39              //         UserNotifications::adminPushMessage($answers['user_id'], '鉴游团通知', '您已通过鉴游团考核并解锁了更高级
40              // 务开去参与吧', 'activityCenter');
41              //     }
42              // }
43              $can = true;
44              // 鉴游团黑名单
45              if ($questionnaireInfo->certification_role_id == Code::ROLE_TYPE_MAP['jianyoutuan']) {
46                  $blackList = Redis::lrange(RedisKey::JYT_BLACK_LIST,0,-1);
47                  if ($blackList && in_array($answers['user_id'],$blackList)) $can = false;
48              }
49              if ($can) {
50                  $rdata['certification'] = true;
51                  UserUserRole::firstOrCreate(['user_id' => $answers['user_id'], 'user_role_id' =>
52                  $questionnaireInfo->certification_role_id]);
53                  $send = true;
54              }
55          }
56      }
57  }
58  }
59  }
60  }
61  }
62  }
63  }
64  }
65  }
66  }
67  }
68  }
69  }
70  }
71  }
72  }
73  }
74  }
75  }
76  }
77  }
78  }
79  }
80  }
81  }
82  }
83  }
84  }
85  }
86  }
87  }
88  }
89  }
90  }
91  }
92  }
93  }
94  }
95  }
96  }
97  }
98  }
99  }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

```

1  <?php
2

```

```
3 namespace App\Http\Controllers\api\v1;
4
5 use App\Http\Controllers\Controller;
6 use App\Service\v1\ApiReturn;
7 use App\Service\v1\Quiz;
8 use Illuminate\Http\Request;
9
10 class QuizController extends Controller
11 {
12     //列表
13     public function list(Request $request)
14     {
15         $user_id = $request->get('user_id');
16         return Quiz::list($user_id);
17     }
18
19     //用户提交答案
20     public function answer(Request $request)
21     {
22         //用户ID
23         $user_id = $request->post('user_id');
24         //问题ID
25         $question_id = $request->post('question_id');
26         if (empty($question_id)) {
27             return ApiReturn::error('请传入问题ID');
28         }
29         //用户提交的答案
30         $answer = $request->post('answer');
31         if (empty($answer)) {
32             return ApiReturn::error('请填写答案');
33         }
34         return Quiz::answer($user_id, $question_id, $answer);
35     }
36
37     //问题详情
38     public function questionInfo(Request $request)
39     {
40         //用户ID
41         $user_id = $request->post('user_id');
42         //问题ID
43         $question_id = $request->post('question_id');
44         if (empty($question_id)) {
45             return ApiReturn::error('请传入问题ID');
46         }
47         return Quiz::questionInfo($user_id, $question_id);
48     }
49
50     //开启宝藏
51
52     public function openTreasure(Request $request)
53     {
54         //用户ID
55         $user_id = $request->post('user_id');
56         $treasure_id = $request->post('treasure_id');
```

```

6     if (empty($treasure_id)) {
7         return ApiReturn::error('请选择要开启的宝藏');
8     }
9     $award_key = $request->post('award_key');
10    if (empty($award_key)) {
11        return ApiReturn::error('请选择要开启的宝藏');
12    }
13    return Quiz::openTreasure($user_id, $treasure_id, $award_key);
14 }
15 //用户详情
16 public function userInfo(Request $request)
17 {
18     //用户ID
19     $user_id = $request->post('user_id');
20     return Quiz::userInfo($user_id);
21 }
22 }
23

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\v1\Article;
7  use App\Models\v1\User;
8  use App\Service\v1\ApiReturn;
9  use Illuminate\Http\Request;
10
11 class RobotController extends Controller
12 {
13     public function syncEsArticle(Request $request): object
14     {
15         try {
16             $userId = $request->post('robot_id');
17             if(User::query()->where('id',$userId)->where('is_robot','!=',0)->first()){
18                 Article::where('user_id',$userId)->searchable();
19                 return ApiReturn::error('同步成功');
20             }
21             return ApiReturn::error('查不到该用户');
22         }catch (\Exception $e){
23             return ApiReturn::error('同步失败'.$e->getMessage());
24         }
25     }
26     public function getYesterdayActiveUser(): object
27     {
28         $startOfDay = date('Y-m-d 00:00:00', strtotime('yesterday'));
29         $endOfDay = date('Y-m-d 23:59:59', strtotime('yesterday'));
30
31         $list = User::query()
32             ->whereBetween('visit_time', [$startOfDay, $endOfDay])
33             ->pluck('id')->toArray();
34         return ApiReturn::success($list);
35     }

```

```

36
37 }
38


---


1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Events\SearchLogEvent;
6  use App\Http\Controllers\Controller;
7  use App\Jobs\ErrorLog;
8  use App\Service\v1\ApiReturn;
9  use App\Service\v1\Common;
10 use App\Service\v1\Search;
11 use Illuminate\Http\Request;
12 use Illuminate\Support\Facades\Log;
13
14 class SearchController extends Controller
15 {
16     /**
17      * 综合搜索页面
18      * @param Request $request
19      * @return Object
20      */
21     public function searchLogAndRecommend(Request $request)
22     {
23         $userId = $request->get('user_id');
24         $deviceNum = (string)$request->get('device_num','');
25         try {
26             $data = Search::searchLogAndRecommend($userId,$deviceNum);
27             return ApiReturn::success($data);
28         }catch (\Throwable $throwable) {
29             Common::log('综合搜索页面错误',$throwable);
30             return ApiReturn::error();
31         }
32     }
33
34     /**
35      * 搜索联想
36      * @param Request $request
37      * @return Object
38      */
39     public function searchAssociate(Request $request)
40     {
41         $keyword = (string)$request->post('keyword','');
42         if(empty($keyword)) return ApiReturn::success();
43         try {
44             $data = Search::searchAssociate($keyword);
45             return ApiReturn::success($data);
46         }catch (\Throwable $throwable) {
47             Common::log('搜索联想错误',$throwable);
48             return ApiReturn::error();
49         }
50     }

```

```
1
2  /**
3   * 综合搜索
4   * @param Request $request
5   * @return Object
6   */
7  public function synthesizeSearch(Request $request)
8  {
9      $name = $request->get('name','');
10     if(empty($name)) return ApiReturn::success();
11     $limit = $request->get('limit',10);
12     $page = $request->get('page',1);
13     $type = $request->get('type',0);
14     $userId = $request->get('user_id',0);
15     $deviceNum = $request->get('device_num');
16     try{
17         switch ($type) {
18             case 1:
19                 $field =
20 ["id","game_name","game_label","game_icon","score","evaluate_num","evaluating_num","operation_score","lang"];
21                 $data = Search::gameSearch($name,$page,$limit,$userId,$field);
22                 break;
23             case 2:
24                 $field =
25 ["id","cover",'rss_guid','rss_task_id','link_type','link_url','title','like_num',"game_id","game_name","game_icon","no_like_num","l
26 abel","user_id","username",
27 "attestation_type","avatar","video","share_num","create_time","update_time","content","is_hot","task_id","elite","images","circl
28 e_id","top","comment_num","type","check_status","videos","status","user","circle","rss_account"];
29                 $data = Search::articleSearch($name,$page,$limit,$userId,$field,$deviceNum);
30                 break;
31             case 3:
32                 $field = ["id","username","avatar","description","fans_num","attestation_type"];
33                 $data = Search::userSearch($name,$page,$limit,$userId,$field);
34                 break;
35             case 4:
36                 $field = ["id","circle_name","avatar","game_id","attention_num","articles_num","discussion_num"];
37                 $data = Search::circleSearch($name,$page,$limit,$userId,$field);
38                 break;
39             default:
40                 $data = Search::synthesizeSearch($name,$page,$limit,$userId,$deviceNum);
41                 break;
42         }
43         SearchLogEvent::dispatch($name,$userId,$deviceNum ?? '');
44         return ApiReturn::success($data);
45     }catch (\Exception $e) {
46         Common::log('综合搜索错误',$e);
47         return ApiReturn::error();
48     }
```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\DomesticGameApproval;
7  use App\Models\Leaderboard;
8  use App\Models\v1\Circle;
9  use App\Models\v1\Game;
10 use App\Service\v1\ApiCode;
11 use App\Service\v1\ApiReturn;
12 use App\Service\v1\CircleService;
13 use App\Service\v1\CommonFnc;
14 use App\Service\v1\SpecialSubject;
15 use App\Service\v1\User;
16 use Facade\FlareClient\Api;
17 use Illuminate\Http\Request;
18 use Illuminate\Support\Facades\Log;
19
20 class SpecialSubjectController extends Controller
21 {
22     public function detail(Request $request)
23     {
24         $id = $request->get('id', 1);
25         if (empty($id)) {
26             return ApiReturn::error('请传入 Id');
27         }
28         return SpecialSubject::detail($id);
29     }
30
31     //视频
32     public function videos(Request $request): object
33     {
34         $limit = $request->get('limit', 10);
35         $special_subject_id = $request->get('special_subject_id', 0);
36         if (empty($special_subject_id)) {
37             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
38         }
39         return SpecialSubject::videos($special_subject_id, $limit);
40     }
41
42     //专题列表
43     public function list(Request $request)
44     {
45         $page = $request->get('page', 1);
46         $limit = $request->get('limit', 5);
47         return SpecialSubject::list($limit);
48     }
49
50     //榜单配置

```

```

1  public function leardboard(Request $request)
2  {
3      $limit = $request->get('limit', 3);

```

```
4      $lang = CommonFnc::lang();
5      $user_id = $request->get('user_id');
6      $list = Leaderboard::query()
7          ->where('lang', $lang)
8          ->where('type', 1)
9          ->select(['id', 'name', 'cover']);
10     if ($limit == 8) {
11         $list = $list->limit(3);
12     }
13     if ($limit >= 8) {
14         $list = $list->withCount('games')->orderBy('sort', 'desc')->get();
15     } else {
16         $list = $list->get();
17     }
18     $list->load(['games.game_labels' => function ($query) use ($lang) {
19         $query->where('lang', $lang);
20     }]);
21     $list->each(function ($leaderboard) use ($limit) {
22         $leaderboard->setRelation('games', $leaderboard->games->take($limit));
23     });
24     $list = isset($list) ? $list->toArray() : [];
25     $data = [];
26     foreach ($list as $k => $v) {
27         $data[$k]['id'] = $v['id'];
28         $data[$k]['name'] = $v['name'];
29         $data[$k]['ranking_count'] = $v['games_count'] ?? 0;
30         $data[$k]['cover'] = CommonFnc::concatUrlStr($v['cover']);
31         $games = [];
32         foreach ($v['games'] as $k1 => $v1) {
33             $games[$k1]['position'] = $k1 + 1;
34             if ($k1 == 0) {
35                 $games[$k1]['check'] = true;
36             } else {
37                 $games[$k1]['check'] = false;
38             }
39             $games[$k1]['id'] = $v1['id'];
40             $games[$k1]['game_name'] = $v1['game_name'];
41             $games[$k1]['score'] = $v1['score'];
42             $games[$k1]['game_icon'] = CommonFnc::concatUrlStrAdmin($v1['game_icon']);
43             $game_label = '';
44             foreach ($v1['game_labels'] as $k2 => $v2) {
45                 if ($k2 <= 2) {
46                     $game_label .= $v2['label_name'] . ' ';
47                 }
48             }
49             $circle = Circle::query()->where('game_id', $v1['id'])->select(['id', 'circle_name'])->first('id');
50             if ($circle) {
51                 $circle['is_attention'] = CircleService::isAttention($user_id, $circle['id']);
52             }
53             $games[$k1]['circle'] = $circle;
54             $games[$k1]['circle_id'] = $circle['id'] ?? 0;
55             $games[$k1]['game_label'] = rtrim($game_label, ' ');
56         }
57     }
```

```

7         $data[$k]['ranking'] = $games;
8     }
9     return ApiReturn::success($data);
10 }
11
12 public function ranking(Request $request)
13 {
14     $id = $request->get('id', 1);
15     $page = $request->get('page', 1);
16     $limit = $request->get('limit', 10);
17     $user_id = $request->get('user_id');
18     if ($page * $limit > 150) {
19         return ApiReturn::success();
20     }
21     $leardboard = Leaderboard::query()->select(['id', 'lang', 'name', 'cover'])->find($id);
22     if (!$leardboard) {
23         return ApiReturn::success();
24     }
25     $lang = $leardboard->lang;
26     $leardboard->cover = CommonFnc::concatUrlStr($leardboard->cover);
27     $list = $leardboard->games()->with(['circle' => function ($query) {
28         $query->select(['id', 'circle_name', 'game_id', 'avatar', 'cover']);
29     }, 'game_labels' => function ($query) use ($lang) {
30         $query->where('lang', $lang);
31     }])->orderBy('sort', 'desc')->paginate($limit);
32     $list = isset($list) ? $list->toArray() : [];
33     $list['leardboard'] = $leardboard;
34     CommonFnc::paginateReturnUnset($list);
35     foreach ($list['data'] as $k => $v) {
36         $game = [];
37         $game['position'] = ($page - 1) * $limit + $k + 1;
38         $game['id'] = $v['id'];
39         $game['score'] = $v['score'];
40         $game['game_name'] = $v['game_name'];
41         if (!empty($v['circle'])) {
42             $v['circle']['is_attention'] = CircleService::isAttention($user_id, $v['circle']['id']);
43         }
44         $game['circle'] = $v['circle'] ?? [];
45         $game['circle_id'] = $v['circle']['id'] ?? 0;
46         $game['game_icon'] = CommonFnc::concatUrlStrAdmin($v['game_icon']);
47         $game_label = '';
48         foreach ($v['game_labels'] as $k1 => $v1) {
49             if ($k1 <= 2) {
50                 $game_label .= $v1['label_name'] . ' ';
51             }
52         }
53         $game['rec_text'] = '';
54         $game['game_label'] = rtrim($game_label, ' ');
55         $list['data'][$k] = $game;
56     }
57     return ApiReturn::success($list);
58 }
59

```

