```php
<?php

namespace App\AppCore;

use App\Article as ArticleModel;
use App\Events\ArticleEvent;
use App\Exceptions\ApiException;
use App\Exceptions\ValidatorException;
use App\Facades\ElasticSearch;
use App\Helpers\AttachmentHelper;
use App\Helpers\BuilderQuery;
use App\Helpers\CacheHelper;
use App\Helpers\ClearContentBlocks;
use App\Helpers\ValidatorHelper;
use App\Recommendation as RecommendationModel;
use App\Share;
use App\Tag;
use App\User as UserModel;
use Carbon\Carbon;
use Exception;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Telegram;

class Article extends App
{
        public static $defaultCount = 20;
        public static $defaultOffset = 0;
        public static $defaultLang = 'ru,bash';
        public static $langAllowed = ['ru', 'bash'];
        public static $defaultLocalization = 'ru';
        public static $defaultTranslated = true;

        public static $defaultSelect = [
                'id' => true,
                'slug' => true,
                'title' => true,
                'cover' => true,
                'pubdate' => true,
                'author' => true,
                'text' => true,
                'categories' => true,
                'tags' => true,
                'status' => true,
                'params' => true,
                'lang' => true,
                'langs' => true,
                'geo_block' => true,
                'negative' => true,
                'theme_day' => true,
                'attachments' => true,
                'indexing' => true,
                'target' => true,
                'is_recommendation' => true,
                'do_scroll' => true,
                'number' => true,
                'longread' => true,
                'description' => true,
                'updated_at' => true,
        ];

        public static $fieldsAllowed = [
                'id',
                'slug',
                'title',
                'pubdate',
                'author',
                'text',
                'categories',
                'tags',
                'cover',
                'lang',
                'langs',
                'is_recommendation',
                'in_image',
                'in_video',
                'in_audio',
                'geo_block',
                'negative',
                'theme_day',
                'attachments',
                'indexing',
                'target',
                'description',
                'updated_at',
        ];

        public static $defaultSelectHidden = [
                'tags',
                'categories',
                'title',
                'text',
                'lang',
                'langs',
                'attachments',
                'target',
                'is_recommendation',
                'do_scroll',
                'number',
                'description',
        ];

        public static function index($params, $fields)
        {
                try {
                        $cache = new
CacheHelper(md5('feed.getFeed|' . json_encode($params->all())));
                        return $cache->return(5, function () use
($params, $fields) {
                                $fieldsAllowed = self::
$fieldsAllowed;
                                $AccessDenied = false;
                                $validRole = ['super-admin', 'chief-
editor', 'middle-editor', 'editor'];
                                $validAbilities = ['all-edit',
'news-edit'];
                                $IR = $IA = [];
                                if (Auth::guard('api')->check()) {
                                        $user = Auth::guard('api')-
>user();
                                        $rolesUser = $user-
>getRoles()->toArray();
                                        $abilitiesUser = $user
                                                ->getAbilities()
                                                ->map(function
($item) {
                                                        return
$item->name;
                                                })
                                                ->toArray();

                                        $IR =
array_intersect($validRole, $rolesUser);
                                        $IA =
array_intersect($validAbilities, $abilitiesUser);

                                        $IR = array_values($IR);
                                        $IA = array_values($IA);

                                        if (count($IR) > 0 &&
count($IA) > 0) {
                                                $AccessDenied =
true;
                                        }
                                }

                                if (Auth::guard('api')->check()) {
                                        $user = Auth::guard('api')-
>user();
                                        $rolesUser = $user-
>getRoles()->toArray();
                                        $abilitiesUser = $user
                                                ->getAbilities()
                                                ->map(function
($item) {
                                                        return
$item->name;
                                                })
                                                ->toArray();

                                        $IR =
array_intersect($validRole, $rolesUser);
```

```
                                        $IA =
array_intersect($validAbilities, $abilitiesUser);
                                        if (count($IR) > 0) {
                                                if (in_array('all-
edit', $IA) || in_array('news-edit', $IA)) {
                                                }
                                        }
                                }

                                if ($AccessDenied !== false) {
                                        array_push($fieldsAllowed,
'status');
                                }
                                if (is_null($params->lang)) {
                                        $params->merge(['lang' =>
self::$defaultLang]);
                                }
                                if (is_null($params->localization)
|| !in_array($params['localization'], self::$langAllowed)) {
                                        $params-
>merge(['localization' => self::$defaultLocalization]);
                                }
                                if (is_null($params->translated)) {
                                        $params->merge(['translated'
=> self::$defaultTranslated]);
                                }
                                $params->merge(['translated' =>
is_true($params->translated)]);

                                if (is_null($params->count)) {
                                        $params->merge(['count' =>
self::$defaultCount]);
                                }
                                if (is_null($params->offset)) {
                                        $params->merge(['offset' =>
self::$defaultOffset]);
                                }
                                if (!is_null($params->only_video)) {
                                        $params->merge(['only_video'
=> is_true($params->only_video)]);
                                } else {
                                        $params->merge(['only_video'
=> false]);
                                }

                                if (is_null($params->theme_day)) {
                                        $params->merge(['theme_day'
=> false]);
                                } else {
                                        $params->merge(['theme_day'
=> is_true($params->theme_day)]);
                                }
                                if (is_null($params->order)) {
                                        $params->merge(['order' =>
'desc']);
                                }

                                if (is_null($params->negative)) {
                                        $params->merge(['negative'
=> true]);
                                } else {
                                        $params->merge(['negative'
=> is_true($params->negative)]);
                                }

                                try {
                                        new ValidatorHelper($params-
>all(), [
                                                'rules' => [
                                                        'count' =>
['filled'],
                                                        'offset' =>
['filled'],

                                                        'negative'
=> ['filled', 'boolean'],
                                                        'only_video'
=> ['filled', 'boolean'],
                                                        'theme_day'
=> ['filled', 'boolean'],
                                                        'categories'
=> ['filled'],
                                                        'tag' =>
['filled', 'string'],
                                                        'order' =>
['filled', 'in:desc,asc'],
                                                        'search' =>
['filled', 'string'],
                                                        'target' =>
['filled', 'string'],
                                                        'fields' =>
['filled'],
                                                        'lang' =>
['required'],
                                                        'localizatio
n' => ['required'],
                                                        'translated'
=> ['required'],
                                                ],
                                                'alias' => [],
                                        ]);
                                } catch (ValidatorException $e) {
                                        throw new ApiException($e-
>getDecodedMessage(), 101);
                                }

                                $author = null;
                                if (!empty($params['author'])) {
                                        $author = 
UserModel::select(['id', 'avatar', 'nicName'])
                                                ->where('id', 
$params->author)
                                                ->whereHas('roles', 
function ($query) {
                                                        // $query-
>Where(function($query) {
                                                        // 
$query->orWhere('name', 'editor');
                                                        // 
$query->orWhere('name', 'middle-editor');
                                                        // 
$query->orWhere('name', 'super-admin');
                                                        // 
$query->orWhere('name', 'chief-editor');
                                                        // });
                                                })
                                                ->first();
                                        // if (!$author) {

                                        // }

unset($fieldsAllowed[array_search('author', $fieldsAllowed)]);
                                        // dd($fieldsAllowed);
                                }

                                $builderQuery = new BuilderQuery([
                                        'getCount' => false,
                                        'fields' => [
                                                'return' => !
empty($fields) ? explode(',', $fields) : [],
                                                'default' => self::
$defaultSelect,
                                                'allowed' => 
$fieldsAllowed,
                                                'critically' => 
['id'],
                                        ],
                                        'table' => 'articles',
                                        'selectHidden' => self::
$defaultSelectHidden,
                                ]);
                                $fields = $builderQuery-
>getFields();

                                $langs_ = explode(',', 
$params['lang']);
                                $langs = [];
                                $load = [];
                                foreach ($langs_ as $lang) {
                                        if (in_array($lang, self::
$langAllowed)) {
                                                $langs[] = $lang;
                                        }
                                }

                                $targets = collect(explode(',', 
$params->target))
                                        ->filter(function ($target) 
{
                                                return 
in_array($target, ['yandex-news', 'sendpulse', 'yandex-turbo', 
'amp']);
                                        })
                                        ->toArray();

                                $articles = 
ArticleModel::select($builderQuery->getSelect())
                                        -
>join('article_localizations', function ($join) use ($params, 
$langs, $AccessDenied) {
                                                $join
                                                        -
>on('articles.id', '=', 'article_localizations.article_id')
                                                        -
>where('article_localizations.lang', $params['localization'])
                                                        -
>where(function ($where) use ($langs) {

foreach ($langs as $lang) {

        $where->orWhere('lang', $lang);
                                                                }
                                                        })
                                                        -
>where(function ($where) use ($params, $AccessDenied) {
                                                                if 
($params->search && $AccessDenied) {

        $where->orWhere('article_localizations.title', 'like', '%' . 
