**FRIENDLY SOCIAL LTD.**
**Unit 4E Enterprise Court, Fairfield Park**
**Rotherham, South Yorkshire**
**England S63 5DB, United Kingdom**

# Technical documentation of E-GOV IT application

**Table of Contents**

## *The structure of directories and files in application*

The application was made based on the Wordpress platform version 4.3.1. To run the web application server is required with support for PHP at least version 5.4. The application requires a MySQL database (InnoDB engine). The main part of the application is created Wordpress template - eGov App (located in the /wp-content/themes/egovapp). An additional element of the application are plugins. They can be divided into internal - designed specifically for the needs of applications and external - shared by other users on the Open Source license (catalog /wp-content/plugins/).

To create a site there were used free (Open Source) libraries that provide ready-made components and facilities for some items.

- Bootstrap v 3.x - HTML, CSS, and JS framework - http://getbootstrap.com/
- jQuery and jQuery UI - javascript libraries
- jQuery UI Touch Punch 0.2.3
- DateTimePicker jQuery plugin v2.4.5
- SheepIt! JQuery Plugin v. 1.1.1 - http://www.mdelrosso.com/sheepit/ - library used to creating dynamic forms
- Chart.js v.1.0.2 - http://chartjs.org/ - Library helping viewing charts
- MarkerClustererPlus for Google Maps V3 - https://code.google.com/p/google-maps-utility-library-v3/

## *Installation and initial configuration*

Installation of site template is done in a standard way for Wordpress templates:

- Install Wordpress page - http://www.wordpress-polska.pl/o-wordpressie/wordpress-instalacja/.
- Log on to the given installation data
- Go to Appearance → Themes
- Turn on the theme eGov App
- Activate modules required for the operation of all application components:
- BuddyPress
- BuddyPress Activity Plus
- BuddyPress Activity Comment Notifier
- Egovapp Base Module - This module must be enabled before it activates the other Egovapp modules
- Egovapp Module Budgets
- Egovapp Module Discussion
- Egovapp Module Voting
- Radio Buttons for Taxonomies
- Wordpress Social Login
- Yet Another Stars Rating
- Create a new menu - Main Menu. Assign them to the position of Primary Menu. Add to menu Types of applications available in the system.
- Create a new menu - Footer Menu. Assign them to the position of Footer Menu. Add links that should be visible in the footer.
- Set up a layout template – Template panel

## *eGov App Template*

Page template was placed in the catalog /wp-content/themes/egovapp/. Description of its most important elements:

- **/bootstrap/** - contains elements of Bootstrap v.3.3.4 - framework HTML, CSS and JS helping by creation of responsive web (http://getbootstrap.com)
- **/css/** - css files template
- **/egovdivisions/** - elements responsible for the operation and maintenance of part of the administrative panel of a site related to administrative sections and sharing them with data from the site
- **/egovmaps/** - elements responsible for the display and operation of map
- **/egovnotification/** - elements related to service notifications in the application
- **/egovpost/** - elements containing a definition and support for egovpost - additional entry type (Wordpress Post Type) - prepared specially for the application. Created post is responsible for the operation and display applications added by users. It is the main entry type used on the site.
- **/fonts/** and /open-sans-fontface/ - catalogs with additional fonts
- **/images/** - pictures that are a part of the template
- **/js/** - javascript scripts
- **/menu/** - elements responsible for modifying site navigation
- **/panel/** - elements responsible for the display and the operation of the template settings
- **/sass/** - CSS preprocessor files (SASS) - facilitating the creation of css style sheets
- **/users/** - files modifying existing types of users and adding new types of user applications
- **functions.php** - standard Wordpress template file, contains the basic configuration template
- **style.css** - the main css file of Wordpress template, required for its proper operation
- **other template files** - files responsible for the display of individual elements of the site, made according to Wordpress standard.
    - archive-egovpost.php - is responsible for the display of archive of entries of egovpost type - thus a view of the list of entries added by users
    - archive.php - view of the list of standard entries. It can be used eg. to create a standard blog based on Wordpress
    - header.php, footer.php - accordingly the header and the footer of the site
    - home.php - view of the home page of the site
    - page.php - view of ordinary subpage - entries of Pages type (Wordpress Page Type)
    - page-budgets.php - view of page listing egovpost entries which are active module budgets
    - page-discussion.php - view of page listing egovpost entries that have active discussion module