```

10 public function domesticGamesApprovals(): object
11 {
12     $list = DomesticGameApproval::query()
13         ->where('status', 1)
14         ->select(['id', 'title', 'cover', 'publication_date', 'link_url'])
15         ->orderBy('id', 'desc')
16         ->get()
17         ->toArray();
18     foreach ($list as $k => $v) {
19         $list[$k]['cover'] = CommonFnc::concatUrlStr($v['cover']);
20     }
21     return ApiReturn::success($list);
22 }
23
24 }
25

```

```

1 <?php
2
3 namespace App\Http\Controllers\api\v1;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\v1\SystemTask;
7 use App\Models\v1\UserDiamondLog;
8 use App\Service\v1\ApiCode;
9 use App\Service\v1\ApiReturn;
10 use App\Service\v1\Common;
11 use App\Service\v1\RedisKey;
12 use App\Service\v1\SystemTaskService;
13 use Illuminate\Http\Request;
14 use Illuminate\Support\Facades\DB;
15 use Illuminate\Support\Facades\Log;
16 use Illuminate\Support\Facades\Redis;
17 use Predis\Client;
18
19 class SystemTaskController extends Controller
20 {
21
22     /**
23      * 任务中心说明
24      * @return Object
25      */
26     public function systemTaskExplain()
27     {
28         try {
29             $redisConf = config('database.redis.default');
30             $client = new Client($redisConf);
31             $info = $client->get(env('SOLO_ADMIN_REDIS_PREFIX').RedisKey::SYSTEM_TASK_EXPLAIN);
32             return ApiReturn::success(['explain' => $info]);
33         } catch (\Throwable $throwable) {
34             Common::log('任务中心说明', $throwable);
35             return ApiReturn::error();
36         }
37     }
38 }

```

```
38
39  /**
40   * 任务中心钻石统计
41   * @param Request $request
42   * @return Object
43   */
44 public function diamondStatistic(Request $request)
45 {
46     $userId = $request->get('user_id');
47     try {
48         $statisticInfo = SystemTaskService::diamondStatistic($userId);
49         return ApiReturn::success($statisticInfo);
50     } catch (\Throwable $throwable) {
51
52         Common::log('任务中心钻石统计',$throwable);
53         return ApiReturn::error();
54     }
55 }
56
57 /**
58  * 收益/支出记录
59  * @param Request $request
60  * @return Object
61  */
62 public function diamondChangeLogs(Request $request)
63 {
64     $action = intval($request->get('action',1));
65     if(!in_array($action,[1,2])) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
66     try {
67         $userId = $request->get('user_id');
68         $page = $request->get('page',1);
69         $limit = $request->get('limit',10);
70         $page = ($page - 1) * $limit;
71         $where = [
72             'user_id' => $userId,
73             'action' => $action
74         ];
75         $field = ['source_id','type','describe','diamond','created_at'];
76         // $diamondLogs = (new UserDiamondLog())->diamondLogs($where,$field,$page,$limit);
77         $diamondLogs = SystemTaskService::diamondChangeLogs($where,$field,$page,$limit);
78         return ApiReturn::success($diamondLogs);
79     } catch (\Throwable $throwable) {
80         Common::log('收益/支出记录',$throwable);
81         return ApiReturn::error();
82     }
83 }
84
85 /**
86  * 签到奖励列表
87  * @param Request $request
88  * @return Object
89  */
90 public function signTasks(Request $request)
91 {
92 }
```

```
41     $userId = $request->get('user_id');
42     try {
43         $list = SystemTaskService::signTasks($userId);
44         return ApiReturn::success($list);
45     } catch (\Throwable $throwable) {
46         Common::log('签到任务',$throwable);
47         return ApiReturn::error();
48     }
49 }
50
```

```
1  /**
2   * 签到
3   * @param Request $request
4   * @return Object
5   */
6  public function signed(Request $request)
7  {
8      $awardId = intval($request->post('award_id',0));
9      $userId = $request->post('user_id');
10     Log::info('签到-入参,['userId' => $userId,'awardId' => $awardId]);
11
12     try {
13         DB::beginTransaction();
14         $re = $awardId > 0 ? SystemTaskService::signed($userId,$awardId) : SystemTaskService::autoSigned($userId);
15         if($re['status']) {
16             DB::commit();
17             Log::info('签到-status为true时返回值', $re);
18             return $awardId > 0 ? ApiReturn::success() : ApiReturn::success(['is_sign' => $re['is_sign'],'award' => $re['award']]);
19         }
20         DB::rollBack();
21         Log::info('签到-status为false时返回值',$re);
22         return ApiReturn::error($re['msg'],$re['code']);
23     } catch (\Throwable $throwable) {
24         DB::rollBack();
25         Common::log('签到动作',$throwable);
26         return ApiReturn::error();
27     }
28 }
29
30 /**
31 * 任务中心任务列表
32 * @param Request $request
33 * @return Object
34 */
35 public function systemTasks(Request $request)
36 {
37     $taskType = intval($request->get('task_type',0));
38     $userId = $request->get('user_id');
39     $page = $request->get('page',1);
40     $limit = $request->get('limit',10);
41     $page = ($page - 1) * $limit;
42     try {
43         $list = SystemTaskService::systemTasks($taskType,$userId,$page,$limit);
```

```

44         return ApiReturn::success($list);
45     } catch (\Throwable $throwable) {
46         Common::log('系统任务列表',$throwable);
47         return ApiReturn::error();
48     }
49 }
50

```

```

1  /**
2   * 任务中心领取奖励
3   * @param Request $request
4   * @return Object
5   */
6  public function getAward(Request $request)
7  {
8      $taskId = intval($request->get('task_id',0));
9      if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
10     $taskInfo = SystemTask::where(['id' => $taskId])>select('id','task_category','award')>first();
11     if(empty($taskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
12     $userId = $request->get('user_id');
13     try {
14         $response = SystemTaskService::getAward($userId,$taskInfo);
15         if($response) return ApiReturn::success();
16         return ApiReturn::error(ApiCode::SYSTEM_TASK_REPEAT_GET['msg'],ApiCode::SYSTEM_TASK_REPEAT_GET['code']);
17     } catch (\Throwable $throwable) {
18         Common::log('任务中心-领取奖励',$throwable);
19         return ApiReturn::error();
20     }
21 }
22
23 }
24

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Http\Controllers\Controller;
6  use App\Http\Requests\NewActivityCenterRequest;
7  use App\Service\v1\ApiCode;
8  use App\Service\v1\ApiReturn;
9  use App\Service\v1\Article;
10 use App\Service\v1\Code;
11 use App\Service\v1\Common;
12 use App\Service\v1\CommonFnc;
13 use App\Service\v1\Task2;
14 use Illuminate\Http\Request;
15 use Illuminate\Support\Facades\Log;
16 use Throwable;
17
18 /**
19  * 任务
20  */
21 class TaskController extends Controller
22 {

```



```

23  /**
24   * 任务列表
25   * @param Request $request
26   * @return object
27   */
28  public function list(Request $request): object
29  {
30      Log::info('获取任务列表入参', $request->all());
31      //获取userId
32      $userId = $request->get('user_id');
33      $limit = (int)$request->get('limit', 10);
34      $page = (int)$request->get('page', 1);
35      $taskStatus = (int)$request->get('task_status', 0);
36      $time = $request->server('REQUEST_TIME');
37      return Task2::list($userId, $time, $page, $limit, $taskStatus);
38  }
39
40  /**
41   * 任务详情
42   * @param Request $request
43   * @return Object
44   */
45  public function info(Request $request): object
46  {
47      $taskId = intval($request->get('task_id', 0));
48      $userId = $request->get('user_id', 0);
49      if(empty($taskId)){
50          return ApiReturn::rdata(ApiCode::TASK_ERROR);
51      }
52      $platform = $request->header('Platform');
53      Log::info('获取任务信息详情入参', $request->all());
54      $version = $request->header('Version');
55      return Task2::info($taskId, $userId, $platform, $version);
56  }
57
58  //参与任务
59  public function participating(NewActivityCenterRequest $request): object
60  {
61      Log::info('用户参与任务入参', $request->all());
62      $param = $request->post();
63      $param['action'] = $request->post('action', Code::ACTION_ADD);
64      $request->validate('participatingTask');
65      $ip = CommonFnc::getRealIP();
66      try {
67          $param['action'] = $request->post('action', Code::ACTION_ADD);
68          $param['status'] = $request->post('status', 0);
69          Log::info('参与任务==入口参数====', $param);
70          // 验证 设置
71          $err = Task2::checkParticipatingTask($param);
72          if ($err) {
73              Log::info('参与任务==校验失败====', $err);
74              return ApiReturn::rdata($err);
75          }
76      }
77  }

```

```

26 // 插入到 article 表的数据
27 $param['data'] = [
28     'cover' => $param['cover'],
29     'game_score' => empty($param['game_score']) ? 0 : $param['game_score'],
30     'check_status' => $param['check_status'] ?? Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
31     'content' => $param['content'],
32     // article 标题 截取
33     'title' => "",
34     'text_content' => $param['text_content'],
35     'content_len' => $param['content_len'],
36     'home_page' => $param['home_page'],
37     'label_list' => $param['label_list'],
38     'abstract' => $param['abstract'],
39     'user_id' => $param['user_id'],
40     'creator_source_id' => $param['creator_source_id'],
41     'task_article_status' => $param['task_article_status'],
42     'state' => 1,
43     'task_tab_id' => $param['task_tab_id']
44 ];
45 if ($param['action'] == Code::ACTION_ADD) {
46     // 新增需要的数据 (修改的时候不可更改的数据)
47     $param['data'] = array_merge($param['data'], [
48         'parent_id' => empty($param['parent_id']) ? 0 : $param['parent_id'],
49         'game_id' => $param['game_id'],
50         'task_id' => $param['task_id'],
51
52         'circle_id' => $param['circle_id'],
53         'type' => $param['type'],
54         'task_tab_id' => $param['task_tab_id'],
55         'task_article_status' => $param['task_article_status'],
56         'status' => $param['status'],
57         'is_writer_task' => $param['is_writer_task'],
58         'creator_source_id' => $param['creator_source_id']
59     ]);
60 }
61 Log::info('参与任务==进入数据库参数====', $param);
62 return Article::addNewArticle($param,$ip);
63 } catch (Throwable $throwable) {
64     Common::log('参与任务', $throwable);
65     return ApiReturn::error($throwable);
66 }
67 }
68
69 /**
70  * 新增/编辑收集地址
71  * @param NewActivityCenterRequest $request
72  * @return Object
73  */
74 public function saveCollectPlatformUrl(NewActivityCenterRequest $request): object
75 {
76     $params = $request->post('activity_center_platform_url', []);
77     if (is_string($params)) $params = json_decode($params, true); // ios 传惨问题
78     $articleId = (int)$request->get('article_id', 0);
79     if (empty($params)) {

```

```
29         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
30     }
31     $userId = $request->post('user_id');
32     return Task2::saveCollectPlatformUrl($userId, $articleId, $params);
33 }
34
35 /**
36  * 获取用户参与任务信息
37  * @param Request $request
38  * @return Object
39  */
40 public function userRecordInfo(Request $request): object
41 {
42     Log::info('获取用户参与任务信息详情入参', $request->all());
43     $user_id = $request->get('user_id', 0);
44     if (empty($user_id)) {
45         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
46     }
47     $task_id = $request->get('task_id', 0);
48     if (empty($task_id)) {
49         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
50     }
51
52     $article_id = $request->get('article_id', 0);
53     if(empty($article_id)){
54         return ApiReturn::error('未查询到您的任务参与信息');
55     }
56     $platform = $request->header('Platform');
57     $version = $request->header('Version');
58     return Task2::userRecordInfo($user_id, $task_id,$article_id,$platform,$version);
59 }
60
61 /**
62  * 用户参与记录
63  * @param Request $request
64  * @return Object
65  */
66 public function userRecords(Request $request): object
67 {
68     $user_id = $request->get('user_id');
69     if (empty($user_id)) {
70         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
71     }
72     $task_id = $request->get('task_id');
73     $limit = (int)$request->get('limit', 10);
74     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)]);
75     $platform = $request->header('Platform');
76     $version = $request->header('Version');
77     return Task2::userRecords($user_id, $limit,$task_id,$platform,$version);
78 }
79
80 /**
81  * 活动中心用户信息
82  * @param Request $request
```

```

32     * @return Object
33     */
34     public function getUserInfo(Request $request): object
35     {
36         $user_id = $request->get('user_id');
37         if (empty($user_id)) {
38             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
39         }
40         return Task2::getUserInfo($user_id);
41     }
42
43     /**
44     * 收藏/取消收藏任务
45     * @param Request $request
46     * @return object
47     */
48     public function toggleCollect(Request $request): object
49     {
50         $taskId = (int)$request->get('task_id', 0);

```

```

1         $userId = (int)$request->get('user_id', 0);
2         if(empty($userId)){
3             return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
4         }
5         if(empty($taskId)){
6             return ApiReturn::error('请选择要收藏的任务');
7         }
8         return Task2::toggleCollect($taskId, $userId);
9     }
10    //用户收藏列表
11    public function collectList(Request $request){
12        $userId = (int)$request->get('user_id', 0);
13        if(empty($userId)){
14            return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
15        }
16        $limit = (int)$request->get('limit', 10);
17        $time = $request->server('REQUEST_TIME');
18        return Task2::collectList($userId,$limit,$time);
19    }
20 }
21