$params->search . '%');
                                                                }
                                                        });
                                        })
                                        // 
->whereHas('localization', function ($query) use ($langs, $params, 
$AccessDenied) {
                                        //      // $query-
>where(function ($where) use ($langs) {
                                        //      //      foreach 
($langs as $lang) {
                                        //      //      
$where->orWhere('lang', $lang);
                                        //      //      }
                                        //      // });
                                        //      $query-
>select('lang');
```

```
                                        //      if
($params['translated'] == true || $AccessDenied === false) {
                                        //              $query-
>where('lang', $params['localization']);
                                        //      }
                                        //      // $query-
>orderBy('id');
                                        // })

                                        ->Where(function ($query)
use ($targets) {
                                                if ($targets) {
                                                        $query-
>whereHas('articleTarget', function ($query) use ($targets) {

$query->Where(function ($where) use ($targets) {

        foreach ($targets as $target) {

                $where->where('key', $target);

        }
                                                                });
                                                        });
                                                }
                                        })
                                        ->Where(function ($query)
use ($AccessDenied) {
                                                if ($AccessDenied

=== false) {
                                                        $query
                                                                -
>where('longread', false)
                                                                -
>where('pubdate', '<=', Carbon::now())
                                                                -
>where('status', 10)
                                                                -
>whereHas('articleCategories', function ($query) {

        $query->where('status', 10);
                                                                });
                                                }
                                        })
                                        ->Where(function ($query)
use ($AccessDenied, $params) {
                                                if (!
empty($params['categories']) && is_string($params['categories'])) {
                                                        $categories
= explode(',', $params['categories']);
                                                        $query-
>whereHas('articleCategories', function ($query) use ($categories,
$params) {
```

```
$query->Where(function ($where) use ($categories) {

        foreach ($categories as $value) {

                if (preg_match('/[0-9]+/m', $value) === 1) {

                        $where->orWhere('id', $value);

                }

                if (preg_match('/[a-z-]+/m', $value) === 1) {

                        $where->orWhere('slug', $value);

                }

        }
                                                                });

                                                                //
->whereHas('localization', function ($query) use ($params) {
                                                                //
        $query->where('lang', $params['localization']);
                                                                // }
);
                                                        });
                                                }
                                        })
                                        ->Where(function ($query)
use ($AccessDenied, $params) {
                                                if (!empty($params-
>tag) && is_string($params->tag)) {
                                                        $query-
>whereHas('articleTags', function ($query) use ($params) {

$query->where('title', '=', $params->tag);
                                                        });
                                                }
                                        })
                                        ->Where(function ($query)
use ($params) {
                                                if ($params-
>negative === false) {
                                                        $query-
>where('articles.negative', '!=', true);
                                                }
                                        });
                                if (!empty($params['author'])) {
                                        $articles = $articles-
>where('articles.author', $params['author']);
                                }

                                if ($params->only_video === true) {
                                        $articles = $articles-
```

```
>whereHas('articleAttachments', function ($query) use ($params,
$AccessDenied) {
                                                $query-
>Where(function ($query) {
                                                        $query-
>orWhere('type', 'embed');
                                                        $query-
>orWhere('type', 'video');
                                                });
                                        });
                                }
                                if ($params->theme_day === true) {
                                        $articles = $articles-
>where('articles.theme_day', true);
                                }

                                $articles = $articles-
>orderBy('articles.pubdate', $params->order);

                                $countAll = $articles-
>count('articles.id');
                                //
                                $countAll = 0;

                                //
$next = $articles
                                //
        ->limit(1)
                                //
        ->offset($params->offset + $params->count + 1)
                                //
        ->get()
                                //
        ->first()
                                //
        ? true
                                //
        : false;

                                $articles = $articles
                                        ->limit($params->count)
                                        ->offset($params->offset)
                                        ->get();

                                if (in_array('tags', $fields)) {
                                        $load['articleTags'] =
function ($query) use ($params) {
                                                $query-
>select(['slug', 'title']);
                                        };
                                }

                                if (in_array('categories', $fields))
{
```

```
                                        $load['articleCategories'] =
function ($query) use ($params) {
                                                $query-
>select(['id', 'slug', 'cover'])->with([
                                                        'localizatio
n' => function ($localization) use ($params) {

$localization->select('category_id', 'title', 'description');

$localization->where('lang', $params['localization']);
                                                        },
                                                ]);
                                        };
                                }

                                if (in_array('target', $fields)) {
                                        $load['articleTarget'] =
function ($query) {
                                                $query->select();
                                        };
                                }

                                if (in_array('is_recommendation',
$fields)) {

$load['articleRecommendation'] = function ($query) use ($params) {
                                                $query-
>where('status', 10);
                                        };
                                }

                                if (in_array('author', $fields)) {
                                        $load['author'] = function
($query) use ($params, $AccessDenied) {
                                                if ($AccessDenied !
== false) {
                                                        $query-
>select(['id', 'avatar', 'nicName', 'name', 'surname']);
                                                } else {
                                                        $query-
>select(['id', 'avatar', 'nicName']);
                                                }
                                        };
                                }
                                if (in_array('cover', $fields)) {
                                        $load['cover'] = function
($query) use ($params) {
                                                $query->select([
                                                        'files.uuid'
,
                                                        'files.width
',
                                                        'files.heigh
t',
                                                        'files.type'
,
                                                        'files.size'
,
                                                        'files.path'
,
                                                        'files.file'
,
                                                        'file_additi
onal_fields.sizes',
                                                ]);
                                                $query-
>leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                        };
                                }
                                if (in_array('articleLangs',
$fields)) {
                                        $load['articleLangs'] =
function ($query) use ($params) {
                                                $query-
>select(['lang']);
                                        };
                                }

                                $load['localization'] = function
($query) use ($params) {
                                        $query->where('lang',
$params['localization']);
                                };
                                if (in_array('attachments', $fields)
|| in_array('text', $fields)) {
                                        $load['articleAttachments']
= function ($query) use ($params) {
                                                $query->with([
                                                        'file' =>
function ($query) {

$query->leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                                        },
                                                ]);
                                        };
                                }

                                $articles = $articles->load($load);

                                $articles = $articles->map(function
($article) use ($builderQuery, $fields) {
                                        // if (in_array('cover',
$fields) && gettype($article->cover) === "object") {
                                        //      $article->cover-
>setAppends(['sizes'])->makeHidden(['additionalFields']);
                                        // }
                                        if (in_array('categories',
$fields)) {
                                                $article-
>articleCategories = $article->articleCategories->map(function
($category) {
                                                        return
$category->setAppends(['description', 'title'])-
>makeHidden(['localization']);
                                                });
                                        }
                                        return $builderQuery-
>setFieldsModel($article);
                                });

                                $response = [
                                        'response' => 'ok',
                                        'items' => $articles,
                                        'count' => $countAll,
                                        //
                'next' => $next,
                                ];

                                if (!empty($params['author'])) {
                                        $response['author'] =
$author;
                                }

                                return response()->json(
                                        $response,
                                        200,
                                        [
                                                'Content-Type' =>
'application/json;charset=UTF-8',
                                                'Charset' =>
'utf-8',
                                        ],
                                        JSON_UNESCAPED_UNICODE
                                );
                        });
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                        'server' => $e-
>getMessage(),
                                ],
                                000
                        );
                }
        }

        public static function getById($params, $fields)
        {
                try {
                        $cache = new
CacheHelper(md5('feed.getById|' . json_encode($params->all())));
                        return $cache->return(5, function () use
($params, $fields) {
                                $AccessDenied = false;
                                $fieldsAllowed = self::
$fieldsAllowed;
                                $validRole = ['super-admin', 'chief-
editor', 'middle-editor', 'editor'];
                                $validAbilities = ['all-edit',
'news-edit'];

                                array_push($fieldsAllowed,
'longread');

                                if (Auth::guard('api')->check()) {
                                        $user = Auth::guard('api')-
>user();
                                        $rolesUser = $user-
>getRoles()->toArray();
                                        $abilitiesUser = $user
                                                ->getAbilities()
                                                ->map(function
($item) {
                                                        return
$item->name;
                                                })
                                                ->toArray();

                                        $IR =
array_intersect($validRole, $rolesUser);
                                        $IA =
array_intersect($validAbilities, $abilitiesUser);
                                        if (count($IR) > 0) {
                                                if (in_array('all-
edit', $IA) || in_array('news-edit', $IA)) {

$AccessDenied = true;
                                                }
                                        }
                                }

                                if ($AccessDenied !== false) {
                                        array_push($fieldsAllowed,
'status');
                                }

                                if (is_null($params->lang)) {
                                        $params->merge(['lang' =>
self::$defaultLang]);
                                }
                                if (is_null($params->localization)
|| !in_array($params['localization'], self::$langAllowed)) {
                                        $params-
>merge(['localization' => self::$defaultLocalization]);
                                }
                                if (is_null($params->translated)) {
                                        $params->merge(['translated'
=> self::$defaultTranslated]);
                                }
                                $params->merge(['translated' =>
is_true($params->translated)]);

                                try {
                                        new ValidatorHelper($params-
>all(), [
                                                'rules' => [
                                                        'id' =>
['required'],
                                                        'fields' =>
['filled'],
                                                        'lang' =>
['required'],
                                                        'localizatio
n' => ['required'],
                                                        'translated'
=> ['required'],
                                                ],
                                                'alias' => [],
                                        ]);
                                } catch (ValidatorException $e) {
                                        throw new ApiException($e-
>getDecodedMessage(), 101);
                                }

                                $defaultSelect = array_merge(self::
$defaultSelect, [
                                        'text' => true,
                                        'categories' => true,
                                        'tags' => true,
                                        'cover' => true,
                                ]);

                                $builderQuery = new BuilderQuery([
                                        'getCount' => false,
                                        'fields' => [
                                                'return' => !