- page-voting.php - view of page listing egovpost entries that have active voting module
- single.php - page template containing a single Wordpress entry
- single-egovpost.php - page template containing a single egovpost entry - view of the application added by the user
- taxonomy-egovtype.php - template responsible for displaying a list of applications (egovpost) according to the type of application (egovtype)

## *Template support elements*

## Egovpost

- Egovpost - for the purposes of the application has been created a new type of Wordpress entry (custom post type).

***wp-content\themes\egovapp\egovpost\egovpost.php***

```
$egovpost_args = array(
        'public' => true,
        'supports' => array( 'title', 'editor', 'comments', 'thumbnail',
'author', 'excerpt' ),
        'labels' => $egovpost_labels,
        'taxonomies' => array( 'category' ),
        'has_archive' => true,
        'query_var' => true,
        'map_meta_cap' => true,
        'capability_type' => array( 'egovpost', 'egovposts' )
);
register_post_type( 'egovpost', $egovpost_args );
```

Objects of entries can be added by creating a class instance eGov Post:

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
class Egovpost {}
```

Creation of a new notification shall be made using the method create() of Egovpost class:

***wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
function create( $title, $type, $category, $content, $author_id, $excerpt,
$status = 'pending' ){}
```

## Taxonomies

Egovpost is linked to a standard taxonomy in Wordpress - categories. In addition, there were created two taxonomies: Egovtype and Egovstate.

Egovtype - taxonomy for defining the type of application. By default, the system creates 4 types of problems, projects, consultations and decisions.

***\wp-content\themes\egovapp\egovpost\egovpost.php***

```
register_taxonomy( 'egovtype', 'egovpost', array(
        'label' => __('Rodzaj', 'egovapp'),
        'labels' => array(
            'name' => __('Rodzaj', 'egovapp'),
            'singular_name' => __('Rodzaj', 'egovapp'),
            'all_items' => __('Wszystkie', 'egovapp'),
```

```
            'edit_item' => __('Edytuj', 'egovapp'),
            'view_item' => __('Zobacz', 'egovapp'),
            'update_item' => __('Aktualizuj', 'egovapp'),
            'add new item' =>   ('Dodaj nowy', 'egovapp'),
            'new_item_name' => __('Nowa nazwa', 'egovapp'),
            'parent_item' => __('Rodzic', 'egovapp'),
            'parent_item_colon' => __('Rodzic:', 'egovapp')
        ),
        'rewrite' => array( 'slug' => 'typ' ),
        'hierarchical' => true,
        'capabilities' => array(
            'manage_terms' => 'manage_egovtypes',
            'edit_terms' => 'manage_egovtypes',
            'delete_terms' => 'manage_egovtypes',
            'assign_terms' => 'edit_egovposts',
        )
    ));
```

Egovstate - taxonomy created for the possibility of determining the current status of the application. By default, the system creates 3 statuses: Report unverified, Report verified, Published report.

***\wp-content\themes\egovapp\egovpost\egovpost.php***

```
register_taxonomy( 'egovstate', 'egovpost', array(
        'label' => __('Status', 'egovapp'),
        'labels' => array(
            'name' => __('Status', 'egovapp'),
            'singular_name' => __('Status', 'egovapp'),
            'all_items' => __('Wszystkie', 'egovapp'),
            'edit_item' => __('Edytuj', 'egovapp'),
            'view_item' => __('Zobacz', 'egovapp'),
            'update_item' => __('Aktualizuj', 'egovapp'),
            'add_new_item' => __('Dodaj nowy', 'egovapp'),
            'new_item_name' => __('Nowa nazwa', 'egovapp'),
            'parent_item' => __('Rodzic', 'egovapp'),
            'parent_item_colon' => __('Rodzic:', 'egovapp')
        ),
        'rewrite' => array( 'slug' => 'status' ),
        'hierarchical' => true,
        'capabilities' => array(
            'manage_terms' => 'manage_egovstates',
            'edit_terms' => 'manage_egovstates',
            'delete_terms' => 'manage_egovstates',
            'assign_terms' => 'manage_egovstates',
        )
    ));
```

## EgovpostManager

EgovpostManager – case management class. It has static methods that facilitate access to the applications added to the system.