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1;
4
5  use App\Biz\WechatUser;
6  use App\Events\LikeTaskEvent;
7  use App\Events\ParamErrorEvent;
8  use App\Events\PersonalTaskEvent;
9  use App\Http\Controllers\Controller;
10 use App\Http\Requests\Address;
11 use App\Http\Requests\v1\UserRequest;
12 use App\Jobs\ErrorLog;
13 use App\Models\Suggestion;

```

```

14 use App\Models\SuggestionImage;
15 use App\Models\UserBrowseLog;
16 use App\Models\UserCollect;
17 use App\Models\UserEvent;
18 use App\Models\UserModuleTab;
19 use App\Models\UserSocial;
20 use App\Models\ParamError;
21 use App\Models\v1\ApplicationAvatar;
22 use App\Models\v1\SearchKeywordLog;
23 use App\Models\v1\ServiceConfig;
24 use App\Models\v1\ShippingAddress;
25 use App\Models\v1\User as V1User;
26 use App\Models\v1\UserAlipush;
27 use App\Models\v1\UserGameSearchLog;
28 use App\Models\v1\UserLike;
29 use App\Models\v1\UserPushConfig;
30 use App\Models\v1\UserSearchLog;
31 use App\Models\v1\UserUserRole;
32 use App\Service\v1\AliContentAudit;
33 use App\Service\v1\AliPush;
34 use App\Service\v1\ApiCode;
35 use App\Service\v1\ApiReturn;
36 use App\Service\v1\Article;
37 use App\Service\v1\Code;
38 use App\Service\v1\Common;
39 use App\Service\v1\CommonFnc;
40 use App\Service\v1\RedisKey;
41 use App\Service\v1\SoloCode;
42 use App\Service\v1\task\ParamErrorTask;
43 use App\Service\v1\User;
44 use Dflydev\DotAccessData\Data;
45 use Hhxs5\LaravelS\Swoole\Task\Task;
46 use Illuminate\Auth\Access\AuthorizationException;
47 use Illuminate\Http\Request;
48 use Illuminate\Support\Facades\DB;
49 use Illuminate\Support\Facades\Log;
50 use Illuminate\Support\Facades\Redis;

```

```

1 use Illuminate\Support\Facades\Route;
2 use Illuminate\Validation\ValidationException;
3 use Psr\Container\ContainerExceptionInterface;
4 use Psr\Container\NotFoundExceptionInterface;
5
6 class UserController extends Controller
7 {
8     const EDIT_FIELD_TYPE = [
9         1 => 'avatar',
10        2 => 'username',
11        3 => 'description',
12        4 => 'sex',
13        5 => 'address_country_id',
14    ];
15
16    /**

```

```

17  * 我的页面个人信息
18  * @param Request $request
19  * @return Object
20  */
21  public function getPrivateUserInfo(Request $request)
22  {
23      $platform = $request->input('platform', 1);
24      $userId = $request->get('user_id');
25      try {
26          $userInfo = User::getPrivateUserInfo($userId);
27          return ApiReturn::success($userInfo);
28      } catch (\Throwable $throwable) {
29          Common::log('我的个人主页', $throwable);
30          return ApiReturn::error();
31      }
32  }
33
34  /**
35  * 个人主页用户信息
36  * @param Request $request
37  * @return Object
38  */
39  public function getUserInfo(Request $request)
40  {
41      $userId = $request->get('user_id');
42      $authorId = intval($request->get('author_id', 0));
43      try {
44          if ($authorId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
45          $userInfo = User::getUserInfo($userId, $authorId);
46          if (empty($userInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
47          return ApiReturn::success($userInfo);
48      } catch (\Throwable $throwable) {
49          Common::log('个人主页用户信息', $throwable);
50          return ApiReturn::error();
51      }
52  }
53
54  //增加用户喜欢社区
55  public function addUserLikeCircles(Request $request)
56  {
57      $userId = $request->post('user_id');
58      $circleIds = $request->post('circle_ids');
59      if (empty($circleIds) && !is_array($circleIds)) {
60          return ApiReturn::success();
61      }
62      //去除空数据
63      $circleIds = array_filter($circleIds);
64      //去重
65      $circleIds = array_unique($circleIds);
66      if (empty($circleIds)) {
67          return ApiReturn::success();
68      }
69      if (empty($userId)) {

```

```
20     return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
21 }
22     return User::addUserLikeCircles($userId, $circleIds);
23 }
24
25 /**
26  * 认证列表
27  * @param Request $request
28  * @return Object
29  */
30 public function getUserCertification(Request $request)
31 {
32     $userId = $request->get('user_id');
33     try {
34         $userInfo = V1User::find($userId);
35         $certification_jyt = UserUserRole::where(['user_id' => $userId, 'user_role_id' => 6])->exists();
36         $jyturl = '';
37         if (!$certification_jyt) {
38             $jyturl = ServiceConfig::where('field_name', 'certification_janyoutuan')->value('value');
39             if ($jyturl) $jyturl = 'solo://game:123/h5?url=' . $jyturl;
40         }
41         $data = [
42             'attestation_expert' => [
43                 'is_certification' => $userInfo->attestation_type == 2,
44                 'scheme' => "",
45             ],
46             'certification_jyt' => [
47                 'is_certification' => $certification_jyt,
48                 'scheme' => $jyturl,
49             ],
50             'show_attestation_expert' => (bool)ServiceConfig::where('field_name', 'show_attestation_expert')->value('value'),
51             'show_certification_jyt' => (bool)ServiceConfig::where('field_name', 'show_certification_jyt')->value('value')
52         ];
53     } catch (\Throwable $throwable) {
54         Common::log('个人主页用户信息', $throwable);
55         return ApiReturn::error();
56     }
57 }
58
59 /**
60  * 综合搜索记录
61  * @param Request $request
62  * @return Object
63  */
64 public function userSearchLogs(Request $request)
65 {
66     $userId = $request->get('user_id', 0);
67     $deviceNum = $request->get('device_num', '');
68     $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
69     try {
70         $searchLogs = UserSearchLog::where($where)->orderBy('id', 'desc')->limit(15)->pluck('keyword');
```

```
23         return ApiReturn::success($searchLogs);
24     } catch (\Throwable $throwable) {
25         Common::log('综合搜索记录', $throwable);
26         return ApiReturn::error();
27     }
28 }
29
30 /**
31  * 清除综合搜索历史记录
32  * @param Request $request
33  * @return Object
34  */
35 public function clearSearchLogs(Request $request)
36 {
37     $userId = $request->get('user_id', 0);
38     $deviceNum = $request->get('device_num', '');
39     $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
40     try {
41         SearchKeywordLog::where($where)->delete();
42         return ApiReturn::success();
43     } catch (\Throwable $throwable) {
44         Common::log('清除综合搜索历史记录', $throwable);
45         return ApiReturn::error();
46     }
47 }
48
49 /**
50  * 游戏搜索历史记录
51
52 1  * @param Request $request
53 2  * @return Object
54 3  */
55 4 public function userGameSearchLogs(Request $request)
56 5 {
57 6     $userId = $request->get('user_id', 0);
58 7     $deviceNum = $request->get('device_num', '');
59 8     $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
60 9     try {
61 10         $searchLogs = UserGameSearchLog::where($where)->orderBy('id', 'desc')->limit(15)->pluck('keyword');
62 11         return ApiReturn::success($searchLogs);
63 12     } catch (\Throwable $throwable) {
64 13         Common::log('我的游戏搜索记录', $throwable);
65 14         return ApiReturn::error();
66 15     }
67 16 }
68
69 17
70 18 /**
71 19  * 清除游戏搜索历史记录
72 20  * @param Request $request
73 21  * @return Object
74 22  */
75 23 public function clearGameSearchLogs(Request $request)
76 24 {
77 25     $userId = $request->get('user_id', 0);
```



```

26     $deviceNum = $request->get('device_num', '');
27     $where = $userId > 0 ? ['user_id' => $userId] : ['device_num' => $deviceNum];
28     try {
29         UserGameSearchLog::where($where)->delete();
30         return ApiReturn::success();
31     } catch (\Throwable $throwable) {
32         Common::log('清除我的游戏搜索记录', $throwable);
33         return ApiReturn::error();
34     }
35 }
36
37 /**
38  * 编辑用户信息
39  * @param Request $request
40  * @return Object
41  */
42 public function editUserInfo(Request $request)
43 {
44     try {
45         $type = $request->post('type', 0);
46         $keyword = trim($request->post('keyword', null));
47         if (!isset(static::EDIT_FIELD_TYPE[$type]) || is_null($keyword)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
48             ApiCode::PARAM_ERROR['code']);
49         $userId = $request->post('user_id');
50         if ($type == 1) {
51             if (ApplicationAvatar::where(['user_id' => $userId, 'check_status' => 1])->exists()) return
52             ApiReturn::error(ApiCode::AVATAR_ERROR['msg'], ApiCode::AVATAR_ERROR['code']);
53
54             $imageStatus = AliContentAudit::imageAudit($keyword);
55             if ($imageStatus == SoloCode::ALI_AUDIT_BLOCK) return ApiReturn::error(ApiCode::ALI_IMAGE_AUDIT['msg'],
56                 ApiCode::ALI_IMAGE_AUDIT['code']);
57             $avatarData = [
58                 'user_id' => $userId,
59                 'check_status' => $imageStatus == SoloCode::ALI_AUDIT_PASS ? 3 : 1,
60                 'avatar' => $keyword,
61             ];
62             ApplicationAvatar::create($avatarData);
63             if ($imageStatus == SoloCode::ALI_AUDIT_BLOCK) return ApiReturn::success();
64         } else if ($type == 2) {
65             $str = Redis::get(RedisKey::NICKNAME_FILTER);
66             if (empty($keyword)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
67             $arr = explode(' ', $str);
68             if (in_array($keyword, $arr)) return ApiReturn::error(ApiCode::NICKNAME_ERROR['msg'],
69                 ApiCode::NICKNAME_ERROR['code']);
70         }
71         if ($type == 2 || $type == 3) {
72             $x = request()->header('X-Requested-From');
73             if (!empty($x) && $x == 'MiniProgram') {
74                 $userInfo = V1User::query()->where('id', $userId)->first();
75                 if (empty($userInfo->xcx_openid)) {
76                     return ApiReturn::error(ApiCode::LOGIN_ERROR['msg'], ApiCode::LOGIN_ERROR['code']);
77                 }
78             }
79             //小程序
80             $res = (new WechatUser())->checkText($keyword, 1, $userInfo['xcx_openid']);

```

```

25         if (!empty($res)) {
26             return ApiReturn::error($res);
27         }
28     }
29 }
30 V1User::where(['id' => $userId])->update([static::EDIT_FIELD_TYPE[$type] => $keyword]);
31 if (V1User::find($userId)) {
32     V1User::find($userId)->searchable();
33 }
34 PersonalTaskEvent::dispatch($userId);
35 return ApiReturn::success();
36 } catch (\Throwable $throwable) {
37     Common::log('编辑个人资料', $throwable);
38     return ApiReturn::error();
39 }
40 }
41
42 /**
43  * 修改用户个人中心背景图
44  * @param Request $request
45  * @return object
46  */
47 public function updateBackground(Request $request): object
48 {
49     $userId = $request->post('user_id');
50     if (empty($userId)) {
51
52         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
53     }
54     $backgroundUrl = $request->post('background_url');
55     if (empty($backgroundUrl)) {
56         return ApiReturn::rdata(ApiCode::PARAM_ERROR);
57     }
58     return User::updateBackground($userId, $backgroundUrl);
59 }
60
61 /**
62  * 钻石数量
63  * @param Request $request
64  * @return Object
65  */
66 public function diamonds(Request $request)
67 {
68     $userId = $request->get('user_id');
69     try {
70         $diamondNum = V1User::where(['id' => $userId])->value('diamond_num');
71         return ApiReturn::success(['diamond_num' => (int)$diamondNum]);
72     } catch (\Throwable $throwable) {
73         Common::log('我的个人主页', $throwable);
74         return ApiReturn::error();
75     }
76 }
77 }

```

```

28  /**
29  * 新增收货地址
30  * @param Address $request
31  * @return Object
32  * @throws \Illuminate\Auth\Access\AuthorizationException
33  * @throws \Illuminate\Validation\ValidationException
34  */
35  public function addShippingAddress(Address $request)
36  {
37      $request->validate();
38      $param = $request->post();
39      Log::info('新增收货地址入参==', $param);
40      try {
41          if ($param['default_address']) {
42              ShippingAddress::where(['user_id' => $param['user_id'], 'default_address' => 1])->update(['default_address' => 0]);
43          }
44          $response = ShippingAddress::create($param);
45          Log::info('新增收货地址成功');
46          return ApiReturn::success($response);
47      } catch (\Throwable $throwable) {
48          Common::log('新增收货地址', $throwable);
49          return ApiReturn::error();
50      }
}

1  }
2
3  /**
4  * 编辑我的收货地址
5  * @param Address $request
6  * @return Object
7  * @throws \Illuminate\Auth\Access\AuthorizationException
8  * @throws \Illuminate\Validation\ValidationException
9  */
10 public function editShippingAddress(Address $request)
11 {
12     $request->validate();
13     $param = $request->post();
14     Log::info('修改收货地址入参==', $param);
15     try {
16         $addressId = intval($request->post('id', 0));
17         if ($addressId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
18         $addressInfo = ShippingAddress::where(['id' => $addressId, 'user_id' => $param['user_id']])->first();
19         if (empty($addressInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
20         if ($param['default_address']) {
21             ShippingAddress::where(['user_id' => $param['user_id'], 'default_address' => 1])->update(['default_address' => 0]);
22         }
23         $addressInfo->name = $param['name'];
24         $addressInfo->mobile = $param['mobile'];
25         $addressInfo->province_id = $param['province_id'];
26         $addressInfo->city_id = $param['city_id'];
27         $addressInfo->district_id = $param['district_id'];
28         $addressInfo->detailed_address = $param['detailed_address'];
29         $addressInfo->default_address = $param['default_address'];
30         $addressInfo->save();

```