empty($fields) ? explode(',', $fields) : [],
                                                'default' =>
$defaultSelect,
                                                'allowed' =>
$fieldsAllowed,
                                                'critically' =>
['id', 'params'],
                                        ],
                                        'selectHidden' => self::
```

```
$defaultSelectHidden,
                                ]);

                                $fields = $builderQuery-
>getFields();

                                $langs_ = explode(',',
$params['lang']);
                                $langs = [];
                                foreach ($langs_ as $lang) {
                                        if (in_array($lang, self::
$langAllowed)) {
                                                $langs[] = $lang;
                                        }
                                }

                                $load = [];

                                $articles =
ArticleModel::where('id', $params->id)
                                        ->select($builderQuery-
>getSelect())
                                        ->whereHas('localization',
function ($query) use ($langs, $params, $AccessDenied) {
                                                $query-
>where(function ($where) use ($langs) {
                                                        foreach
($langs as $lang) {

$where->orWhere('lang', $lang);
                                                        }
                                                });
                                                if
($params['translated'] == true || $AccessDenied === false) {
                                                        $query-
>where('lang', $params['localization']);
                                                }
                                        })
                                        -
>whereHas('articleCategories', function ($query) use ($params) {
                                                $query-
>whereHas('localization', function ($query) use ($params) {
                                                        $query-
>where('lang', $params['localization']);
                                                });
                                        });

                                if (!$AccessDenied) {
                                        $articles = $articles
                                                ->where('pubdate',
'<', Carbon::now())
                                                ->where('status',
10)
                                                -
```

```
>whereHas('articleCategories', function ($query) {
                                                        $query-
>where('status', 10);
                                                });
                                }

                                if (in_array('is_recommendation',
$fields)) {

$load['articleRecommendation'] = function ($query) use ($params) {
                                                $query-
>where('status', 10);
                                        };
                                }

                                if (in_array('tags', $fields)) {
                                        $load['articleTags'] =
function ($query) use ($params) {
                                                $query-
>select(['slug', 'title']);
                                        };
                                }

                                if (in_array('categories', $fields))
{
                                        $load['articleCategories'] =
function ($query) use ($params) {
                                                $query-
>select(['id', 'slug', 'cover'])->with([
                                                        'localizatio
n' => function ($localization) use ($params) {

$localization->select('category_id', 'title', 'description');

$localization->where('lang', $params['localization']);
                                                        },
                                                ]);
                                        };
                                }

                                if (in_array('author', $fields)) {
                                        $load['author'] = function
($query) use ($params, $AccessDenied) {
                                                if ($AccessDenied !
== false) {
                                                        $query-
>select(['id', 'avatar', 'nicName', 'name', 'surname']);
                                                } else {
                                                        $query-
>select(['id', 'avatar', 'nicName']);
                                                }
                                        };
                                }
                                if (in_array('cover', $fields)) {
                                        $load['cover'] = function
($query) use ($params) {
                                                $query->select([
                                                        'files.uuid'
,
                                                        'files.width
',
                                                        'files.heigh
t',
                                                        'files.type'
,
                                                        'files.size'
,
                                                        'files.path'
,
                                                        'files.file'
,
                                                        'file_additi
onal_fields.sizes',
                                                ]);
                                                $query-
>leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                        };
                                }

                                if (in_array('target', $fields)) {
                                        $load['articleTarget'] =
function ($query) {
                                                $query->select();
                                        };
                                }

                                $load['localization'] = function
($query) use ($params) {
                                        $query->where('lang',
$params['localization']);
                                };

                                if (in_array('attachments', $fields)
|| in_array('text', $fields)) {
                                        $load['articleAttachments']
= function ($query) use ($params) {
                                                $query->with([
                                                        'file' =>
function ($query) {

$query->leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                                        },
                                                ]);
                                        };
                                }
                                $articles = $articles->get()-
>load($load);

                                if (!empty($articles)) {
                                        $article = $articles
                                                ->map(function
($article) use ($builderQuery, $fields) {
                                                        if
(in_array('categories', $fields) && $article->articleCategories) {

$article->articleCategories = $article->articleCategories-
>map(function ($category) {

        return $category->setAppends(['description', 'title'])-
>makeHidden(['localization']);
                                                                });
                                                        }
                                                        return
$builderQuery->setFieldsModel($article);
                                                })
                                                ->first();
                                        if ($article) {
                                                return response()-
>json(
                                                        [
                                                                'res
ponse' => 'ok',
                                                                'art
icle' => $article,
                                                        ],
                                                        200,
                                                        [
                                                                'Con
tent-Type' => 'application/json;charset=UTF-8',
                                                                'Cha
rset' => 'utf-8',
                                                        ],

JSON_UNESCAPED_UNICODE
                                                );
                                        } else {
                                                throw new
ApiException(
                                                        [
                                                                'art
icle' => 'Article not found',
                                                        ],
                                                        404
                                                );
                                        }
                                } else {
                                        throw new ApiException(
                                                [
                                                        'article' =>
'Article not found',
                                                ],
                                                404
                                        );
                                }
                        });
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        // dd($e);
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                        'server1' => $e-
>getMessage(),
                                ],
                                000
                        );
                }
        }

        /**
         * @throws ApiException
         */
        public static function getSimilarArticles(Request $request,
string $fields = '')
        {
                try {
                        $cache = new
CacheHelper(md5('feed.getById|' . json_encode($request->all())));

                        //                      return $cache-
>return(5, function () use ($params, $fields) {
                        if (is_null($request->get('lang'))) {
                                $request->merge(['lang' => self::
$defaultLang]);
                        }
                        if (is_null($request->get('localization'))
|| !in_array($request['localization'], self::$langAllowed)) {
                                $request->merge(['localization' =>
self::$defaultLocalization]);
                        }
                        if (is_null($request->get('translated'))) {
                                $request->merge(['translated' =>
self::$defaultTranslated]);
                        }
                        $request->merge(['translated' =>
is_true($request->get('translated'))]);

                        if (is_null($request->get('count'))) {
                                $request->merge(['count' => self::
$defaultCount]);
                        }
                        try {
                                new ValidatorHelper($request->all(),
[
                                        'rules' => [
                                                'id' =>
['required'],
                                                'count' =>
['filled'],
                                                'fields' =>
['filled'],
                                                'lang' =>
['required'],
                                                'localization' =>
['required'],
                                                'translated' =>
['required'],
                                        ],
                                        'alias' => [],
                                ]);
                        } catch (ValidatorException $e) {
                                throw new ApiException($e-
>getDecodedMessage(), 101);
                        }

                        $languages = collect(explode(',', $request-
>get('lang')))
                                ->filter(function ($lang) {
                                        return in_array($lang,
self::$langAllowed);
                                })
                                ->toArray();

                        $article = ArticleModel::select('id')
                                ->where('id', $request->get('id'))
                                ->whereHas('localization', function
($query) use ($languages, $request) {
                                        $query->where(function
($where) use ($languages) {
                                                foreach ($languages
as $lang) {
                                                        $where-
>orWhere('lang', $lang);
                                                }
                                        });
                                        $query->where('lang',
$request->get('localization'));
                                })
                                ->where('pubdate', '<',
Carbon::now())
                                ->where('status', 10)
                                ->whereHas('articleCategories',
function ($query) {
                                        $query->where('status', 10);
                                })
                                ->first();
                        if ($article) {
                                $article = $article->load([
                                        'localization' =>
ArticleModel::defineSearchLoad()['localization'],
                                        'articleTags' =>
ArticleModel::defineSearchLoad()['articleTags'],
                                ]);

                                $tags = $article
                                        ->getTags()
                                        ->filter(function (Tag $tag)
{
                                                return
mb_strlen($tag->title) > 2;
                                        })
                                        ->values()
                                        ->pluck('title')
                                        ->map(function ($value) {
                                                return
mb_strtolower($value);
                                        })
                                        ->toArray();

                                //
dd($tags);

                                $result =
ElasticSearch::get('article', [
                                        'size' => $request-
>get('count'),
                                        'sort' => [],
                                        'query' => [
                                                'bool' => [
                                                        'must_not'
=> [
                                                                [

        'match' => [

                'id' => $article->id,

        ],
                                                                ],
                                                        ],
                                                        'must' => [
                                                                [

        'match' => [

                'lang' => $request->get('localization'),

        ],
                                                                ],
                                                                [
        'match' => [
                'type' => 'news',
        ],
                                                                ],
                                                                [
        'range' => [
                'pubdate' => [
                        'gte' => Carbon::now()
                                ->addMonths(-1)
                                ->toISOString(),
                        'lte' => Carbon::now()->toISOString(),
                ],
        ],
                                                                ],
                                                                [
        'match' => [
                'title' => [
                        'fuzziness' => 2,
                        'query' => $article->title,
                ],
        ],
                                                                ],
                                                                //
                                                                [
                                                                //
        'terms' => [
                                                                //
                'boost' => 1,
                                                                //
                'tags' => $tags,
                                                                //
```

```
        ],
                                                                //
                                                                ],
                                                        ],
                                                ],
                                        ],
                                ]);

                                $ids = collect($result-
>get('hits'))->map(function (array $item) {
                                        return $item['_source']
['id'];
                                });

                                $builderQuery = new BuilderQuery([
                                        'getCount' => false,
                                        'fields' => [
                                                'return' => !