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
class EgovpostManager {}
```

Exemplary methods of this class:

*Download a list of url to the files to download associated with the notification*

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
public static function getDownloads( $post_id ){
	return get_post_meta( $post_id, 'egovpost_downloads', true );
}
```

*Download a list of files associated with the notification*

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
public static function getLinks( $post_id ){
	return get_post_meta( $post_id, 'egovpost_links', true );
}
```

*Download the location data for the notification*

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
public static function getLocalization( $post_id ){
	return get_post_meta( $post_id, '_localization', true );
}
```

*Download all types (Egovtype) associated with the notification*

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
public static function getEgovpostType( $post_id ){
	$terms = wp_get_post_terms( $post_id, 'egovtype' );

	return $terms;
}
```

*Download all status (Egovstate) associated with the notification*

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
public static function getEgovpostState( $post_id ){
	$terms = wp_get_post_terms( $post_id, 'egovstate' );

	return $terms;
}
```

*Download all categories (Category) associated with the notification*

**\wp-content\themes\egovapp\egovpost\EgovpostManager.php**

```
public static function getEgovpostCategories( $post_id ){
	$terms = wp_get_post_terms( $post_id, 'category' );

	return $terms;
}
```

*Download a list of applications based on taxonomy (date id and the name of the taxonomy)*

**\wp-content\themes\egovapp\egovpost\EgovpostManager.php**

```
public static function getEgovpostsByTax( $term_id, $taxonomy ){}
```

*Download a  list of applications based on the author id's report*

**\wp-content\themes\egovapp\egovpost\EgovpostManager.php**

```
public static function getEgovpostsByAuthor( $author_id ){}
```

*Download a list of applications based on the date (in a format compatible with DateTime). By default, it downloads a list of applications on the basis of the current date.*

**wp-content\themes\egovapp\egovpost\EgovpostManager.php**

```
public static function getEgovpostsByDate( $date = '' ) {}
```

*Download a list of entries from a specific period of time*

**wp-content\themes\egovapp\egovpost\EgovpostManager.php**

```
public static function getEgovpostsByDateRange( $date_from = '', $date_to =
'' ) {}
```

*Download all published notifications*

**\wp-content\themes\egovapp\egovpost\EgovpostManager.php**

```
public static function getAllEgovPosts(){}
```

*Download all published notices complying with the additional requirements. Additional arguments should be given as an array of arguments compatible with WP_Query*

**\wp-content\themes\egovapp\egovpost\EgovpostManager.php**

```
public static function getSelectedEgovPosts( $args = array() ){
	$egov_posts_args_required =
	array(
		'posts_per_page' => -1,
		'post_status' => 'publish',
		'post_type' => 'egovpost'
	);
```

```
        $args = array_merge( $args, $egov_posts_args_required );

        $egov_posts = get_posts( $args );

        return $egov_posts;
}
```

*Checking on the specified module for the application*

***\wp-content\themes\egovapp\egovpost\EgovpostManager.php***

```
public static function checkModuleAvaliable( $egovpost_id, $module_name ){}
```

## Compass – open-source CSS Authoring Framework

When creating a page template using CSS framework - Compass (http://compass-style.org/), which uses Sass - CSS3 extension.

Usage (requires prior installation of Compass accordance with the instructions http://compass-style.org/install/):

- In the console, navigate to the directory Wordpress template *(\wp-content\themes\egovapp\)*
- run the command
  ```
  compass watch
  ```
- it will start a process of monitoring changes in sass files - catalog \wp-content\themes\egovapp\sass\ (as Compass configuration located in \wp-content\themes\egovapp\config.rb)

There were used the following facilities of Compass and Sass framework in the egovapp template:

- CSS code compression – changes in SASS files are stored in compressed resulting CSS files (catalog \wp-content\themes\egovapp\css\)
- pictures from the catalog \wp-content\themes\egovapp\images\theme\ are automatically combined into a single image (sprite) located in the parent directory (\wp-content\themes\egovapp\images\)
- there were used variables (\wp-content\themes\egovapp\sass_variables.scss) in Sass files to describe some of the frequently recurring elements layout site (fonts, colors, backgrounds, shadows).