```
31     Log::info('修改收货地址成功');
32     return ApiReturn::success($addressInfo);
33 } catch (\Throwable $throwable) {
34     Common::log('修改收货地址报错', $throwable);
35     return ApiReturn::error();
36 }
37 }
38
39
40 /**
41  * 我到收货地址列表
42  * @param Request $request
43  * @return Object
44  */
45 public function shippingAddresses(Request $request)
46 {
47     try {
48         $userId = $request->get('user_id');
49         $address = ShippingAddress::query()
50             ->where('user_id', $userId)
51
1   ->select('id', 'province_id', 'city_id', 'district_id', 'detailed_address', 'name', 'mobile', 'default_address')
2   ->get()->toArray();
3         return ApiReturn::success($address);
4     } catch (\Throwable $throwable) {
5         Common::log('我的收货地址列表', $throwable);
6         return ApiReturn::error();
7     }
8 }
9
10 /**
11  * 设置默认地址
12  * @param Request $request
13  * @return Object
14  */
15 public function setDefaultAddress(Request $request)
16 {
17     try {
18         DB::beginTransaction();
19         $addressId = intval($request->post('address_id', 0));
20         if ($addressId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
21         $userId = $request->post('user_id');
22         $addressInfo = ShippingAddress::where(['user_id' => $userId, 'id' => $addressId])->first();
23         if (empty($addressInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
24         if (empty($addressInfo->default_address)) {
25             ShippingAddress::where(['user_id' => $userId, 'default_address' => 1])->update(['default_address' => 0]);
26             $addressInfo->default_address = 1;
27             $addressInfo->save();
28         }
29         DB::commit();
30         return ApiReturn::success();
31     } catch (\Throwable $throwable) {
32         DB::rollBack();
33         Common::log('设置默认地址', $throwable);
```

```

34         return ApiReturn::error();
35     }
36 }
37
38 /**
39  * 点赞/取消点赞
40  * @param Request $request
41  * @return Object
42  */
43 public function userLike(Request $request)
44 {
45     try {
46         $relevancId = (int)$request->post('relevance_id', 0);
47         $type = (int)$request->post('type', 1);
48         $authorId = intval($request->post('author_id', 0));
49         $platform = $request->header('platform');
50         $versionCode = $request->header('version-code');

```

```

1         if ($platform == 2 && $versionCode == '3.0.0') {
2             $rid = $relevancId;
3             $relevancId = $authorId;
4             $authorId = $rid;
5         }
6         Log::info("用户点赞和取消点赞入参", ['relevancId' => $relevancId, 'type' => $type, 'authorId' => $authorId]);
7         if ($relevancId <= 0 || $authorId <= 0 || !in_array($type, SoloCode::LIKE_TYPE)) {
8             ParamErrorEvent::dispatch($request, '点赞/取消点赞');
9             return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
10        }
11        $userId = $request->post('user_id');
12        // if (Redis::exists(RedisKey::CAN_LIKE_TIME . $userId)) return ApiReturn::error('操作太频繁，请稍后再试...');
13        // Redis::set(RedisKey::CAN_LIKE_TIME . $userId, 1, 'PX', 300, 'NX');
14        // $isCanLike = User::canLike($userId);
15        // if(empty($isCanLike)) return ApiReturn::error('操作太频繁，请稍后再试...');
16        $response = User::userLike($userId, $relevancId, $type, $authorId);
17        if ($response == 1) LikeTaskEvent::dispatch($relevancId, $userId, $type);
18        return ApiReturn::success(['status' => $response], $response == 1 ? '点赞成功' : '取消点赞');
19    } catch (\Throwable $throwable) {
20        Common::log('点赞', $throwable);
21        return ApiReturn::error();
22    }
23 }
24
25 public function robotLogin(UserRequest $request)
26 {
27     $uid = $request->post('uid');
28     $userInfo = V1User::query()->where('is_robot', '!=', 0)->where('id', $uid)->first();
29     if (!$userInfo) {
30         return ApiReturn::error('当前用户非机器人');
31     }
32     $api_token = $userInfo->api_token;
33     if (!Redis::exists('token:' . $api_token)) {
34         //如果请求接口返回请先登录则去生成一个token并存入redis中
35         $api_token = CommonFnc::generateToken();
36         //如果登录过期时间一周

```

```

37         Redis::setex('token:' . $api_token, 5184000, $userInfo['id']);
38         V1User::query()->where('id', $userInfo->id)->update(['api_token' => $api_token]);
39     }
40     return ApiReturn::success(['api_token' => $api_token]);
41 }
42
43 /**
44  * 设置用户活页时间
45  * @param Request $request
46  * @return Object
47  */
48 public function userVisitTime(Request $request)
49 {
50     $userId = $request->get('user_id');

```

```

1     if ($userId > 0) {
2         V1User::query()->where(['id' => $userId])->update(['visit_time' => date('Y-m-d H:i:s')]);
3         UserEvent::query()
4             ->insert([
5                 'user_id' => $userId,
6                 'created_at' => date('Y-m-d H:i:s'),
7                 'type' => UserEvent::TYPE['visit']
8             ]);
9     }
10    return ApiReturn::success();
11 }
12
13 /**
14  * 新增修改认证信息
15  * @catalog
16  * @title 新增修改认证信息
17  * @description 描述信息
18  * @method post
19  * @url /api/user/authentication
20  * @param idcard 必选 string 身份证号码
21  * @param name 必选 string 姓名
22  * @return {"code":200,"message":"ok"}
23  */
24 public function authentication(UserRequest $request)
25 {
26     $userId = $request->post('user_id');
27     $request->validate('authentication');
28     $data = $request->validated();
29     try {
30         if (V1User::where('idcard', $data['idcard'])->where('id', '<>', $userId)->first()) return
ApiReturn::rdata(ApiCode::HAS_BIND_ADCARD);
31         $rs = V1User::where('id', $userId)->update($data);
32         $age = User::getAge($data['idcard']);
33         //获取用户是否为青少年
34         $res['is_teenager'] = $age <= 18;
35         if ($rs) return ApiReturn::success($res);
36         else return ApiReturn::error();
37     } catch (\Exception $th) {
38         Common::log('实名认证', $th);

```

```

39         return ApiReturn::error();
40     }
41 }
42
43 /**
44  * 获取用户实名信息
45  * @catalog
46  * @title 获取用户实名信息
47  * @description 获取用户实名信息
48  * @method get
49  * @url /api/user/getAuthenticationInfo
50  * @return {"code":200,"message":"ok","data":{"idcard":8,"name": "姓名"}}

```

```

1  * @return_param idcard string 身份证号码
2  * @return_param name string 姓名
3  */
4  public function getAuthenticationInfo(Request $request)
5  {
6      $userId = $request->post('user_id');
7      try {
8          $info = V1User::find($userId);
9          if (!$info) return ApiReturn::error();
10         return ApiReturn::success([
11             'idcard' => $info->idcard,
12             'name' => $info->name,
13         ]);
14     } catch (\Exception $th) {
15         Common::log('实名认证详情', $th);
16         return ApiReturn::error();
17     }
18 }
19
20 /**
21  * 用户推送设置
22  * @catalog
23  * @title 用户推送设置
24  * @description 描述信息
25  * @method get
26  * @url /api/user/pushConfigs
27  * @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}}
28  * @return_param id int 配置id
29  * @return_param user_id int 用户id
30  * @return_param comment_status boolean 评论开关 : true 开, false 关
31  * @return_param like_status boolean 点赞/收藏开关 : true 开, false 关
32  * @return_param attention_status boolean 新增关注开关 : true 开, false 关
33  * @return_param remind_status boolean 提醒(被@)开关 : true 开, false 关
34  * @return_param system_status boolean 系统开关 : true 开, false 关
35  */
36  public function pushConfigs(Request $request)
37  {
38      $userId = $request->post('user_id');
39      try {
40          $pushConfig = UserPushConfig::query()->firstOrCreate(['user_id' => $userId], [

```

```

41         'comment_status' => 1,
42         'like_status' => 1,
43         'attention_status' => 1,
44         'remind_status' => 1,
45         'system_status' => 1,
46     ]);
47     return ApiReturn::success($pushConfig);
48 } catch (\Throwable $throwable) {
49     Common::log('获取用户推送配置信息', $throwable);
50     return ApiReturn::error();
}

1     }
2 }
3
4 /**
5  * 获取用户tap和qq的绑定信息
6  * @param Request $request
7  * @return Object
8  */
9 public function getPlatformTapAndQqBinding(Request $request): object
10 {
11     $user_id = $request->get('user_id');
12     if (empty($user_id)) {
13         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
14     }
15     return User::getPlatformTapAndQqBinding((int)$user_id);
16 }
17
18 /**
19  * 绑定tap账号和qq
20  * @param Request $request
21  * @return object
22  */
23 public function bindPlatformTapAndQq(Request $request): object
24 {
25     $user_id = $request->post('user_id');
26     if (empty($user_id)) {
27         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
28     }
29     $qq_account = $request->post('qq_account');
30     $tap_account = $request->post('tap_account');
31     if (empty($tap_account)) {
32         return ApiReturn::error('您还没有填写哦~');
33     }
34     if (empty($tap_account['platform_user_id'])) {
35         return ApiReturn::error('您还没有填写哦~');
36     }
37     return User::bindPlatformTapAndQq($user_id, $tap_account, $qq_account);
38 }
39 /**
40  * 小程序code换unionId
41  *
42  * @param [type] $code
43  */

```



```
44 // "openid": "o6_bmjrPTlm6_2sgVt7hMZOPxxxx",
45 // "session_key": "tiihtNczf5v6AKRyJwExxx=",
46 // "unionid": "o6_bmasdasdsad6_2sgVt7hMZOxxxx",
47 // "errcode": 0,
48 // "errmsg": "ok"
49 public function miniAppLoginSession($code): object
50 {
1   try {
2       $app = new WechatUser();
3       $result = $app->code2Session($code);
4       return ApiReturn::success($result);
5   } catch (\Throwable $th) {
6       //throw $th;
7       info($th);
8       return ApiReturn::error();
9   }
10 }
11
12 //微信小程序登录
13 public function wechatMiniProgramLogin(UserRequest $request): object
14 {
15     try {
16         $code = $request->post('code');
17         if (empty($code)) {
18             return ApiReturn::error('参数错误');
19         }
20         $app = new WechatUser();
21         $result = $app->code2Session($code);
22         if (empty($result['openid']) || empty($result['session_key']) || empty($result['unionid'])) {
23             return ApiReturn::error('登录失败');
24         }
25         $isApp = $request->post('is_app', true);
26         $data = [
27             'union_wechat_open_id' => $result['unionid'],
28             'wechat_open_id' => $result['openid']
29         ];
30         $where = ['wechat_open_id' => $data['union_wechat_open_id']];
31         $userInfo = V1User::query()->where($where)->first();
32         if (!$userInfo) {
33             $userInfo = V1User::query()->where('wechat_open_id', $data['wechat_open_id'])->first();
34             if ($userInfo) V1User::query()->where('wechat_open_id', $data['wechat_open_id'])->update(['wechat_open_id' =>
35                 $data['union_wechat_open_id']]); // 更换 openid 为 unionid
36         }
37         if ($userInfo && empty($userInfo->xcx_openid)) {
38             V1User::query()->where('id', $userInfo->id)->update(['xcx_openid' => $data['wechat_open_id']]);
39         }
40         if ($userInfo && $userInfo->account_status == 2) {
41             return ApiReturn::rdata(ApiCode::ACCOUNT_FREEZE);
42         }
43         // 添加/修改登录数据
44         $time = time();
45         $ip = Request::createFromGlobals()->getClientIp();
46         // 生成 token
```

```

46     while (true) {
47         $token = CommonFnc::generateToken();
48         if (!Redis::exists('token' . $token)) break;
49     }
50     $rdata = [
1
        'token' => $token
2
    ];
3
    // 查询是否 有该账户 没有 则 创建
4
    if ($userInfo) {
5
        $rdata['avatar'] = $userInfo['avatar'];
6
        $rdata['username'] = $userInfo['username'];
7
        $rdata['uid'] = $userInfo['id'];
8
        $rdata['mobile'] = $userInfo['mobile'];
9
        $rdata['wx'] = $isApp ? (bool)$userInfo['app_wx_openid'] : (bool)$userInfo['wechat_open_id']; // 是否绑定 wx
10
        $rdata['qq'] = (bool)$userInfo['qq_open_id']; // 是否绑定 qq
11
        $rdata['idcard'] = (bool)$userInfo['idcard']; // 是否认证身份
12
        $rdata['qq_num'] = (bool)$userInfo['qq']; // 是否填写qq号码
13
        $rdata['is_binding_mobile'] = true;
14
        $this->roles($rdata);
15
        $userInfo = $userInfo->toArray();
16
17
        Redis::setex('token:' . $token, Code::TOKEN_EXPIRE_TIME, $userInfo['id']);
18
        V1User::query()->where('id', $userInfo['id'])->update(['login_time' => $time, 'login_ip' => $ip, 'api_token' =>
$token]);
19
        // 存入redis 数据
20
        $userInfo['login_time'] = $time;
21
        $userInfo['login_ip'] = $ip;
22
        $userInfo['api_token'] = $token;
23
        // 存入 redis
24
        (new V1User())->setUserInfoToRedis($userInfo['id'], $userInfo);
25
        $rdata['user_role'] = $userInfo['user_role'];
26
    } else {
27
        $nickname = '用户' . CommonFnc::randomNumeric(10);
28
        $idata = [];
29
        $data = array_merge([
30
            'username' => $nickname,
31
            'login_time' => $time,
32
            'login_ip' => $ip,
33
            'is_robot' => 0, //是否为机器人
34
            'api_token' => $token,
35
            'xcx_openid' => $data['wechat_open_id']
36
        ], $idata);
37
        // 默认头像
38
        $data['avatar'] = CommonFnc::defaultAvatar();
39
        $rs = V1User::query()->firstOrCreate($where, $data)->toArray();
40
        $userInfo = V1User::query()->find($rs['id'])->toArray();
41
        Redis::setex('token:' . $token, Code::TOKEN_EXPIRE_TIME, $rs['id']);
42
        $rdata['avatar'] = $userInfo['avatar'];
43
        $rdata['username'] = $userInfo['username'];
44
        $rdata['uid'] = $userInfo['id'];
45
        $rdata['mobile'] = $userInfo['mobile'];
46
        $rdata['wx'] = (bool)$userInfo['wechat_open_id']; // 是否绑定 wx
47
        $rdata['qq'] = (bool)$userInfo['qq_open_id']; // 是否绑定 qq
    