empty($fields) ? explode(',', $fields) : [],
                                                'default' => self::
$defaultSelect,
                                                'allowed' => self::
$fieldsAllowed,
                                                'critically' =>
['id'],
                                        ],
                                        'table' => 'articles',
                                        'selectHidden' => self::
$defaultSelectHidden,
                                ]);
                                $fields = $builderQuery-
>getFields();
                                $articles =
ArticleModel::select($builderQuery->getSelect())
                                        ->whereIn('articles.id',
$ids)
                                        -
>join('article_localizations', function ($join) use ($request,
$languages) {
                                                $join
                                                        -
>on('articles.id', '=', 'article_localizations.article_id')
                                                        -
>where('article_localizations.lang', $request->get('localization'))
                                                        -
>where(function ($where) use ($languages) {

foreach ($languages as $lang) {

        $where->orWhere('lang', $lang);
                                                                }
                                                        });
                                        })
                                        ->Where(function ($query) {
                                                $query
                                                        -
>where('longread', false)
                                                        -
>where('pubdate', '<=', Carbon::now())
                                                        -
>where('status', 10)
                                                        -
>whereHas('articleCategories', function ($query) {

$query->where('status', 10);
                                                        });
                                        });

                                $articles = $articles->get();

                                if (in_array('tags', $fields)) {
                                        $load['articleTags'] =
function ($query) {
                                                $query-
>select(['slug', 'title']);
                                        };
                                }

                                if (in_array('categories', $fields))
{
                                        $load['articleCategories'] =
function ($query) use ($request) {
                                                $query-
>select(['id', 'slug', 'cover'])->with([
                                                        'localizatio
n' => function ($localization) use ($request) {

$localization->select('category_id', 'title', 'description');

$localization->where('lang', $request->get('localization'));
                                                        },
                                                ]);
                                        };
                                }

                                if (in_array('target', $fields)) {
                                        $load['articleTarget'] =
function ($query) {
                                                $query->select();
                                        };
                                }

                                if (in_array('is_recommendation',
$fields)) {

$load['articleRecommendation'] = function ($query) {
                                                $query-
>where('status', 10);
```

```
                                        };
                                }

                                if (in_array('author', $fields)) {
                                        $load['author'] = function
($query) {
                                                $query-
>select(['id', 'avatar', 'nicName']);
                                        };
                                }
                                if (in_array('cover', $fields)) {
                                        $load['cover'] = function
($query) {
                                                $query->select([
                                                        'files.uuid'
,
                                                        'files.width
',
                                                        'files.heigh
t',
                                                        'files.type'
,
                                                        'files.size'
,
                                                        'files.path'
,
                                                        'files.file'
,
                                                        'file_additi
onal_fields.sizes',
                                                ]);
                                                $query-
>leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                        };
                                }
                                if (in_array('articleLangs',
$fields)) {
                                        $load['articleLangs'] =
function ($query) use ($request) {
                                                $query-
>select(['lang']);
                                        };
                                }

                                $load['localization'] = function
($query) use ($request) {
                                        $query->where('lang',
$request->get('localization'));
                                };
                                if (in_array('attachments', $fields)
|| in_array('text', $fields)) {
                                        $load['articleAttachments']
= function ($query) use ($request) {
```

```
                                                $query->with([
                                                        'file' =>
function ($query) {

$query->leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                                        },
                                                ]);
                                        };
                                }

                                //
$articles = ;

                                return response()->json(
                                        [
                                                'response' => 'ok',
                                                'items' =>
$articles->load($load)->map(function ($article) use ($builderQuery,
$fields) {
                                                        if
(in_array('categories', $fields)) {

$article->articleCategories = $article->articleCategories-
>map(function ($category) {

        return $category->setAppends(['description', 'title'])-
>makeHidden(['localization']);
                                                                });
                                                        }
                                                        return
$builderQuery->setFieldsModel($article);
                                                }),
                                        ],
                                        200,
                                        [
                                                'Content-Type' =>
'application/json;charset=UTF-8',
                                                'Charset' =>
'utf-8',
                                        ],
                                        JSON_UNESCAPED_UNICODE
                                );
                        } else {
                                throw new ApiException(
                                        [
                                                'article' =>
'Article not found',
                                        ],
                                        404
                                );
                        }
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        dd($e);
                        throw new ApiException(
                                [
                                        'server' => 'oops!',
                                        'server1' => $e-
>getMessage(),
                                ],
                                000
                        );
                }
        }

        public static function getItemLongread($params, $fields)
        {
                try {
                        $cache = new
CacheHelper(md5('feed.getItemLongread|' . json_encode($params-
>all())));
                        return $cache->return(5, function () use
($params, $fields) {
                                $AccessDenied = false;
                                $fieldsAllowed = self::
$fieldsAllowed;

                                array_push($fieldsAllowed,
'longread');
                                $validRole = ['super-admin', 'chief-
editor', 'middle-editor', 'editor'];
                                $validAbilities = ['all-edit',
'news-edit'];
                                if (Auth::guard('api')->check()) {
                                        $user = Auth::guard('api')-
>user();
                                        $rolesUser = $user-
>getRoles()->toArray();
                                        $abilitiesUser = $user
                                                ->getAbilities()
                                                ->map(function
($item) {
                                                        return
$item->name;
                                                })
                                                ->toArray();

                                        $IR =
array_intersect($validRole, $rolesUser);
                                        $IA =
array_intersect($validAbilities, $abilitiesUser);
                                        if (count($IR) > 0) {
                                                if (in_array('all-
edit', $IA) || in_array('news-edit', $IA)) {

$AccessDenied = true;
```

```
                                                }
                                        }
                                }

                                if ($AccessDenied !== false) {
                                        array_push($fieldsAllowed,
'status');
                                }

                                if (is_null($params->lang)) {
                                        $params->merge(['lang' =>
self::$defaultLang]);
                                }
                                if (is_null($params->localization)
|| !in_array($params['localization'], self::$langAllowed)) {
                                        $params-
>merge(['localization' => self::$defaultLocalization]);
                                }
                                if (is_null($params->translated)) {
                                        $params->merge(['translated'
=> self::$defaultTranslated]);
                                }
                                $params->merge(['translated' =>
is_true($params->translated)]);

                                try {
                                        new ValidatorHelper($params-
>all(), [
                                                'rules' => [
                                                        'slug' =>
['required'],
                                                        'fields' =>
['filled'],
                                                        'lang' =>
['required'],
                                                        'localizatio
n' => ['required'],
                                                        'translated'
=> ['required'],
                                                ],
                                                'alias' => [],
                                        ]);
                                } catch (ValidatorException $e) {
                                        throw new ApiException($e-
>getDecodedMessage(), 101);
                                }

                                $defaultSelect = array_merge(self::
$defaultSelect, [
                                        'text' => true,
                                        'categories' => true,
                                        'tags' => true,
                                        'cover' => true,
                                ]);
                                $builderQuery = new BuilderQuery([
                                        'getCount' => false,
                                        'fields' => [
                                                'return' => !