You can modify and use the other facilities framework Compass and Sass.

## *Plugins*

## Internal plugins

There were prepared 3 main modules: Budgets, Discussion and Vote, for the purposes of the application. Each of them is an extension of the basic module (Egovapp Base Module).

## *Egovapp Base Module*

- \wp-content\plugins\egovapp-module-base\ - Egovapp Base Module - base module. It contains the basic methods of integrating with the template page. It makes it easier to create additional modules. Besides methods of connecting the module to the site, it also includes ready-made templates for the view of the module on the home page and the notification.
- \wp-content\plugins\egovapp-module-budgets - Egovapp Module Budgets - a module that allows to define budgets for each notification.

Elements of the module:

- \css\ - CSS files modifying the views of budgets module:
  - budgets_admin.css - oscillation for the administrative panel visible in the view report
  - budgets_home.css - style for the view of the home page
  - budgets_voting_panel.css - style for the administrative panel of the vote on budgets
  - style.css - the main file of module styles
- \ js \ - javascript files associated with the plugin
  - budgets.js - basic scripts file, responsible for the view and the operation of the module from the user's site.
  - budgets_home.js - scripts responsible for the display and operation of the module on the home page of your site.
  - budgets_admin.js - scripts responsible for the operation of components of the module from the administrator. It is responsible for creating and editing budgets for the notification.
  - budgets_voting_panel.js - allows you to add scripts to a panel vote on budgets (admin)
- \templates\ - view module files, override the views defined in the base module

## *Egovapp Module Budgets*

- egovapp-module-budgets.php – Egovapp Module Budgets – the main plug-in class, expanding the base module.

It contains methods for adding new panels for the administrative panel notification - a panel defining the budget and the panel of budget proposals for the notification.

```
function register_egovpost_budgets_panel() {}
function register_egovpost_budgets_voting_panel() {}
```

It also adds a new bookmark to the administrative panel of site - Budgets:

```
function egov_register_budgets_voting_page(){}
```

- \Includes\ - additional classes used by the module
- EgovBudget.php - EgovBudget class - used for reading and writing data of a single budget for the notification.

```
$budget = new EgovBudget( $egovpost_id, $user_id );
```

*Downloading budget data:*

```
$budget->getBudget();
```

*Recording budget settings:*

```
$budget->setBudget( $budget_args );
$budget->saveBuget();
```

- EgovBudgetManager.php – EgovBudgetManager class – used to download information about the budgets for the notification (using static methods):

*Download budget data based on notification and user*

```
public static function getBudgetByPostAndUser( $egovpost_id, $user_id ){}
```

*Download information about the budgets assigned to the notification*

```
public static function getBudgetsByPost( $egovpost_id ){}
```

*Download information about budgets defined by the user*

```
public static function getBudgetsByUser( $user_id ){}
```

The structure of the database table for budget notifications (*_egovpost_egovappmodulebudgets* ):

| Column | Type | Attributes | Null | Default | Description |
|---|---|---|---|---|---|
| id | int(10) | UNSIGNED | No | Lack | AUTO_INCREMENT |
| egovpost_id | int(10) | UNSIGNED | No | Lack | Notification id |
| user_id | int(10) | UNSIGNED | No | Lack | Budget author id |
| budget | longtext | | Yes | NULL | Details of the budget (Table subjected to serialization) |
| last_change | datetime | | No | Lack | Last updated |

## *Egovapp Module Discussion*

- \wp-content\plugins\egovapp-module-discussion\ – Egovapp Module Discussion – module adding the possibility to discuss about the application. It requires Buddypress plugin (version 2.4.2).