```

```

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

```

```

1      $this->roles($rdata);
2      // 查询新数据 存入 redis
3      (new V1User()->setUserInfoToRedis($userInfo['id'], $userInfo);
4      $rdata['user_role'] = $userInfo['user_role'];
5      }
6      self::updateSessionKey($rdata['uid'], $result['session_key'], $result['openid']);
7      $rdata['wx_login_info'] = [
8          'openid' => $result['openid'],
9          'unionid' => $result['unionid']
10     ];
11     return ApiReturn::success($rdata);
12 } catch (\Throwable $th) {
13     Common::log('微信小程序登录异常', $th);
14     return ApiReturn::error();
15 }
16 }
17
18 //更新微信sessionKey
19 public static function updateSessionKey($user_id, $session_key, $openid)
20 {
21     if (!empty($user_id) && !empty($session_key)) {
22         UserSocial::query()->updateOrCreate(['user_id' => $user_id, 'type' => 'wechat_mini_program', 'app_id' =>
23             'wxd0215facf5f99001'], ['session_key' => $session_key, 'openid' => $openid]);
24     }
25 }
26 /**
27  * 用户角色信息
28  * @param array $data
29  */
30 public function roles(array &$data)
31 {
32     $re = UserUserRole::query()->has('user_role')->where(['user_id' => $data['uid']])->get();
33     $data['roles']['is_channel'] = false;
34     foreach ($re as $v) {
35         if ($v->user_role_id == 7) {
36             $data['roles']['is_channel'] = true;
37             break;
38         }
39     }
40 }
41
42 /**
43  * 个人中心功能区
44  */
45 public function getUserFunctionTab(Request $request)
46 {
47     $platform = $request->get('platform');
48     try {
49         $unButtons = [];

```

```

50      $where['status'] = 1;

1      $config = ServiceConfig::query()
2      ->whereIn('field_name', [
3          'ios_show',
4          'android_show',
5          'ios_user_disable_module',
6          'android_user_disable_module'
7      ])->pluck('value', 'field_name');
8      if ($platform == 2) { // ios 模块
9          if (!$config['ios_show']) {
10             $unButtons = json_decode($config['ios_user_disable_module'], true);
11         }
12         $where['ios_status'] = 1;
13     } else if ($platform == 1) { // 安卓模块
14         if (!$config['android_show']) {
15             $unButtons = json_decode($config['android_user_disable_module'], true);
16         }
17         $where['android_status'] = 1;
18     }
19     $data = UserModuleTab::query()
20     ->where($where)
21     ->with(['buttons' => function ($q) use ($unButtons, $where) {
22         if ($unButtons) $q->whereNotIn('link', $unButtons);
23         $q->where($where)->select('id', 'tab_id', 'icon', 'title', 'link', 'link', 'weblink', 'desc')->orderBy('sort')->orderBy('id');
24     }])->select('id', 'tab_type', 'tab_name')->orderBy('sort')->orderBy('id')->get();
25     return ApiReturn::success($data);
26 } catch (\Throwable $e) {
27     return ApiReturn::error();
28 }
29 }

30
31 /**
32  * 用户认证页面
33  * @param Request $request
34  * @return Object
35  */
36 public function userAttestation(Request $request): object
37 {
38     $userId = $request->get('user_id', 0);
39     try {
40         $userAttestationConfigs = User::userAttestation($userId);
41         return ApiReturn::success($userAttestationConfigs);
42     } catch (\Exception $e) {
43         return ApiReturn::error();
44     }
45 }

46
47 /**
48  * 意见反馈
49  * @param UserRequest $request
50  * @return Object

1      * @throws AuthorizationException
2      * @throws ValidationException

```

```

3      */
4      public function submitSuggestion(UserRequest $request): object
5      {
6          $uid = $request->post('user_id');
7          $content = $request->post('suggestion_content', '');
8          $contact = $request->post('contact', '');
9          if ($content == null) $content = '';
10         if ($contact == null) $contact = '';
11         $images = $request->post('images', []);
12         $images = is_array($images) ? $images : json_decode($images, true);
13         if ($content) {
14             $request->validate('submitSuggestion');
15         } else if (!$images) {
16             return ApiReturn::error('请上传图片或内容');
17         }
18         try {
19             DB::beginTransaction();
20             $rs = Suggestion::query()->create([
21                 'user_id' => $uid,
22                 'content' => $content,
23                 'contact' => $contact
24             ]);
25             if ($images) {
26                 // 维护时间
27                 $time = time();
28                 $arr = [];
29                 foreach ($images as $k => $v) {
30                     $arr [] = [
31                         'suggestion_id' => $rs->id,
32                         'url' => $v,
33                         'created_at' => $time,
34                         'updated_at' => $time,
35                         'sort' => ($k + 1)
36                     ];
37                 }
38                 // 图片限制 截取 前n个图片
39                 $arr = array_slice($arr, 0, 5);
40                 SuggestionImage::query()->insert($arr);
41             }
42             DB::commit();
43             return ApiReturn::success((bool)$rs);
44         } catch (\Exception $e) {
45             DB::rollback();
46             return ApiReturn::error();
47         }
48     }
49
50     //用户是否设置青少年模式密码

```

```

1      public function isSetTeenModePas(Request $request): object
2      {
3          $user_id = $request->post('user_id');
4          $deviceId = $request->post('device_num');
5          if (empty($deviceId) && empty($user_id)) {

```

```

6         return ApiReturn::success();
7         return ApiReturn::error('参数错误');
8     }
9     return User::isSetTeenModePas($user_id, $deviceId);
10 }
11
12 //校验用户青少年密码是否正确
13 public function checkTeenModePas(Request $request): object
14 {
15     return ApiReturn::success();
16     $user_id = $request->post('user_id');
17     $deviceId = $request->post('device_num');
18     if (empty($deviceId) && empty($user_id)) {
19         return ApiReturn::error('参数错误');
20     }
21     $password = $request->post('password');
22     if (empty($password)) {
23         return ApiReturn::error('请输入密码');
24     }
25     if (!preg_match('/^[0-9]{4}$/', $password)) {
26         return ApiReturn::error('密码必须是4位数字');
27     }
28     return User::checkTeenModePas($user_id, $deviceId, $password);
29 }
30
31 //充值青少年模式密码
32 public function resetTeenModePass(Request $request): object
33 {
34     $deviceId = $request->post('device_num');
35     $user_id = $request->post('user_id');
36     $name = $request->post('name');
37     if (empty($name)) {
38         return ApiReturn::error('请输入姓名');
39     }
40     $id_card = $request->post('id_card');
41     if (empty($id_card)) {
42         return ApiReturn::error('请输入身份证号');
43     }
44     if (!Common::checkIdCard($id_card)) {
45         return ApiReturn::error('身份证号格式不正确');
46     }
47     $age = User::getAge($id_card);
48     if ($age < 18) {
49         return ApiReturn::error('用户必须年满 18 岁');
50     }
51 }
52
53 $password = $request->post('password');
54 if (empty($password)) {
55     return ApiReturn::error('请输入密码');
56 }
57 if (!preg_match('/^[0-9]{4}$/', $password)) {
58     return ApiReturn::error('密码必须是4位数字');
59 }
60 if (empty($deviceId) && empty($user_id)) {

```

```
9         return ApiReturn::success();
10        return ApiReturn::error('参数错误');
11    }
12    return User::resetTeenModePass($user_id, $deviceId, $password, $name, $id_card);
13 }
14
15 //设置模式
16 public function setTeenMode(Request $request): object
17 {
18     $status = $request->post('status');
19     if (!in_array($status, [0, 1])) {
20         return ApiReturn::success();
21         return ApiReturn::error('参数错误');
22     }
23     $password = $request->post('password');
24     $type = $request->post('type');
25     if ($status == 0 && empty($type)) {
26         if (empty($password)) {
27             Log::info('设置青少年模式请输入密码');
28             return ApiReturn::error('请输入密码');
29         }
30         //密码必须是4位数字
31         if (!preg_match('/^[0-9]{4}$/', $password)) {
32             return ApiReturn::error('密码必须是4位数字');
33         }
34     }
35     $deviceId = $request->post('device_num');
36     $user_id = $request->post('user_id');
37     if (empty($deviceId) && empty($user_id)) {
38         return ApiReturn::success();
39         return ApiReturn::error('参数错误');
40     }
41     return User::setTeenMode($deviceId, $user_id, $status, $password);
42 }
43
44 //检验青少年模式身份证号
45 public function checkTeenModeIdCard(Request $request): object
46 {
47     $name = $request->post('name');
48     if (empty($name)) {
49         return ApiReturn::error('请输入姓名');
50     }
51
52     $id_card = $request->post('id_card');
53     if (empty($id_card)) {
54         return ApiReturn::error('请输入身份证号');
55     }
56     if (!Common::checkIdCard($id_card)) {
57         return ApiReturn::error('身份证号格式不正确');
58     }
59     $age = User::getAge($id_card);
60     if ($age < 18) {
61         return ApiReturn::error('用户必须年满 18 岁');
62     }
63 }
```

```

12     return ApiReturn::success();
13 }
14
15 /**
16  * 历史记录
17  * @param Request $request
18  * @return Object
19  */
20 public function getUserBrowseList(Request $request): object
21 {
22     $limit = $request->get('limit', 10);
23     $uid = $request->get('user_id');
24     $is_review = intval($request->get('is_review', 0));
25     $deviceId = $request->get('device_num');
26     return User::getUserBrowseList($uid, $is_review, $limit, $deviceId);
27 }
28
29 /**
30  * 合并资源
31  * @param array $articleList
32  * @param $user_id
33  * @return array
34  * @throws ContainerExceptionInterface
35  * @throws NotFoundExceptionInterface
36  */
37 public static function handleResorce($articleList, $user_id, $platform): array
38 {
39     $data = [];
40     if (!empty($articleList['data'])) {
41         foreach ($articleList['data'] as $k => $v) {
42             $article = $v['article'];
43             $article['label_list'] = empty($v['article']['label_list']) ? [] : json_decode($v['article']['label_list'], true);
44             $article['is_like'] = UserLike::query()
45                 ->where('relevance_id', $v['article']['id'])
46                 ->where('user_id', $user_id)
47                 ->where('type', $v['article']['type'])
48                 ->where('status', 1)
49                 ->exists();
50             $article['is_no_like'] = 0;

```

```

1         if (isset($v['article']['user_award'])) {
2             $article['is_awarded'] = $v['article']['user_award']['task_award_id'] > 0;
3         }
4         if (isset($v['article']['create_time'])) {
5             $article['create_time'] = CommonFnc::disposeDateTime(strtotime($v['article']['create_time']));
6         }
7         $article['content'] = $v['article']['text_content'] ? mb_substr($v['article']['text_content'], 0, 255) : '';
8         unset($article['text_content']);
9         // ios 要不同的数据格式
10        if ($platform == 2) {
11            $article['video'] = empty($article['video']['url']) ? '' : $v['article']['video']['url'];
12            $article['images'] = empty($article['images']) ? [] : array_column($v['article']['images'], 'url');
13            foreach ($article['images'] as $k1 => $v1) {
14                $article['images'][$k1] = Common::concatUrlStr($v1);

```



```

15         }
16         if (!empty($v['article']['video']) && strpos($v['article']['cover'], 'sologame/webapp/transparent.png')) {
17             $article['cover'] = $v['article']['video'] . '?x-oss-process=video/snapshot,t_1000,m_fast';
18         }
19     } else {
20         if (!empty($v['article']['video']) && !empty($v['article']['video']['url']) && strpos($v['article']['cover'],
'sologame/webapp/transparent.png')) {
21             $article['cover'] = $v['article']['video']['url'] . '?x-oss-process=video/snapshot,t_1000,m_fast';
22         }
23         if (!$article['video']) $v['article']['video'] = (object)[];
24         foreach ($v['article']['images'] as $k1 => $v1) {
25             $article['images'][$k1] = [
26                 'article_id' => $article['id'],
27                 'url' => Common::concatUrlStr($v1['url']),
28             ];
29         }
30     }
31     if (empty($v['article']['game'])) {
32         $article['game'] = (object)[];
33     }
34     if (empty($v['article']['circle'])) {
35         $article['circle'] = (object)[];
36     }
37     if (isset($v['article']['rss_guid'])) {
38         $article['content'] = $v['article']['title'];
39     }
40     $data[] = $article;
41 }
42 }
43 return $data;
44 }
45
46 /**
47  * 差评/取消差评
48  * @param Request $request
49  * @return Object
50  * @throws \Throwable

```

```

1  */
2  public function addUserNoLike(Request $request): object
3  {
4      $authorId = intval($request->post('author_id', 0));
5      $relevancId = intval($request->post('relevance_id', 0));
6      $type = intval($request->post('type', 1));
7      $userId = $request->post('user_id');
8      try {
9          if (Redis::exists('user:nolike:' . $userId)) {
10             return ApiReturn::error('操作太频繁');
11         }
12         if (empty($authorId) || empty($relevancId)) {
13             return ApiReturn::rdata(ApiCode::PARAM_ERROR);
14         }
15         if (!in_array($type, [1, 2, 3, 4, 5, 6, 7, 8, 9])) {
16             return ApiReturn::rdata(ApiCode::PARAM_ERROR);

```