empty($fields) ? explode(',', $fields) : [],
                                                'default' =>
$defaultSelect,
                                                'allowed' =>
$fieldsAllowed,
                                                'critically' =>
['id', 'params'],
                                        ],
                                        'selectHidden' => self::
$defaultSelectHidden,
                                ]);

                                $fields = $builderQuery-
>getFields();

                                $langs_ = explode(',',
$params['lang']);
                                $langs = [];
                                foreach ($langs_ as $lang) {
                                        if (in_array($lang, self::
$langAllowed)) {
                                                $langs[] = $lang;
                                        }
                                }

                                $load = [];

                                $articles =
ArticleModel::where('slug', $params->slug)
                                        ->select($builderQuery-
>getSelect())
                                        ->where('longread', true)
                                        ->whereHas('localization',
function ($query) use ($langs, $params, $AccessDenied) {
                                                $query-
>where(function ($where) use ($langs) {
                                                        foreach
($langs as $lang) {

$where->orWhere('lang', $lang);
                                                        }
                                                });
                                                if
($params['translated'] == true || $AccessDenied === false) {
                                                        $query-
>where('lang', $params['localization']);
                                                }
                                        })
                                        -
```

```
>whereHas('articleCategories', function ($query) use ($params) {
                                                $query-
>whereHas('localization', function ($query) use ($params) {
                                                        $query-
>where('lang', $params['localization']);
                                                });
                                        });

                                if (!$AccessDenied) {
                                        $articles = $articles
                                                ->where('pubdate',
'<', Carbon::now())
                                                ->where('status',
10)
                                                -
>whereHas('articleCategories', function ($query) {
                                                        $query-
>where('status', 10);
                                                });
                                }

                                if (in_array('is_recommendation',
$fields)) {

$load['articleRecommendation'] = function ($query) use ($params) {
                                                $query-
>where('status', 10);
                                        };
                                }

                                if (in_array('tags', $fields)) {
                                        $load['articleTags'] =
function ($query) use ($params) {
                                                $query-
>select(['slug', 'title']);
                                        };
                                }

                                if (in_array('categories', $fields))
{
                                        $load['articleCategories'] =
function ($query) use ($params) {
                                                $query-
>select(['id', 'slug', 'cover'])->with([
                                                        'localizatio
n' => function ($localization) use ($params) {

$localization->select('category_id', 'title', 'description');

$localization->where('lang', $params['localization']);
                                                        },
                                                ]);
                                        };
                                }
                                if (in_array('author', $fields)) {
                                        $load['author'] = function
($query) use ($params, $AccessDenied) {
                                                if ($AccessDenied !
== false) {
                                                        $query-
>select(['id', 'avatar', 'nicName', 'name', 'surname']);
                                                } else {
                                                        $query-
>select(['id', 'avatar', 'nicName']);
                                                }
                                        };
                                }

                                if (in_array('cover', $fields)) {
                                        $load['cover'] = function
($query) use ($params) {
                                                $query->select([
                                                        'files.uuid'
,
                                                        'files.width
',
                                                        'files.heigh
t',
                                                        'files.type'
,
                                                        'files.size'
,
                                                        'files.path'
,
                                                        'files.file'
,
                                                        'file_additi
onal_fields.sizes',
                                                ]);
                                                $query-
>leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                        };
                                }

                                if (in_array('target', $fields)) {
                                        $load['articleTarget'] =
function ($query) {
                                                $query->select();
                                        };
                                }

                                $load['localization'] = function
($query) use ($params) {
                                        $query->where('lang',
$params['localization']);
                                };
                                if (in_array('attachments', $fields)
|| in_array('text', $fields)) {
                                        $load['articleAttachments']
= function ($query) use ($params) {
                                                $query->with([
                                                        'file' =>
function ($query) {

$query->leftJoin('file_additional_fields', 'files.uuid', '=',
'file_additional_fields.file_uuid');
                                                        },
                                                ]);
                                        };
                                }

                                $articles = $articles->get()-
>load($load);

                                if (!empty($articles)) {
                                        $article = $articles
                                                ->map(function
($article) use ($builderQuery, $fields) {
                                                        return
$builderQuery->setFieldsModel($article);
                                                })
                                                ->first();

                                        if ($article) {
                                                return response()-
>json(
                                                        [
                                                                'res
ponse' => 'ok',
                                                                'art
icle' => $article,
                                                        ],
                                                        200,
                                                        [
                                                                'Con
tent-Type' => 'application/json;charset=UTF-8',
                                                                'Cha
rset' => 'utf-8',
                                                        ],

JSON_UNESCAPED_UNICODE
                                                );
                                        } else {
                                                throw new
ApiException(
                                                        [
                                                                'art
icle' => 'Article not found',
                                                        ],
                                                        404
                                                );
                                        }
                                } else {
                                        throw new ApiException(
                                                [
                                                        'article' =>
'Article not found',
                                                ],
                                                404
                                        );
                                }
                        });
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        // dd($e);
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                ],
                                000
                        );
                }
            }

            public static function editStatus($params, $action)
            {
                try {
                        $AccessDenied = false;
                        $validRole = ['super-admin', 'chief-editor',
'middle-editor', 'editor'];
                        $validAbilities = ['all-edit', 'news-edit'];
                        $IR = $IA = [];
                        if (Auth::guard('api')->check()) {
                                $user = Auth::guard('api')->user();
                                $rolesUser = $user->getRoles()-
>toArray();

                                $abilitiesUser = $user
                                        ->getAbilities()
                                        ->map(function ($item) {
                                                return $item->name;
                                        })
                                        ->toArray();

                                $IR = array_intersect($validRole,
$rolesUser);
                                $IA =
array_intersect($validAbilities, $abilitiesUser);

                                $IR = array_values($IR);
                                $IA = array_values($IA);
                                if (count($IR) > 0 && count($IA) >
0) {
```

```
                                                $AccessDenied = true;
                                        }
                                }
                                if (!$AccessDenied) {
                                        throw new ApiException(
                                                [
                                                        'method' => 'access
denied',
                                                ],
                                                403
                                        );
                                }

                                try {
                                        new ValidatorHelper($params->all(),
[
                                                'rules' => [
                                                        'id' =>
['required'],
                                                ],
                                                'alias' => [],
                                        ]);
                                } catch (ValidatorException $e) {
                                        throw new ApiException($e-
>getDecodedMessage(), 101);
                                }

                                $status = null;
                                switch ($action) {
                                        case 'delete':
                                                $status = -100;
                                                break;
                                        case 'restore':
                                                $status = 10;
                                                break;
                                }

                                $article = ArticleModel::where('id',
$params->id)
                                        ->where('status', '!=', $status)
                                        ->first();

                                if (!$article) {
                                        throw new ApiException(
                                                [
                                                        'article' =>
'Article not found',
                                                ],
                                                404
                                        );
                                }

                                if (in_array('all-edit', $IA) ||
in_array('news-edit', $IA)) {
```

```
                                $AccessDenied = true;
                                if (count($IR) === 1 && $IR[0] ===
'editor') {
                                        if (($article->author ===
$user->id) === true) {
                                                $AccessDenied =
true;
                                        } else {
                                                $AccessDenied =
false;
                                        }
                                }
                        } else {
                                $AccessDenied = false;
                        }

                        if (!$AccessDenied) {
                                throw new ApiException(
                                        [
                                                'method' => 'access
denied',
                                        ],
                                        403
                                );
                        }

                        $article->status = $status;
                        $article->save();
                        event(new ArticleEvent($article));
                        return response()->json([
                                'response' => 'ok',
                        ]);
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                ],
                                000
                        );
                }
        }

        public static function save($params, $action)
        {
                try {
                        $AccessDenied = false;
                        $validRole = ['super-admin', 'chief-editor',
'middle-editor', 'editor'];
                        $validAbilities = ['all-edit', 'news-edit'];
                        $IR = $IA = [];
                        if (Auth::guard('api')->check()) {
                                $user = Auth::guard('api')->user();
                                $rolesUser = $user->getRoles()-
>toArray();
                                $abilitiesUser = $user
                                        ->getAbilities()
                                        ->map(function ($item) {
                                                return $item->name;
                                        })
                                        ->toArray();