Elements of the module:

- \css\ - css files associated with a module
- \js\ - javascript files associated with the module.
  - discussion.js - contains the functions responsible for the operation of the module elements discussed at the user site (navigation panel for comments, quoting statements by other users, marking, reporting and sharing comments, and vote on the speech of others.
- \templates\ - view module files, override the views defined in the base module
- \activity\ - templates override templates comments and a list of comments provided by BuddyPress
  - comment.php - template overrides the comments thread view (activity), provided by Buddypress
  - entry.php - template overrides the main entry thread view, provided by Buddypress
  - comments.php - comments view template - is used to override the standard comments view provided by Wordpress
  - egovapp-module-discussion.php - EgovappModuleDiscussion - the main class of plug-in, expanding the base mode. Adds support templates for comments and discussion panel settings for users.
- \includes\ - additional classes used by the module
  - EgovDiscussionBookmark.pl - Class EgovDiscussionBookmark – allow to mark the selected comments by users. Class methods use the recording function (bp_activity_update_meta) and read (bp_activity_get_meta) meta data provided by Buddypress.

*Reading data for the selected comment*

```
public function getBookmark(){}
```

*Recording data for the selected comment (marking the comment)*

```
public function setBookmark(){}
```

  - EgovDiscussionExpertComment.php – a class used for marking the comment as an entry expert. Class methods use the recording function (bp_activity_update_meta) and read (bp_activity_get_meta) meta data provided by Buddypress.

*Checking whether the selected comment is marked as an expert*

```
public function getExpertStatus(){}
```

*Marking the comment as an expert*

```
public function setExpertStatus(){}
```

*Checking if the given user (currently logged if $user_id = null) is an expert.*

```
public static function checkUserIsExpert( $user_id = null ){}
```

  - EgovDiscussionSummary.php – EgovDiscussionSummary class – used to download data summary discussion for the given notification.

- EgovDiscussionVote.php – EgovDiscussionVote class – saves the user voices for comment

*Sets a new value for the user's voice comment*
```
function create( $vote, $comment_id, $user_id ){}
```

*Writes the set value*
```
function addNew(){}
```

- EgovDiscussionVoteManager.php –EgovDiscussionVoteManager class – contains static methods receiving information about the votes associated with the selected comment

*Download the sum of the votes for the comment*
```
public static function getCommentVotes( $comment_id ){}
```

*Ensures that the sum of votes has not reached a value indicative of a comment about the low value*
public static function checkCommentWeak( $votes_sum ){}

*Checking whether a vote on the selected comment is the voice of a devoted by the currently logged in user.*
```
public static function checkIsMyVote( $comment_id, $vote_value ){}
```

The structure of the database table for the vote on comments *(_egovappmodulediscussion_vote)*:

| Column | Type | Attributes | Null | Default | Description |
|---|---|---|---|---|---|
| id | int(10) | UNSIGNED | No | Lack | AUTO_INCREMENT |
| comment_id | int(10) | UNSIGNED | No | Lack | Comment id |
| vote | int(11) | | No | Lack | The value of the voice (0, 1 or -1) |
| user_id | int(10) | UNSIGNED | No | Lack | User voting id |

## *Egovapp Module Voting*

- \wp-content\plugins\egovapp-module-voting\ - plugin adding voting module over notifications in the system

Elements of the module:

- \css\ - CSS files associated with the module
  - style.css – file of basic styles
  - delegation_panel.css - file of styles describing the votes delegation panel
- \js\ - javascript files associated with the module
  - egovapp_delegation_panel.js - scripts associated with the votes delegation panel (admin page)
  - vote.js - scripts responsible for the display and operation of the vote on the report from the user of the site (frontend)
  - vote_admin.js - scripts associated with the module - from the admin (backend)
- \charts\ - jQuery library files - Chart (http://chartjs.org/) - used to display graphs for voting on the notification
- \datetimepicker\ - jQuery library files – DateTimePicker (https://github.com/xdan/datetimepicker) - used to display the panel to choose the date based on the date and time.
- \templates\ - view module files, override the views defined in the base module
  - egovapp-module-voting.php - EgovappModuleVoting - the main plug-in class, expanding the base module. In addition, it adds database tables that store data on the vote declarations. It also adds settings panel vote for the statement:

```
function egovpost_voting_panel( $post ){}
```

and the votes delegation panels:

*Panel of posting users to vote on selected categories*

```
function egovapp_voting_delegation_panel(){}
```

*Panel of votes casted by delegated users*

```
function egovapp_voting_delegation_panel_votes(){}
```

*Panel displaying the logged-on user's rights to vote on behalf of someone else*

```
function egovapp_voting_delegation_panel_permissions(){}
```

EgovappModuleVoting class also includes a method of giving Ajax support vote for the statement:

```
function addNewVote(){}
```

- \includes\ - additional classes used by the module
  - EgovVote.php - EgovVote class - is responsible for the addition or modification of the voting on the application for the user. Voices (vote_id) take values: 0 – No, 1 - Yes, 2 - Abstain, 3 - Veto. In the list of users (users_id's) associated with the voice, there are visible only users explicitly voting

The structure of the database table, used to store information about the votes casted on the application by the users (*_egovappmodulevoting_vote*)
.

| Column | Type | Attributes | Null | Default | Description |
|---|---|---|---|---|---|
| id | int(10) | UNSIGNED | No | Lack | AUTO_INCREMENT |
| egovpost_id | int(10) | UNSIGNED | No | Lack | Notification Id |
| users_ids | longtext | | Yes | NULL | Users id table subjected to serialization |
| vote_id | int(11) | | No | Lack | Vote id (0, 1, 2 or 3) |
| count | int(11) | | Yes | NULL | Number of votes |
| last_change | datetime | | No | Lack | Last update of row |

- EgovVoteData.php - EgovVoteData class – is used to record information about all the users who voted for data entry. List of users (*users_ids*) contains explicit and anonymous users, who gave voice to any notification.

The structure of a database table (*_egovappmodulevoting_vote_post_users*):

| Column | Type | Attributes | Null | Default | Description |
|---|---|---|---|---|---|
| id | int(10) | UNSIGNED | No | Lack | AUTO_INCREMENT |
| egovpost_id | int(10) | UNSIGNED | No | Lack | Notification id |
| users_ids | longtext | | Yes | NULL | Users id table subjected to serialization |
| last_change | datetime | | No | Lack | Last update of row |

- EgovVoteDelegation.php – EgovVoteDelegation class – management of delegated votes. Saves and downloads voting data by members delegated to vote.

The structure of a database table ( *_egovappmodulevoting_delegation*):

| Column | Type | Attributes | Null | Default | Description |
|---|---|---|---|---|---|
| id | int(10) | UNSIGNED | No | Lack | AUTO_INCREMENT |
| delegated_id | int(10) | UNSIGNED | No | Lack | Delegated member’s id |
| users_ids | longtext | | Yes | NULL | User id table on whose behalf voted delegated, subjected to serialization |
| egovpost_id | int(11) | | No | Lack | Notification id |
| vote_id | int(11) | | No | Lack | Vote id (0, 1, 2 or 3) |
| last_change | datetime | | No | Lack | Last update of row |

- EgovVoteDelegationCategory.php – EgovVoteDelegationCategory class – management of users delegated and delegating within the category of applications.

The structure of a database table ( *_egovappmodulevoting_delegation_category*):

| Column | Type | Attributes | Null | Default | Description |
|---|---|---|---|---|---|
| id | int(10) | UNSIGNED | No | Lack | AUTO_INCREMENT |
| category_id | int(10) | UNSIGNED | No | Lack | Category id for which delegating allows delegated vote on their behalf |
| delegating_id | int(10) | UNSIGNED | No | Lack | Delegating member's id |
| user_id | int(10) | UNSIGNED | Yes | NULL | Delegated member's id |
| last_change | datetime | | No | Lack | Last update of row |

- EgovVoteManager.php – EgovVoteManager class – management of voting data on the report. It downloads information about the votes cast and voting on the basis of the notification and the user.