```

17     }
18     $re = User::addUserNoLike($userId, $authorId, $relevancId, $type);
19     return ApiReturn::success(['status' => $re], $re == 1 ? '差评成功' : '取消差评');
20 } catch (\Exception $e) {
21     return ApiReturn::error();
22 }
23 }
24
25 /**
26  * 收藏列表
27  * @param Request $request
28  * @return Object
29  * @throws ContainerExceptionInterface
30  * @throws NotFoundExceptionInterface
31  */
32 public function getUserCollectList(Request $request): object
33 {
34     $limit = $request->get('limit', 10);
35     $uid = $request->get('user_id');
36     $platform = $request->get('platforma', 1);
37     try {
38         $type = intval($request->get('type', 0));
39         $type = $type ? [$type] : array_values([1, 2, 3, 4, 5]);
40         $list = UserCollect::query()
41             ->whereIn('article_type', $type)
42             ->where('user_id', $uid)
43             ->where('status', 1)
44             ->orderBy('update_time', 'desc')
45             ->whereHas('article', function ($query) {
46                 $query->where('status', 1);
47                 $query->where('check_status', 2);
48             })
49             ->with('article', function ($query) {
50                 $query->with([
51
52                     'user' => function ($query) {
53                         $query->select('id', 'username', 'avatar');
54                     },
55                     'game' => function ($query) {
56                         $query->select('id', 'game_name', 'game_icon');
57                     },
58                     'circle' => function ($query) {
59                         $query->select('id', 'circle_name', 'avatar');
60                     },
61                     'video' => function ($query) {
62                         $query->select('id', 'url', 'article_id', 'duration');
63                     },
64                     'images'
65                 ]);
66                 $query->select('id', 'title', 'user_id', 'cover', 'type', 'game_id', 'circle_id', 'text_content', 'label_list', 'create_time',
67 'rss_guid', 'rss_task_id', 'link_type', 'link_url');
68             })
69             ->paginate($limit)
70             ->toArray();

```

```

19     $list = empty($list) ? [] : $list;
20     CommonFnc::paginateReturnUnset($list);
21     $data = self::handleResorce($list, $uid, $platform);
22     $res = [
23         'total' => $list['total'] ?? 0,
24         'current_page' => $list['current_page'] ?? 1,
25         'last_page' => $list['last_page'] ?? 1,
26         'data' => $data,
27         'platform' => $platform
28     ];
29     return ApiReturn::success($res);
30 } catch (\Exception $e) {
31     return ApiReturn::error($e->getMessage());
32 }
33 }
34
35 /**
36  * 修改用户推送设置
37  * @catalog
38  * @title 修改用户推送设置
39  * @description 描述信息
40  * @method post
41  * @url /api/user/editPushConfigs
42  * @param field_name 必选 string 字段类型：comment_status 评论,like_status 点赞/收藏,attention_status 首次关
,remind_status 被@,system_status 系统
43  * @param field_value 必选 boolean true 开, false 关
44  * @return {"code":200,"message":"ok"}
45  */
46 public function editPushConfigs(Request $request)
47 {
48     $fieldName = (string)$request->post('field_name', '');
49     $fieldValue = (bool)$request->post('field_value', true);
50     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']);

```

```

1     $userId = $request->post('user_id');
2     try {
3         UserPushConfig::query()->where(['user_id' => $userId])->update([$fieldName => $fieldValue]);
4         return ApiReturn::success();
5     } catch (\Throwable $throwable) {
6         Common::log('修改用户推送配置信息', $throwable);
7         return ApiReturn::error();
8     }
9 }
10
11 /**
12  * 添加阿里云push绑定记录
13  * @param Request $request
14  * @return Object
15  */
16 public function addUserDevice(Request $request): object
17 {
18     $deviceNum = $request->post('device_num', '');
19     $platform = $request->post('platform', 1);

```

```

20     $uni_app = $request->post('uni_app', 0);
21     $userId = $request->post('user_id', 0);
22     Log::info('添加用户设备信息入参', [
23         'device_num' => $deviceNum,
24         'platform' => $platform,
25         'uni_app' => $uni_app,
26         'user_id' => $userId
27     ]);
28     try {
29         if (!empty($deviceNum) && in_array($platform, [1, 2]) && $uni_app == 0) {
30             $deviceIds = $platform == 1 ? AliPush::getAndroidDeviceIds($deviceNum) : AliPush::getIosDeviceIds($deviceNum);
31             if (empty($deviceIds)) {
32                 return ApiReturn::success('阿里推送设备ID为空');
33             }
34             $data = [
35                 'ali_device_num' => $deviceIds['DeviceId'][0],
36                 'device_type' => $uni_app != 0 ? 3 : $platform
37             ];
38             $info = UserAlipush::query()->where($data)->first();
39             if (empty($info)) {
40                 if ($userId > 0) {
41                     UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
42                 }
43                 $data['user_id'] = $userId;
44                 $data['device_num'] = $deviceNum;
45                 $data['check_date'] = date('Y-m-d H:i:s');
46                 UserAlipush::query()->create($data);
47             } else {
48                 $info->device_num = $deviceNum;
49                 if ($userId > 0 && $userId != $info->user_id) {
50                     UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
51
52                     $info->user_id = $userId;
53                 }
54                 $info->check_date = date('Y-m-d H:i:s');
55                 $info->status = 1;
56                 $info->save();
57             }
58         }
59         if (!empty($deviceNum) && $uni_app != 0) {
60             $data = [
61                 'ali_device_num' => $deviceNum,
62                 'device_type' => 3
63             ];
64             $info = UserAlipush::query()->where($data)->first();
65             if (empty($info)) {
66                 if ($userId > 0) {
67                     UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
68                 }
69                 $data['user_id'] = $userId;
70                 $data['device_num'] = $deviceNum;
71                 $data['check_date'] = date('Y-m-d H:i:s');
72                 UserAlipush::query()->create($data);
73             } else {

```

```
23         $info->device_num = $deviceNum;
24         if ($userId > 0 && $userId != $info->user_id) {
25             UserAlipush::query()->where(['user_id' => $userId])->update(['user_id' => 0]);
26             $info->user_id = $userId;
27         }
28         $info->check_date = date('Y-m-d H:i:s');
29         $info->status = 1;
30         $info->save();
31     }
32 }
33 return ApiReturn::success();
34 } catch (\Exception $e) {
35     Common::log('添加阿里云push绑定记录', $e);
36     ErrorLog::dispatch(Route::current()->getActionName(), CommonFnc::getRealIP(), json_encode($e), ['file' =>
37     $e->getFile(), 'line' => $e->getLine(), 'code' => $e->getCode(), 'message' => $e->getMessage()],
38     time()->onQueue('error_log');
39     return ApiReturn::error();
40 }
41 //用户发布文章
42 public function userArticles(Request $request)
43 {
44     $userId = $request->get('user_id', 0);
45     $limit = $request->get('limit', 10);
46     $is_review = $request->get('is_review');
47     $author_id = $request->get('author_id');
48     $device_id = $request->get('device_num');
49     return User::userArticles($userId, $limit, $is_review, $author_id, $device_id);
50 }
```

```
1
2 //用户收藏文章
3 public function userCollectArticles(Request $request)
4 {
5     $userId = $request->get('user_id', 0);
6     $limit = $request->get('limit', 10);
7     $authorID = $request->get('author_id');
8     $device_id = $request->get('device_num');
9     return User::userCollectArticles($authorID, $userId, $limit, $device_id);
10 }
11
12 //用户点赞文章
13 public function userLikeArticles(Request $request)
14 {
15     $userId = $request->get('user_id', 0);
16     $limit = $request->get('limit', 10);
17     $authorID = $request->get('author_id');
18     $device_id = $request->get('device_num');
19     return User::userLikeArticles($authorID, $userId, $limit, $device_id);
20 }
21
22 //勋章列表
23 public function medals(Request $request)
```

```
24 {
25     $userId = $request->get('user_id', 0);
26     return User::medals($userId);
27 }
28
29 //佩戴和取消佩勋章
30 public function wearMedal(Request $request)
31 {
32     $userId = $request->post('user_id');
33     if (empty($userId)) {
34         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
35     }
36     $medalId = $request->post('medal_id');
37     if (empty($medalId)) {
38         return ApiReturn::error('请选择要佩戴的勋章');
39     }
40     $type = $request->post('type', 1);
41     return User::wearMedal($userId, $medalId, $type);
42 }
43
44 //用户关注圈子列表
45 public function circleAttentionList(Request $request)
46 {
47     $authorId = $request->get('author_id', 0);
48     $limit = $request->get('limit', 10);
49     $user_id = $request->get('user_id', 0);
50     $type = $request->get('type', 1);
51
52     $name = $request->get('name');
53     return User::userAttentionCircle($authorId, $user_id, $limit, $type, $name);
54 }
55
56 //用户关注用户列表
57 public function AttentionUserList(Request $request)
58 {
59     try {
60         $user_id = $request->get('user_id');
61         $uid = $request->get('uid', 0); // 要查看的人
62         if ($uid == 0) $uid = $user_id;
63         $limit = $request->get('limit', 10);
64         $name = $request->get('name');
65         return User::AttentionUserList($uid, $user_id, $limit, $name);
66     } catch (\Exception $e) {
67         return ApiReturn::error($e->getMessage());
68     }
69 }
70
71 //用户是否在国外
72 public function isForeign(Request $request)
73 {
74     return User::isForeign();
75 }
76
77 //生成小程序generatescheme链接
```

```
27 public function generateScheme(Request $request)
28 {
29
30     $path = $request->post('path');
31     $query = $request->post('query');
32     $env_version = $request->post('env_version');
33     return User::generateScheme($path, $query, $env_version);
34 }
35
36 //通过手机号校验用户为新用户或者是否已绑定微信
37 public function checkMobile(Request $request)
38 {
39     $mobile = $request->post('mobile');
40     //校验手机号是否正确
41     if (empty($mobile) || !preg_match('/^1[3456789][0-9]{9}$/', $mobile)) {
42         return ApiReturn::error('请输入正确的手机号');
43     }
44     return User::checkMobile($mobile);
45 }
46
47 //获取微信手机号
48 public function getWxMobile(Request $request)
49 {
50     $user_id = $request->post('user_id');
51
52     if (empty($user_id)) {
53         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
54     }
55     $type = $request->post('type', 1);
56     $code = $request->post('code');
57     if ($type == 1) {
58         if (empty($code)) {
59             return ApiReturn::error('非法code');
60         }
61     }
62     $app_code = $request->post('app_code');
63     return User::getWxMobile($user_id, $code, $type, $app_code);
64 }
65
66 //通过code获取token
67 public function loginByCode(Request $request)
68 {
69     Log::info('loginByCode',$request->all());
70     $code = $request->post('code');
71     if (empty($code)) {
72         return ApiReturn::error('非法code');
73     }
74     return User::loginByCode($code);
75 }
76
77 //校验小程序的用户和app用户ID是否一致
78 public function checkAppUser(Request $request)
79 {
80     $user_id = $request->get('user_id');
```

```
30     $app_user_id = $request->get('app_user_id');
31     if (empty($user_id) || empty($app_user_id)) {
32         return ApiReturn::error('非法参数');
33     }
34     return User::checkAppUser($user_id, $app_user_id);
35 }
36 //绑定社媒账号
37 public function bindSocialAccount(Request $request)
38 {
39     $userId = $request->post('user_id');
40     if (empty($userId)) {
41         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
42     }
43     $type = $request->post('type');
44     if($type == 'steam'){
45         $code = $request->post('code');
46         if(empty($code)){
47             return ApiReturn::error('请输入好友码');
48         }
49         return User::bindPlatformFriendCode($userId, $code, $type);
50     }

```

```
1     $accounts = $request->post('accounts');
2     if(empty($accounts)){
3         return ApiReturn::error('请选择要绑定的账号');
4     }
5     if(!is_array($accounts)){
6         return ApiReturn::error('请选择要绑定的账号');
7     }
8     $domainRules = [
9         'dy' => ['douyin.com'],
10        'xhs' => ['xiaohongshu.com'],
11        'wb' => ['weibo.com'],
12        'bili' => ['bilibili.com', 'b23.tv'],
13    ];
14    $cnMap = [
15        'dy' => '抖音',
16        'xhs' => '小红书',
17        'wb' => '微博',
18        'bili' => '哔哩哔哩',
19    ];
20
21    foreach ($accounts as $k => $value) {
22        if (!isset($domainRules[$value['type']])) {
23            return ApiReturn::error("不支持的平台类型 ". $value['type']);
24        }
25        if(!empty($value['link'])){
26            if (!filter_var($value['link'], FILTER_VALIDATE_URL)) {
27                preg_match('/https?:\/\/[^\s]+\/u/', $value['link'], $matches);
28                if (isset($matches[0])) {
29                    $value['link'] = $matches[0]; // 提取第一个 URL 替换原字段
30                    $accounts[$k]['link'] = $matches[0];
31                }
32            }

```



```
33
34     if (!filter_var($value['link'], FILTER_VALIDATE_URL)) {
35         return ApiReturn::error("{ $cnMap[$value['type']] 链接不是有效的网址格式");
36     }
37     $host = parse_url($value['link'], PHP_URL_HOST);
38     $valid = false;
39     foreach ($domainRules[$value['type']] as $domain) {
40         if (stripos($host, $domain) !== false) {
41             $valid = true;
42             break;
43         }
44     }
45     if (!$valid) {
46         return ApiReturn::error("{ $cnMap[$value['type']] 链接不合法");
47     }
48 }
49 }
50 return User::bindSocialAccount($userId, $accounts);
```