                                $IR = array_intersect($validRole,
$rolesUser);
                                $IA =
array_intersect($validAbilities, $abilitiesUser);

                                $IR = array_values($IR);
                                $IA = array_values($IA);

                                if (count($IR) > 0 && count($IA) >
0) {
                                        $AccessDenied = true;
                                }
                        }
                        if (!$AccessDenied) {
                                throw new ApiException(
                                        [
                                                'method' => 'access
denied',
                                        ],
                                        403
                                );
                        }

                        $validatorRules = [
                                'tags' => ['required', 'string',
'keywords'],
                                'categories' => ['required',
'string', 'keywords'],

                                'pubdate' => ['filled', 'string'],
                                'localization' => ['required',
'array'],
                        ];
                        if ($action === 'edit') {
                                $validatorRules = array_merge(
                                        [
                                                'id' =>
['required'],
                                                'slug' =>
['present'],
                                        ],
                                        $validatorRules
                                );
                        }
                        try {
                                new ValidatorHelper($params->all(),
[
                                        'rules' => $validatorRules,
                                        'alias' => [
                                                'localization' =>
'localizationArray',
                                        ],
                                ]);
                        } catch (ValidatorException $e) {
                                throw new ApiException($e-
>getDecodedMessage(), 101);
                        }

                        $localizations = [];

                        if ($action === 'add') {
                                $article = new ArticleModel();
                                $article->params = [];
                                $article->author = $user->id;
                        }
                        if ($action === 'edit') {
                                $article =
ArticleModel::find($params->id);
                                if (!$article) {
                                        throw new ApiException(
                                                [
                                                        'article' =>
'Article not found',
                                                ],
                                                404
                                        );
                                }
                                if (in_array('all-edit', $IA) ||
in_array('news-edit', $IA)) {
                                        $AccessDenied = true;
                                        if (count($IR) === 1 &&
$IR[0] === 'editor') {
                                                if (($article-
>author === $user->id) === true) {

$AccessDenied = true;
                                                } else {

$AccessDenied = false;
                                                }
                                        }
                                } else {
                                        $AccessDenied = false;
                                }
                                if ($AccessDenied === false) {
                                        throw new ApiException(
                                                [
                                                        'method' =>
'access denied',
                                                ],
                                                403
                                        );
                                }
                        }

                        $attachments = $article->articleAttachments
                                ->map(function ($item) {
                                        return $item->uuid;
                                })
                                ->toArray();

                        foreach ($params->localization as $key =>
$localization) {
                                try {
                                        new
ValidatorHelper($localization, [
                                                'rules' => [
                                                        'lang' =>
['required'],
                                                        'title' =>
['required', 'string', 'min:30', 'max:100'],
                                                        'description
' => ['required', 'string', 'min:140', 'max:250'],
                                                        'text' =>
['required', 'array'],
                                                ],
                                                'alias' => [],
                                        ]);
                                } catch (ValidatorException $e) {
                                        throw new ApiException($e-
>getDecodedMessage(), 101);
                                }

                                $uniqueTitle =
ArticleModel::whereHas('localization', function ($query) use
($localization) {
                                        $query->where('title',
$localization['title']);
                                });

                                if (!empty($article->id)) {
                                        $uniqueTitle = $uniqueTitle-
>where('id', '!=', $article->id);
                                }
                                $uniqueTitle = $uniqueTitle-
>count();

                                // print_r($uniqueTitle);
                                if ($uniqueTitle > 0) {
                                        // На Башкирию обрушилась
сильная метель

                                        $htmlMessage =
"*Посягательство на SEO оптимизацию @kaltfeuer*\n";
                                        $htmlMessage .= "Повтор
заголовка\n";
                                        $htmlMessage .=
"`{$localization['title']}`";
                                        $htmlMessage .= "\nАвтор:
" . $user->name . ' ' . $user->surname;

                                        $keyboard = [
                                                'inline_keyboard' =>
[
                                                        [
                                                                //
['text' => 'Помиловать', "callback_data" => "dfdsfdsfsfd"]
                                                        ],
                                                ],
                                        ];
                                        if (!empty($article->id)) {
                                                // ['text' =>
'Открыть', 'url' => "https://admin.bash.news/news/{$article->id}"]

array_push($keyboard['inline_keyboard'][0], [
                                                        'text' =>
'Открыть',
                                                        'url' =>
"https://admin.bash.news/news/{$article->id}",
                                                ]);
                                        }

                                        //
                $encodedKeyboard =
Telegram::replyKeyboardMarkup($keyboard);

                                        $response =
Telegram::sendMessage([
                                                'chat_id' =>
-453685428,
                                                'parse_mode' =>
'markdown',
                                                'text' =>
$htmlMessage,
                                                'reply_markup' =>
json_encode($keyboard),
                                        ]);

                                        throw new ApiException(
                                                [
                                                        'article.tit
le' => 'Такой заголовок уже сущесвует',
                                                ],
                                                101
                                        );
                                }
```

```
                              try {
                                        $clearBlocks = new
ClearContentBlocks\ClearContentBlocks(

$localization['text'],
                                                      [
                                                              'video' => [
                                                                      'vid
eo' => [

        'type' => 'array',

        'data' => function ($value) {},
                                                                      ],
                                                              ],
                                                              'video' => [
                                                                      'vid
eo' => [

        'type' => 'array',

        'data' => function ($value) {

                if (array_key_exists('uuid', $value)) {

                        return [

                                'uuid' => [

                                        'type' => 'string',

                                        'required' => true,

                                ],

                        ];

                }

                if (array_key_exists('url', $value)) {

                        return [

                                'url' => [

                                        'type' => 'string',

                                        'required' => true,

                                ],

                        ];

                }
                return false;

        },
                                                                      ],
                                                              ],
                                                      ],
                                                      [
                                                              'noIndex' =>
function ($value) {
                                                                      if
(is_bool($value)) {

        return boolval($value);
                                                                      }

return false;
                                                              },
                                                      ]
                                        );
                                } catch
(ClearContentBlocks\ClearContentBlocksException $e) {
                                        throw new ApiException(
                                                      [
                                                              'text' =>
$e->getMessage(),
                                                      ],
                                                      101
                                        );
                                }

                                $tags = getIdTags(explode(',',
$params->tags));

                                if (count($tags) < 5 || count($tags)
> 10) {
                                        throw new ApiException(
                                                      [
                                                              'tag' =>
'Количество keyword = 5–10 слов',
                                                      ],
                                                      101
                                        );
                                }

                                $categories =
getIdCategories(explode(',', $params->categories));

                                list($text, $attachmentsCurrentLang)
= AttachmentHelper::setAttachmentBlocks($clearBlocks->getBlocks());
                                $localization['text'] = $text;
                                if
(empty($localization['description'])) {
                                        $localization['description']
```

```
= null;
                                }

                                $attachments =
array_merge($attachments, $attachmentsCurrentLang);

                                array_push($localizations,
$localization);
                        }

                        $attachments = array_unique($attachments,
SORT_REGULAR);
                        $attachments = array_values($attachments);

                        if ($params->pubdate) {
                                try {
                                        $pubdate =
Carbon::createFromTimeString($params->pubdate);
                                } catch (Exception $e) {
                                        throw new ApiException(
                                                [
                                                        'pubdate' =>
'Invalid date format',
                                                ],
                                                101
                                        );
                                }
                        } else {
                                $pubdate = Carbon::now();
                        }
                        if (!empty($params->slug)) {
                                $slug = $params->slug ?? '';
                        }

                        foreach ($localizations as $key =>
$localization) {
                                if ($action === 'add') {
                                        $slug =
getSlug($localization['title']);
                                }
                                $contentVideo =
array_filter($localization['text'], function ($item) {
                                        return $item['type'] ===
'video' || $item['type'] === 'videoPlayList';
                                });

                                if (count($contentVideo) > 0) {
                                        $article->in_video = true;
                                }
                                $contentImages =
array_filter($localization['text'], function ($item) {
                                        return $item['type'] ===
'images' || $item['type'] === 'gallery';
                                });
                                if (count($contentImages) > 0) {
                                        $article->in_images = true;
                                }
                                $contentImages =
array_filter($localization['text'], function ($item) {
                                        return $item['type'] ===
'audio' || $item['type'] === 'audioPlayList';
                                });
                                if (count($contentImages) > 0) {
                                        $article->in_audio = true;
                                }
                        }
                        $article->slug = $slug;
                        $article->cover = $params->cover;
                        $article->pubdate = $pubdate;
                        $article->save();
                        foreach ($localizations as $key =>
$localization) {
                                $article->localization()-
>updateOrCreate(
                                        ['lang' =>
$localization['lang']],
                                        [
                                                'title' =>
getShortenString($localization['title'], 200),
                                                'text' =>
$localization['text'],
                                                'description' =>
$localization['description'],
                                        ]
                                );
                        }

                        $article->articleCategories()->detach();
                        $article->articleCategories()-
>attach($categories);

                        $article->articleAttachments()->detach();
                        $article->articleAttachments()-
>attach($attachments);

                        $article->articleTags()->detach();
                        $article->articleTags()->attach($tags);
                        $article->save();

                        if ($action === 'add') {
                                $valid = [
                                        'yandex-news',
                                        // 'yandex-zen',
                                        'yandex-turbo',
                                        'amp',
                                ];
                                foreach ($valid as $value) {
                                        $article->articleTarget()-
```

```
>updateOrCreate(['key' => $value], []);
                                }
                        }

                        event(new ArticleEvent($article));
                        return response()->json([
                                'response' => 'ok',
                                'article' => $article,
                        ]);
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                        'server2' => $e-
>getMessage(),
                                ],
                                000
                        );
                }
        }

        public static function getRecommended($params, $fields)
        {
                try {
                        $cache = new
CacheHelper(md5('feed.getRecommended|' . json_encode($params-
>all())));
                        return $cache->return(5, function () use
($params, $fields) {
                                $AccessDenied =
self::checkAccessDenied();
                                $fieldsAllowed = self::
$fieldsAllowed;

                                if (is_null($params->lang)) {
                                        $params->merge(['lang' =>
self::$defaultLang]);
                                }
                                if (is_null($params->localization)
|| !in_array($params['localization'], self::$langAllowed)) {
                                        $params-
>merge(['localization' => self::$defaultLocalization]);
                                }
                                if (is_null($params->translated)) {
                                        $params->merge(['translated'
=> self::$defaultTranslated]);
                                }

                                if (is_null($params->count)) {
                                        $params->merge(['count' =>
5]);
                                }
                                if (is_null($params->offset)) {
                                        $params->merge(['offset' =>
self::$defaultOffset]);
                                }

                                $params->merge(['translated' =>
is_true($params->translated)]);

                                try {
                                        new ValidatorHelper($params-
>all(), [
                                                'rules' => [
                                                        'id' =>
['required'],
                                                        'fields' =>
['filled'],

                                                        'count' =>
['filled'],
                                                        'offset' =>
['filled'],

                                                        'lang' =>
['required'],
                                                        'localizatio
n' => ['required'],
                                                        'translated'
=> ['required'],
                                                ],
                                                'alias' => [],
                                        ]);
                                } catch (ValidatorException $e) {
                                        throw new ApiException($e-
>getDecodedMessage(), 101);
                                }

                                $article = ArticleModel::where('id',
$params->id)
                                        ->first()
                                        -
>load(['articleCategories']);

                                if ($article) {
                                        $defaultSelect =
array_merge(self::$defaultSelect, [
                                                'text' => true,
                                                'categories' =>
true,
                                                'tags' => true,
                                                'cover' => true,
                                        ]);

                                        $builderQuery = new
BuilderQuery([
                                                'getCount' => false,
                                                'fields' => [
                                                        'return'
=> !empty($fields) ? explode(',', $fields) : [],
                                                        'default' =>
$defaultSelect,
                                                        'allowed' =>
$fieldsAllowed,
                                                        'critically'
=> ['id', 'params'],
                                                ],
                                                'selectHidden' =>
self::$defaultSelectHidden,
                                        ]);

                                        $fields = $builderQuery-
>getFields();

                                        $langs_ = explode(',',
$params['lang']);
                                        $langs = [];
                                        foreach ($langs_ as $lang) {
                                                if (in_array($lang,
self::$langAllowed)) {
                                                        $langs[] =
$lang;
                                                }
                                        }

                                        $load = [];

                                        $categories = $article-
>articleCategories->map(function ($item) {
                                                return $item->id;
                                        });

                                        $articlesRecommended =
ArticleModel::select($builderQuery->getSelect())
                                                -
>where('is_recommendation', true)
                                                -
>whereHas('localization', function ($query) use ($langs, $params,
$AccessDenied) {
                                                        $query-
>where(function ($where) use ($langs) {

foreach ($langs as $lang) {

        $where->orWhere('lang', $lang);
                                                                }
                                                        });
                                                        if
($params['translated'] == true || $AccessDenied === false) {
```

```
$query->where('lang', $params['localization']);
                                                        }
                                                })
                                                -
>whereHas('articleCategories', function ($query) use ($params) {
                                                        $query-
>whereHas('localization', function ($query) use ($params) {

$query->where('lang', $params['localization']);
                                                        });
                                                })
                                                ->where('id', '!=',
$article->id)
                                                ->limit($params-
>count)
                                                ->offset($params-
>offset)
                                                ->orderBy('pubdate',
'desc');

                                        if ($AccessDenied === false)
{
                                                $articlesRecommended
= $articlesRecommended
                                                        -
>where('pubdate', '<', Carbon::now())
                                                        -
>where('status', 10)
                                                        -
>whereHas('articleCategories', function ($query) {

$query->where('status', 10);
                                                        });
                                        }

                                        $articlesRecommended =
$articlesRecommended->get();

                                        if
(count($articlesRecommended) < $params->count) {

$articlesRecommendedCategory = ArticleModel::select($builderQuery-
>getSelect())
                                                        -
>whereHas('localization', function ($query) use ($langs, $params,
$AccessDenied) {

$query->where(function ($where) use ($langs) {

        foreach ($langs as $lang) {

                $where->orWhere('lang', $lang);

        }
```

```
                                                            });
                                                            if
($params['translated'] == true || $AccessDenied === false) {

        $query->where('lang', $params['localization']);
                                                            }
                                                    })
                                                    -
>whereHas('articleCategories', function ($query) use ($params,
$categories) {

$query->whereHas('localization', function ($query) use ($params) {

        $query->where('lang', $params['localization']);
                                                            });

$query->Where(function ($query) use ($categories) {

        foreach ($categories as $key => $value) {

                $query->orWhere('id', $value);

        }
                                                            });
                                                    })
                                                    -
>where('id', '!=', $article->id)
                                                    -
>limit($params->count - count($articlesRecommended))
                                                    -
>offset($params->offset)
                                                    -
>orderBy('pubdate', 'desc');

                                            if ($AccessDenied
=== false) {

$articlesRecommendedCategory = $articlesRecommendedCategory
                                                            -
>where('pubdate', '<', Carbon::now())
                                                            -
>where('status', 10)
                                                            -
>whereHas('articleCategories', function ($query) {

        $query->where('status', 10);
                                                            });
                                            }


$articlesRecommendedCategory = $articlesRecommendedCategory->get();
                                            $articlesRecommended
= $articlesRecommended->concat($articlesRecommendedCategory);
                                    }
```

```
                                        if (in_array('tags',
$fields)) {
                                                $load['articleTags']
= function ($query) use ($params) {
                                                        $query-
>select(['slug', 'title']);
                                                };
                                        }

                                        if (in_array('categories',
$fields)) {

$load['articleCategories'] = function ($query) use ($params) {
                                                        $query-
>select(['id', 'slug', 'cover'])->with([
                                                                'loc
alization' => function ($localization) use ($params) {

        $localization->select('category_id', 'title',
'description');

        $localization->where('lang', $params['localization']);
                                                                },
                                                        ]);
                                                };
                                        }

                                        if (in_array('author',
$fields)) {
                                                $load['author'] =
function ($query) use ($params, $AccessDenied) {
                                                        if
($AccessDenied !== false) {

$query->select(['id', 'avatar', 'nicName', 'name', 'surname']);
                                                        } else {

$query->select(['id', 'avatar', 'nicName']);
                                                        }
                                                };
                                        }

                                        if (in_array('cover',
$fields)) {
                                                $load['cover'] =
function ($query) use ($params) {
                                                        $query-
>select(['uuid', 'width', 'height', 'type', 'size', 'path',
'file']);
                                                        $query-
>with('additionalFields');
                                                };
                                        }
                                        $load['localization'] =
function ($query) use ($params) {
                                                $query-
>where('lang', $params['localization']);
                                        };

                                        $attachments = [];
                                        $articlesRecommended =
$articlesRecommended->load($load);

                                        if (in_array('attachments',
$fields) || in_array('text', $fields)) {
                                                $attachments = new
AttachmentHelper($articlesRecommended);
                                        }

                                        $articlesRecommended-
>map(function ($article) use ($fields, $builderQuery, $attachments)
{
                                                if
(in_array('cover', $fields)) {
                                                        $article-
>cover->setAppends(['sizes'])->makeHidden(['additionalFields']);
                                                }
                                                if (!empty($article-
>articleCategories->toArray())) {
                                                        $article-
>articleCategories = $article->articleCategories->map(function
($category) {

return $category->setAppends(['description', 'title'])-
>makeHidden(['localization']);
                                                        });
                                                }

                                                if
(in_array('attachments', $fields) || in_array('text', $fields)) {
                                                        $article-
>setAttachmentsAttribute($attachments->getAttachments($article-
>id));
                                                }

                                                if (in_array('text',
$fields)) {