```
1 }
2 //绑定平台好友码
3 public function bindPlatformFriendCode(Request $request)
4 {
5     $userId = $request->post('user_id');
6     if (empty($userId)) {
7         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
8     }
9     $code = $request->post('code');
10    if(empty($code)){
11        return ApiReturn::error('请输入好友码');
12    }
13    $platform = $request->post('type','steam');
14    return User::bindPlatformFriendCode($userId, $code,$platform);
15 }
16 //获取平台用户信息
17 public function getPlatformUserInfo(Request $request)
18 {
19     $userId = $request->get('user_id');
20     if (empty($userId)) {
21         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
22     }
23     $platform = $request->get('type','steam');
24     return User::getPlatformUserInfo($userId,$platform);
25 }
26 //设置平台隐私设置
27 public function setPlatformUserPrivacy(Request $request)
28 {
29     $userId = $request->post('user_id');
30     if (empty($userId)) {
31         return ApiReturn::rdata(ApiCode::LOGIN_ERROR);
32     }
33     $platform = $request->post('type','steam');
34     $privacy_profile = $request->post('privacy_profile');
35     if(empty($privacy_profile)){
```

```

36         return ApiReturn::error('请选择隐私设置');
37     }
38     if(!in_array($privacy_profile,[1,2,3])){
39         return ApiReturn::error('请选择正确的隐私设置');
40     }
41     return User::setPlatformUserPrivacy($userId,$platform,$privacy_profile);
42 }
43
44
45
46 }
47

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1\channel;
4
5  use App\Http\Controllers\Controller;
6  use App\Models\v1\channel\CreatorSource;
7  use App\Models\v1\channel\CreatorTask;
8  use App\Models\v1\channel\CreatorTaskImport;
9  use App\Models\v1\channel\UserCreatorResourceConfigAudit;
10 use App\Service\v1\ApiCode;
11 use App\Service\v1\ApiReturn;
12 use App\Service\v1\Common;
13 use App\Service\v1\Imports\CreatorSourceImport;
14 use Illuminate\Http\Request;
15 use App\Service\v1\channel\Channel as ChannelService;
16 use Maatwebsite\Excel\Facades\Excel;
17
18 class ChannelController extends Controller
19 {
20     /**
21      * 创作营列表
22      * @param Request $request
23      * @return Object
24      */
25     public function tasks(Request $request)
26     {
27         $userId = $request->get('user_id');
28         // $page = (int)$request->get('page',1);
29         $limit = (int)$request->get('limit',10);
30         // $page = ($page - 1) * $limit;
31         $keyword = (string)$request->get('keyword','');
32         $date = (string)$request->get('date','');
33         try {
34             // $list = ChannelService::tasks($userId,$page,$limit,$keyword,$date);
35             $list = ChannelService::paginateTasks($userId,$limit,$keyword,$date);
36             return ApiReturn::success($list);
37         } catch (\Throwable $throwable) {
38             Common::log('创作营列表',$throwable);
39             return ApiReturn::error();
40         }
41     }

```

```
42
43  /**
44   * 创作营提费用列表
45   * @param Request $request
46   * @return Object
47   */
48 public function supplierTasks(Request $request)
49 {
50     $userId = $request->get('user_id');
51
52     $page = (int)$request->get('page',1);
53     $type = (int)$request->get('type',1);
54     $limit = (int)$request->get('limit',10);
55     $page = ($page - 1) * $limit;
56     try {
57         $list = ChannelService::supplierTasks($userId,$type,$page,$limit);
58         return ApiReturn::success($list);
59     } catch (\Throwable $throwable) {
60         Common::log('创作营提费用列表',$throwable);
61         return ApiReturn::error();
62     }
63 }
64
65 /**
66  * 任务内容列表 ( 渠道进度 )
67  * @param Request $request
68  * @return Object
69  */
70 public function taskContents(Request $request)
71 {
72     $taskId = (int)$request->get('task_id',0);
73     if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
74     $userId = $request->get('user_id');
75     $page = (int)$request->get('page',1);
76     $limit = (int)$request->get('limit',10);
77     $page = ($page - 1) * $limit;
78     try {
79         $list = ChannelService::taskContents($userId,$taskId,$page,$limit);
80         return ApiReturn::success($list);
81     } catch (\Throwable $throwable) {
82         Common::log('任务内容列表 ( 渠道进度 )',$throwable);
83         return ApiReturn::error();
84     }
85 }
86
87 /**
88  * 渠道导入内容
89  * @param Request $request
90  * @return Object
91  */
92 public function creatorSourceImport(Request $request)
93 {
94     $taskId = (int)$request->input('task_id',0);
```

```

45     $fileInfo = $request->file('file_path');
46     if($taskId <= 0 || empty($fileInfo)) return
    ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
47     try {
48         $importStatus = CreatorTask::query()->where(['id' => $taskId])->value('import_status');
49         if(empty($importStatus)) return ApiReturn::error(ApiCode::ROLE_ERROR['msg'],ApiCode::ROLE_ERROR['code']);
50         CreatorSource::query()->where(['creator_task_id' => $taskId])->delete();

1      CreatorTaskImport::query()->updateOrCreate([
2          'task_id' => $taskId
3      ],[
4          'name' => $fileInfo->getClientOriginalName(),
5          'file_size' => ceil($fileInfo->getSize()/1024)
6      ]);
7      Excel::import(new CreatorSourceImport($taskId),$fileInfo);
8      return ApiReturn::success();
9  }catch (\Throwable $throwable) {
10     return ApiReturn::error($throwable->getMessage());
11 }
12 }
13
14 /**
15  * 渠道修改内容链接
16  * @param Request $request
17  * @return Object
18  */
19 public function updateCreatorSource(Request $request)
20 {
21     $getContentId = (int)$request->get('get_content_id',0);
22     $url = (string)$request->get('url','');
23     if($getContentId <= 0 || !preg_match("/http[s]?://(?:[a-z]+\.)+[a-z0-9]*\.[a-z]{2,4}(?:[/?#]|$)/", $url,$match)) return
    ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
24     $userId = $request->get('user_id');
25     try {
26         $creatorSourceInfo = CreatorSource::where('id',$getContentId)->whereHas('creator_task', function ($q) use ($userId) {
27             $q->where('user_id',$userId);
28         }->first();
29         if(empty($creatorSourceInfo)) return
    ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
30         if ($url == $creatorSourceInfo->url) return ApiReturn::success();
31         $urlPath = parse_url($match[0]);
32         $creatorSourceInfo->url = $urlPath['scheme'].'://'.$urlPath['host'].$urlPath['path'];
33         // 初始化参数
34         $creatorSourceInfo->remark = '';
35         $creatorSourceInfo->url_game_name = '';
36         $creatorSourceInfo->game_time_str = '';
37         $creatorSourceInfo->game_time = 0;
38         $creatorSourceInfo->url_status = 0;
39         $creatorSourceInfo->practical_released_at = null;
40         $creatorSourceInfo->save();
41         return ApiReturn::success();
42     } catch (\Throwable $throwable) {
43         Common::log('渠道修改内容链接',$throwable);
44         return ApiReturn::error();

```

```
45     }
46 }
47
48 /**
49  * 渠道资源配置列表
50  * @param Request $request
51
52  * @return Object
53  */
54 public static function getResourceConfig(Request $request)
55 {
56     $userId = $request->get('user_id');
57     $limit = (int)$request->get('limit',10);
58     $lists = ChannelService::resourceConfigList($userId,$limit);
59     return ApiReturn::success($lists);
60 }
61
62 /**
63  * 渠道资源配置 平台列表
64  * @param Request $request
65  * @return Object
66  */
67 public function getResourceConfigPlatforms(Request $request)
68 {
69     $userId = $request->get('user_id');
70     $lists = ChannelService::resourceConfigPlatforms($userId);
71     return ApiReturn::success($lists);
72 }
73
74 /**
75  * 渠道资源配置 资源列表
76  * @param Request $request
77  * @return Object
78  */
79 public function getResourceConfigResources(Request $request)
80 {
81     $userId = $request->get('user_id');
82     $lists = ChannelService::resourceConfigResources($userId);
83     return ApiReturn::success($lists);
84 }
85
86 /**
87  * 渠道资源配置 历史提交记录
88  * @param Request $request
89  * @param int $config_id 资源配置ID
90  * @return Object
91  */
92 public function getResourceConfigHistory(Request $request)
93 {
94     $userId = $request->get('user_id');
95     $configId = $request->get('config_id', 0);
96     if ($configId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
97     $lists = ChannelService::resourceConfigHistory($userId, $configId);
98     return ApiReturn::success($lists);
99 }
```

```

48     }
49
50     /**
1   * 渠道资源配置 新增审核
2   * @param Request $request
3   * @param int $creator_resource_config_id 资源 id
4   * @param int $platform_id 平台ID
5   * @return Object
6   */
7   public function addResourceConfig (Request $request)
8   {
9       $userId = $request->get('user_id');
10      $configId = $request->get('creator_resource_config_id', 0);
11      $platformId = $request->get('platform_id', 0);
12      $unitPrice = $request->get('unit_price', 0);
13      if (!preg_match('/^\d{1,4}(\.\d{1,4})?$/',$unitPrice)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
14      ApiCode::PARAM_ERROR['code']);
15      if ($configId <= 0 || $platformId <= 0 || $unitPrice <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
16      ApiCode::PARAM_ERROR['code']);
17      return ChannelService::addResourceConfig($userId, $configId, $platformId, $unitPrice);
18  }
19
20  /**
21  * 渠道资源配置 修改配置审核
22  * @param Request $request
23  * @param int $config_id 资源配置ID
24  * @param int $platform_id 平台ID
25  * @param int $unit_price 单价
26  * @return Object
27  */
28  public function updateResourceConfig (Request $request)
29  {
30      $userId = $request->get('user_id');
31      $configId = $request->get('config_id', 0);
32      $unitPrice = $request->get('unit_price', 0);
33      if (!preg_match('/^\d{1,4}(\.\d{1,4})?$/',$unitPrice)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
34      ApiCode::PARAM_ERROR['code']);
35      if ($configId <= 0 || $unitPrice <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],
36      ApiCode::PARAM_ERROR['code']);
37      return ChannelService::updateResourceConfig($userId, $configId, $unitPrice);
38  }
39  }
40  }

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v1\channel;
4
5  use App\Http\Controllers\Controller;
6  use App\Http\Requests\ChannelBinding;
7  use App\Jobs\CheckCreatorSourceUrl;
8  use App\Models\v1\ActivityCenterPlatform;
9  use App\Models\v1\channel\CreatorSource;
10 use App\Models\v1\channel\CreatorTask;

```

```

11 use App\Models\v1\channel\UserDefaultPlatform;
12 use App\Models\v1\ServiceConfig;
13 use App\Models\v1\User;
14 use App\Service\v1\ApiCode;
15 use App\Service\v1\ApiReturn;
16 use App\Service\v1\channel\Channel as ChannelService;
17 use App\Service\v1\Code;
18 use App\Service\v1\Common;
19 use App\Service\v1\RedisKey;
20 use Illuminate\Http\Request;
21 use Illuminate\Support\Facades\Redis;
22
23 class FramerController extends Controller
24 {
25
26     /**
27      * 创作者是否绑定平台账号
28      * @param ChannelBinding $request
29      * @return Object
30      * @throws \Illuminate\Auth\Access\AuthorizationException
31      * @throws \Illuminate\Validation\ValidationException
32      */
33     public function channelBinding(ChannelBinding $request)
34     {
35         $request->validate('task_check');
36         $taskId = (int)$request->get('task_id',0);
37         if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
38         $userId = $request->get('user_id');
39         try {
40             $creatorTaskInfo = CreatorTask::query()->where(['id' => $taskId])->select('channel','end_status')->first();
41             if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
42             if($creatorTaskInfo->end_status == 1) return
ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
43             if($creatorTaskInfo->task_type != 0) return
ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);
44             $info = ChannelService::channelBinding($userId,$taskId,$creatorTaskInfo->channel);
45             return ApiReturn::success($info);
46         } catch (\Throwable $throwable) {
47             Common::log('创作者是否绑定平台账号',$throwable);
48             return ApiReturn::error();
49         }
50     }

```

```

1
2     /**
3      * 创作者领取词条
4      * @param ChannelBinding $request
5      * @return Object
6      * @throws \Illuminate\Auth\Access\AuthorizationException
7      * @throws \Illuminate\Validation\ValidationException
8      */
9     public function getContent(ChannelBinding $request)
10     {
11         $request->validate('task_check');

```