                                                        $article-
>setAttachmentsBlocksAttribute($attachments-
>setAttachmentsBlocks($article->text, $article->id));
                                                }
                                                return
$builderQuery->setFieldsModel($article);
                                        });

                                        return [
                                                'response' => 'oк',
                                                'articles' =>
$articlesRecommended,
                                        ];
                                } else {
                                        throw new ApiException(
                                                [
                                                        'article' =>
'Article not found',
                                                ],
                                                404
                                        );
                                }
                        });
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                ],
                                000
                        );
                }
        }

        public static function setSettings($params)
        {
                try {
                        $AccessDenied = self::checkAccessDenied();
                        if ($AccessDenied === false) {
                                throw new ApiException(
                                        [
                                                'method' => 'access
denied',
                                        ],
                                        403
                                );
                        }

                        try {
                                new ValidatorHelper($params->all(),
[
                                        'rules' => [
                                                'id' =>
['required'],
                                                'settings' =>
['required'],
                                        ],
                                        'alias' => [],
                                ]);
                        } catch (ValidatorException $e) {
                                throw new ApiException($e-
>getDecodedMessage(), 101);
                        }
                        $article = ArticleModel::where('id',
$params->id)->first();

                        if (!$article) {
                                throw new ApiException(
                                        [
                                                'article' =>
'Article not found',
                                        ],
                                        404
                                );
                        }

                        foreach ($params->settings as $key =>
$value) {
                                switch ($key) {
                                        case 'geo_block':
                                                $article->geo_block
= $value;
                                                break;
                                        case 'negative':
                                                $article->negative =
$value;
                                                break;
                                        case 'theme_day':
                                                if
(count($AccessDenied['IR']) > 0) {
                                                        $article-
>theme_day = $value;
                                                }
                                                break;
                                        case 'indexing':
                                                if
(count($AccessDenied['IR']) > 0) {
                                                        $article-
>indexing = $value;
                                                }
                                                break;
                                        default:
                                                break;
                                }
                        }

                        $article->save();
                        event(new ArticleEvent($article));
                        return response()->json([
                                'response' => 'ok',
                        ]);
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        echo 1;
                        return $e;
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                ],
                                000
                        );
                }
        }

        public static function share($params)
        {
                try {
                        $AccessDenied = self::checkAccessDenied();
                        if ($AccessDenied === false) {
                                throw new ApiException(
                                        [
                                                'method' => 'access
denied',
                                        ],
                                        403
                                );
                        }

                        try {
                                new ValidatorHelper($params->all(),
[
                                        'rules' => [
                                                'id' =>
['required'],
                                                'keysShare' =>
['required', 'array'],
                                        ],
                                        'alias' => [],
                                ]);
                        } catch (ValidatorException $e) {
                                throw new ApiException($e-
>getDecodedMessage(), 101);
                        }

                        $article = ArticleModel::where('id',
$params->id)->first();

                        if (!$article) {
                                throw new ApiException(
                                        [
                                                'article' =>
'Article not found',
                                        ],
                                        404
                                );
                        }

                        $validshare = ['vk-bst', 'vk-bstbash', 'vk-
yuldash', 'fb-bst', 'ok-bst'];
```

```
                        foreach ($params->keysShare as $value) {
                                if (!in_array($value, $validshare))
{
                                        throw new ApiException(
                                                [
                                                        'method' =>
'access denied',
                                                ],
                                                403
                                        );
                                }
                        }
                        foreach ($params->keysShare as $value) {
                                $share = new Share();
                                $share->key = $value;
                                $article->articleShare()-
>save($share);
                                event(new ArticleEvent($article));
                        }
                        return response()->json([
                                'response' => 'ok',
                                'share' => $share,
                        ]);
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        echo 1;
                        return $e;
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                ],
                                000
                        );
                }
        }

        public static function target($params)
        {
                try {
                        $AccessDenied = self::checkAccessDenied();
                        if ($AccessDenied === false) {
                                throw new ApiException(
                                        [
                                                'method' => 'access
denied',
                                        ],
                                        403
                                );
                        }
                        if (empty($params->keys)) {
                                $params->merge(['keys' => []]);
                        }
                        try {
                                new ValidatorHelper($params->all(),
[
                                        'rules' => [
                                                'id' =>
['required'],
                                                'keys' => ['array'],
                                        ],
                                        'alias' => [],
                                ]);
                        } catch (ValidatorException $e) {
                                throw new ApiException($e-
>getDecodedMessage(), 101);
                        }

                        $article = ArticleModel::where('id',
$params->id)->first();

                        if (!$article) {
                                throw new ApiException(
                                        [
                                                'article' =>
'Article not found',
                                        ],
                                        404
                                );
                        }

                        // return 1;

                        $valid = ['yandex-news', 'yandex-zen',
'yandex-turbo', 'sendpulse', 'amp'];
                        foreach ($params->keys as $value) {
                                if (!in_array($value, $valid)) {
                                        throw new ApiException(
                                                [
                                                        'method' =>
'access denied',
                                                ],
                                                403
                                        );
                                }
                        }
                        $article->articleTarget()->delete();
                        foreach ($params->keys as $value) {
                                $article->articleTarget()-
>updateOrCreate(['key' => $value], []);
                        }
                        event(new ArticleEvent($article));
                        return response()->json([
                                'response' => 'ok',
                        ]);
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        echo 1;
                        return $e;
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                ],
                                000
                        );
                }
        }

        public static function setRecommendation($params)
        {
                try {
                        $AccessDenied = self::checkAccessDenied();
                        if ($AccessDenied === false) {
                                throw new ApiException(
                                        [
                                                'method' => 'access
denied',
                                        ],
                                        403
                                );
                        }

                        try {
                                new ValidatorHelper($params->all(),
[
                                        'rules' => [
                                                'id' =>
['required'],
                                                'status' =>
['required', 'bool'],
                                        ],
                                        'alias' => [],
                                ]);
                        } catch (ValidatorException $e) {
                                throw new ApiException($e-
>getDecodedMessage(), 101);
                        }

                        $article = ArticleModel::where('id',
$params->id)->first();

                        if (!$article && $article->status !== (!
$params->status ? 10 : -100)) {
                                throw new ApiException(
                                        [
                                                'article' =>
'Article not found',
                                        ],
                                        404
                                );
                        }

                        // return [
                        //      "recommendationable_id" => $article-
>id,
                        //      "recommendationable_type" =>
get_class($article),
                        //      "status" => $params->status ===
false ? -100 : 10,
                        //      "do_scroll" => false,
                        //      "number" => null,
                        // ];

                        RecommendationModel::updateOrCreate(
                                [
                                        'recommendationable_id' =>
$article->id,
                                        'recommendationable_type' =>
get_class($article),
                                ],
                                [
                                        'status' => $params->status
=== false ? -100 : 10,
                                        'do_scroll' => false,
                                        'number' => null,
                                ]
                        );

                        // return get_class($article);

                        event(new ArticleEvent($article));
                        return response()->json([
                                'response' => 'ok',
                        ]);
                } catch (ApiException $e) {
                        throw $e;
                } catch (Exception $e) {
                        echo 1;
                        return $e;
                        throw new ApiException(
                                [
                                        'server' => 'ooops!',
                                ],
                                000
                        );
                }
        }

        public static function checkAccessDenied($validRole = null,
$validAbilities = null)
        {
                if (is_null($validRole)) {
                        $validRole = ['super-admin', 'chief-editor',
'middle-editor', 'editor'];
```

```
                }
                if (is_null($validAbilities)) {
                        $validAbilities = ['all-edit', 'news-edit'];
                }
                if (Auth::guard('api')->check()) {
                        $user = Auth::guard('api')->user();
                        $rolesUser = $user->getRoles()->toArray();
                        $abilitiesUser = $user
                                ->getAbilities()
                                ->map(function ($item) {
                                        return $item->name;
                                })
                                ->toArray();

                        $IR = array_intersect($validRole,
$rolesUser);
                        $IA = array_intersect($validAbilities,
$abilitiesUser);
                        $IR = array_values($IR);
                        $IA = array_values($IA);

                        if (count($IR) > 0) {
                                if (count($IA) > 0) {
                                        return [
                                                'IR' => $IR,
                                                'IA' => $IA,
                                        ];
                                }
                        }
                }
                return false;
        }
}
```