```

12     $taskId = (int)$request->get('task_id',0);
13     if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
14     $userId = $request->get('user_id');
15     try {
16         $creatorTaskInfo = CreatorTask::query()->where(['id' =>
17 $taskId])->select('channel','end_status','get_time_limit','binding_platform_switch')->first();
18         if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
19         if($creatorTaskInfo->end_status == 1) return
20 ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
21         if($creatorTaskInfo->task_type != 0) return
22 ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);
23         $count = CreatorSource::query()->where(['creator_task_id' => $taskId,'channel' =>
24 $creatorTaskInfo->channel,'user_id' => $userId])->count();
25         if($count > 0) return
26 ApiReturn::error(ApiCode::SYSTEM_TASK_REPEAT_GET['msg'],ApiCode::SYSTEM_TASK_REPEAT_GET['code']);
27 //         $getContentTime = ServiceConfig::query()->where(['field_name' => 'get_content_time']->value('value');
28         $redisTime = (int)Redis::get(RedisKey::GET_CONTENT_TIME.$taskId);
29         $nowTime = time();
30         $time = $redisTime + $creatorTaskInfo->get_time_limit - $nowTime;
31         if($time > 0) return ApiReturn::success(['status' => false,'time' => $redisTime +
32 $creatorTaskInfo->get_time_limit,'msg' => ApiCode::GET_CONTENT_TIME['msg'],'creator_source' => new \stdClass()]);
33         $info = CreatorSource::query()->where(['creator_task_id' => $taskId,'channel' => $creatorTaskInfo->channel,'user_id'
34 => 0])
35         ->where('released_at','<=',date('Y-m-d'))
36         ->select('id','content','score')->first();
37         if(empty($info)) return ApiReturn::error(ApiCode::GET_CONTENT['msg'],ApiCode::GET_CONTENT['code']);
38         $info->user_id = $userId;
39         $info->get_time = date('Y-m-d H:i:s',$nowTime);
40         $info->save();
41         if($creatorTaskInfo->binding_platform_switch == 1) {
42             $checkContentTime = ServiceConfig::query()->where(['field_name' => 'check_content_time']->value('value');
43             dispatch(new
44 CheckCreatorSourceUrl($info->id))->onQueue('check_creator_source_url')->delay(now()->addSeconds($checkContentTime));
45         }
46         Redis::set(RedisKey::GET_CONTENT_TIME.$taskId,$nowTime,'EX',3600);
47         return ApiReturn::success(['status' => true,'time' => 0,'msg' => '','creator_source' => $info]);
48     } catch (\Throwable $throwable) {
49         Common::log('创作者领取词条',$throwable);
50         return ApiReturn::error();
51     }
52 }
53
54 /**
55  * 创作者提交平台连接
56  * @param Request $request
57  * @return Object
58  */

```

```

1     public function addContentUrl(Request $request)
2     {
3         $getContentId = (int)$request->post('get_content_id',0);
4         $url = (string)$request->post('url','');
5         if($getContentId <= 0 || !preg_match("/http[s]?://[w.]+[w/v]*[w.]*?[/w=&\\+\\%]*/is",$url,$match)) return
6 ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);

```



```

6      $userId = $request->post('user_id');
7      try {
8          $creatorSourceInfo = CreatorSource::where(['id' => $getContentId,'user_id' =>
$userId])->with('creator_task')->select('id','url','creator_task_id')->first();
9          if(empty($creatorSourceInfo)) return
ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
10         // 需要收集截图
11         if ($creatorSourceInfo->creator_task->need_pic) {
12             $pic = (string)$request->post('pic','');
13             if(empty($pic)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
14             $creatorSourceInfo->pic = $pic;
15         }
16         $urlPath = parse_url($match[0]);
17         $creatorSourceInfo->url = $urlPath['scheme'].'://'.$urlPath['host'].$urlPath['path'];
18         $creatorSourceInfo->save();
19         return ApiReturn::success();
20     } catch (\Throwable $throwable) {
21         Common::log('创作者提交平台连接',$throwable);
22         return ApiReturn::error();
23     }
24 }
25
26 /**
27  * 创作者提交图片
28  * @param Request $request
29  * @return Object
30  */
31 public function addOrUpdateContentPic(Request $request)
32 {
33     $pic = (string)$request->post('pic','');
34     if(empty($pic)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
35     $userId = $request->post('user_id');
36     $taskId = (int)$request->get('task_id',0);
37     if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
38     try {
39         $creatorTaskInfo = CreatorTask::where(['id' =>
$taskId])->with('game')->select('channel','end_status','task_type','unit_price')->first();
40         if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
41         if($creatorTaskInfo->end_status == 1) return
ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
42         if($creatorTaskInfo->task_type != 1) return
ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);
43
44         $creatorSourceInfo = CreatorSource::where(['creator_task_id' => $taskId,'user_id' =>
$userId])->with('creator_task')->select('id','url','creator_task_id')->first();
45         if(empty($creatorSourceInfo)) {
46             $creatorSourceInfo = new CreatorSource();
47             // $creatorSourceInfo->url_status = 1;
48             $creatorSourceInfo->price = $creatorTaskInfo->unit_price;
49             $creatorSourceInfo->creator_task_id = $taskId;
50             $creatorSourceInfo->user_id = $userId;

```

```

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

```

```
3     }
4     $creatorSourceInfo->pic = $pic;
5     $creatorSourceInfo->save();
6     return ApiReturn::success();
7 } catch (\Throwable $throwable) {
8     Common::log('创作者提交图片',$throwable);
9     return ApiReturn::error();
10 }
11 }
12
13
14 /**
15  * 创作者提交文本任务
16  * @param Request $request
17  * @return Object
18  */
19 public function addOrUpdateCreatorFileTaskSource(Request $request)
20 {
21     $pic = (string)$request->post('pic','');
22     $content = (string)$request->post('content','');
23     $validCount = (int)$request->post('valid_count', 1);
24     $file = (string)$request->post('file','');
25     // 三个参数不能同时为空
26     if(empty($pic) && empty($content) && empty($file)) return
    ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
27     if ($validCount <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'], ApiCode::PARAM_ERROR['code']);
28     $userId = $request->post('user_id');
29     $taskId = (int)$request->get('task_id',0);
30     if($taskId <= 0) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
31     try {
32         $creatorTaskInfo = CreatorTask::where(['id' =>
    $taskId])->with('game')->select('channel','end_status','task_type','unit_price')->first();
33         if(empty($creatorTaskInfo)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
34         if($creatorTaskInfo->end_status == 1) return
    ApiReturn::error(ApiCode::TASK_END_ERROR['msg'],ApiCode::TASK_END_ERROR['code']);
35         if($creatorTaskInfo->task_type != 2) return
    ApiReturn::error(ApiCode::TASK_TYPE_ERROR['msg'],ApiCode::TASK_TYPE_ERROR['code']);
36
37         $creatorSourceInfo = CreatorSource::where(['creator_task_id' => $taskId,'user_id' =>
    $userId])->with('creator_task')->select('id','url','creator_task_id')->first();
38         if(empty($creatorSourceInfo)) {
39             $creatorSourceInfo = new CreatorSource();
40             // $creatorSourceInfo->url_status = 1;
41             $creatorSourceInfo->price = $creatorTaskInfo->unit_price;
42             $creatorSourceInfo->creator_task_id = $taskId;
43             $creatorSourceInfo->user_id = $userId;
44             $creatorSourceInfo->channel = $creatorTaskInfo->channel;
45             $creatorSourceInfo->game_name = $creatorTaskInfo->game ? $creatorTaskInfo->game->game_name : '';
46         }
47         $creatorSourceInfo->pic = $pic;
48         $creatorSourceInfo->content = $content;
49         $creatorSourceInfo->valid_count = $validCount;
50         if ($content) {
```

```

1      $creatorSourceInfo->content_md5 = md5($content);
2      }
3      $creatorSourceInfo->file = $file;
4      $creatorSourceInfo->save();
5      return ApiReturn::success();
6      } catch (\Throwable $throwable) {
7          Common::log('创作者提交图片',$throwable);
8          return ApiReturn::error();
9      }
10     }
11
12     /**
13      * 创作者绑定平台账户
14      * @param ChannelBinding $request
15      * @return Object
16      * @throws \Illuminate\Auth\Access\AuthorizationException
17      * @throws \Illuminate\Validation\ValidationException
18      */
19     public function addChannelBinding(ChannelBinding $request)
20     {
21         $request->validate('binding');
22         $params = $request->post();
23         try {
24             $count = UserDefaultPlatform::query()->where(['user_id' => $params['user_id'],'platform_type' =>
25 $params['platform_type']])->count();
26             if($count == 0) {
27                 UserDefaultPlatform::query()->create($params);
28             }
29             return ApiReturn::success();
30         } catch (\Throwable $throwable) {
31             Common::log('创作者绑定平台账户',$throwable);
32             return ApiReturn::error();
33         }
34     }
35
36     /**
37      * 绑定平台列表
38      * @param Request $request
39      * @return Object
40      */
41     public function channels(Request $request)
42     {
43         $userId = $request->get('user_id');
44         try {
45             $platformIds = UserDefaultPlatform::query()->where(['user_id' => $userId])->pluck('platform_type')->toArray();
46             $platformTypes = ActivityCenterPlatform::query()->select('id','name','icon')->get();
47             foreach ($platformTypes as &$platformType) {
48                 $platformType->is_binding = in_array($platformType->id,$platformIds) ? true : false;
49             }
50             return ApiReturn::success(['list' => $platformTypes]);
51         } catch (\Throwable $throwable) {
52             Common::log('绑定平台列表',$throwable);
53             return ApiReturn::error();
54         }
55     }

```

```

3      }
4  }
5
6  /**
7   * 领取词条列表/创作的内容
8   * @param Request $request
9   * @return Object
10  */
11  public function getContents(Request $request)
12  {
13      $userId = $request->get('user_id');
14      $page = (int)$request->get('page',1);
15      $limit = (int)$request->get('limit',10);
16      $page = ($page - 1) * $limit;
17      try {
18          $creatorSources = CreatorSource::where(['user_id' => $userId])
19              ->select('id','get_time','content','file','pic','created_at')
20              ->offset($page)->limit($limit)->orderBy('id','desc')->get();
21          foreach ($creatorSources as &$creatorSource) {
22              $creatorSource->content = mb_strlen($creatorSource->content) > 50 ?
mb_substr($creatorSource->content,0,50).'...' : $creatorSource->content;
23              if ($creatorSource->get_time == null) {
24                  $creatorSource->get_time = $creatorSource->created_at;
25                  $creatorSource->get_time = $creatorSource->get_time->format('Y-m-d H:i:s');
26              }
27              $creatorSource->file = Common::concatUrlStr($creatorSource->file);
28              $creatorSource->pic = Common::concatUrlStr($creatorSource->pic);
29              $creatorSource->created_at = null;
30          }
31          return ApiReturn::success(['creator_sources' => $creatorSources]);
32      } catch (\Throwable $throwable) {
33          Common::log('领取词条列表',$throwable);
34          return ApiReturn::error();
35      }
36  }
37
38  /**
39   * 创作者添加qq号
40   * @param Request $request
41   * @return Object
42   */
43  public function addQq(Request $request)
44  {
45      $qq = (string)$request->post('qq','');
46      if(empty($qq)) return ApiReturn::error(ApiCode::PARAM_ERROR['msg'],ApiCode::PARAM_ERROR['code']);
47      $userId = $request->post('user_id');
48      try {
49          User::query()->where(['id' => $userId])->update(['qq' => $qq]);
50          return ApiReturn::success();
51      } catch (\Throwable $throwable) {
52          Common::log('创作者添加qq号',$throwable);
53          return ApiReturn::error();
54      }
55  }

```

```

5     }
6
7 }
8

```

```

1  <?php
2
3  namespace App\Http\Controllers\api\v2;
4
5  use App\Http\Controllers\Controller;
6  use App\Http\Requests\GameRequest;
7  use App\Service\v1\ApiReturn;
8  use App\Service\v2\Article as V2Article;
9  use App\Service\v1\Code;
10 use App\Service\v1\Common;
11 use Illuminate\Http\Request;
12 use Illuminate\Support\Facades\Log;
13
14 class ArticleController extends Controller
15 {
16     //发布动态
17     public function publishDynamic(Request $request)
18     {
19         try {
20             $param = $request->post();
21             $param['action'] = $request->post('action', Code::ACTION_ADD);
22             $param['type'] = Code::ARTICLE_TYPE_MAP['dynamic'];
23             // 验证 设置
24             Log::info('发布动态-入口参数-----' . json_encode($param, 256));
25             $err = (array)V2Article::checkArticleParam($param);
26             Log::info('发布动态-校验文章信息-----' . json_encode($err, 256));
27             if ($err) return ApiReturn::rdata($err);
28             // 插入到 article 表的数据
29             $param['data'] = [
30                 'cover' => $param['cover'],
31                 'check_status' => Code::CHECK_STATUS_MAP['in_review'], // 审核状态为 1 未审核
32                 'content' => $param['content'],
33                 // article 标题 截取
34                 'title' => $param['title'] ?? '',
35                 'text_content' => $param['text_content'],
36                 'content_len' => $param['content_len'],
37                 'home_page' => $param['home_page'],
38                 'label_list' => $param['label_list'],
39                 'abstract' => $param['abstract'],
40                 'user_id' => $param['user_id'],
41             ];
42             if (!empty($data['title'])) {
43                 if (!V2Article::checkContentBySensitiveWord($data['title'])) {
44                     $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
45                 }
46             }
47             // 敏感词检测-内容
48             if (!empty($data['content'])) {
49                 if (!V2Article::checkContentBySensitiveWord($data['content'])) {

```

```

50      $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
1
2      }
3      }
4      if (!empty($data['text_content'])) {
5          if (!V2Article::checkContentBySensitiveWord($data['text_content'])) {
6              $param['data']['check_status'] = Code::CHECK_STATUS_MAP['in_review'];
7          }
8      }
9      if ($param['action'] == Code::ACTION_ADD) {
10         // 新增需要的数据 (修改的时候不可更改的数据)
11         $param['data'] = array_merge($param['data'], [
12             'game_id' => $param['game_id'],
13             'task_id' => $param['task_id'],
14             'circle_id' => $param['circle_id'],
15             'type' => $param['type'],
16             'is_writer_task' => $param['is_writer_task'],
17             'ip_location' => $param['ip_location'] ?? '',
18         ]);
19         if (isset($param['created_at'])) {
20             $param['data']['create_time'] = strtotime($param['created_at']);
21             $param['data']['update_time'] = strtotime($param['created_at']);
22         }
23     }
24     Log::error('发布动态-插入数据-----', $param);
25     $res = V2Article::addArticle($param);
26     return $res;
27 } catch (\Throwable $throwable) {
28     Log::error('发布动态-插入数据-----' . $throwable->getMessage());
29     return ApiReturn::error($throwable->getMessage());
30 }
31
32 }
33

```