Exemplary methods of this class:

*Download voting data (vote type and the number of votes of each type) for the selected application:*

```
public static function getVotesByPostID( $post_id ){}
```

*Download a vote casted by the user to the selected notification*

```
public static function getVotesByPostAndUserID( $post_id, $user_id ){}
```

*Download a list of users who have given vote to any selected notification*

```
public static function getPostVotingUsers( $post_id ){}
```

*Check that the vote on the selected notification is active*

```
public static function checkVotingStatus( $egovpost_id ){}
```

*Check that the specified user (if the user isn't specified - checking the currently logged in) can vote for the chosen notification*

```
public static function checkUserCanVote( $post_id, $user_id = null ){}
```

*Check if the user delegated to vote, gave vote to the selected notification*

```
public static function checkUserEgovpostHasDelegation( $user_id,
$egovpost_id ){}
```

*Download a list of users on whose behalf, was the vote casted to report by the delegated user.*

```
public static function getUserDelegatedVotes( $user_id, $post_id ){}
```

*Download delegated voting data based on the user delegated and delegating*

```
public static function getDelegationsByUser( $delegated_id, $user_id ){}
```

*Download delegated voting data based on the user delegated and delegating and also notifications*

```
public static function getDelegationsByUserAndPost( $delegated_id, $user_id, $egovpost_id ){}
```

## *Example module*

- \wp-content\plugins\egovapp-module-example\ - an example of the module formed by the base module By default, unused in the system. It contains the basic elements required to create a working module, in accordance with the template site

## External plugins

### *BuddyPress*

The plugin helping by creation of social networking site, adds features such as activity streams - https://buddypress.org/. It is necessary for the proper operation of the module discussion.

### *BuddyPress Activity Plus*

The plugin adds the possibility of posting photos, videos or links to the stream of activity - https://pl.wordpress.org/plugins/buddypress-activity-plus/.

### *BuddyPress Activity Comment Notifier*

The plugin expanding the notification system for BuddyPress activity on notifications for comments - https://wordpress.org/plugins/bp-activity-comment-notifier/.

### *Radio Buttons for Taxonomies*

The plugin that allows the inclusion of limitations for selecting one item for any taxonomies (eg. category, status, or type) - https://wordpress.org/plugins/radio-buttons-for-taxonomies/. Required for proper application - the system automatically switches to the status of limit (egovstate) and type (egovtype) applications.

### *WordPress Social Login*

The plugin adding the possibility to log on to the website through popular social media: Facebook, Google and Twitter - https://wordpress.org/plugins/wordpress-social-login/. Required if the administrator wants to allow log in using a supported social media.

### *Loco Translate*

The plugin helping by translation of the site into another language - https://wordpress.org/plugins/loco-translate/. This plugin is not required.

### *Yet Another Stars Rating*

The plugin that allows the evaluation of the application - https://wordpress.org/plugins/yet-another-stars-rating/.
This plugin is not required.

- Installation of a clean version.

## *Additional*

## Generate API documentation (with an example of phpDocumentor 2.8.5)

```
phpdoc --template="responsive" -d /wp-content/themes/egovapp -d /wp-content/plugins/egovapp-module-base -d /wp-content/plugins/egovapp-module-budgets -d /wp-content/plugins/egovapp-module-discussion -d /wp-content/plugins/egovapp-module-voting -t /DOCUMENTATION_PATH/egovapp
```

## Creating a module based on a base plugin

In the system, you can easily create your own modules, based on the internal base module (Egovapp Module Base):

- The basic structure of a new module:

/new-module-folder
- /css/style.css - css file associated with the module
- /templates/ - module templates directory allows to overwrite the templates defined in /egovapp-module-base/templates/
- home_view.php - the template module file for the home page. Overrides template /egovapp-module-base/templates/home_view.php
- single_view.php - the template module file for a single page. Overrides template /egovapp-module-base/templates/single_view.php
- egovapp-module-main-file.php (example file name) - the main module file
- uninstall.php - a file containing the actions called during the uninstall of module

- Requirements of created module, for easy integration of module and template page:

The module should be created in the form of a class that extends the EgovappModule class and implementing the EgovappModuleShow interface:

***egovapp-module-main-file.php (przykładowa nazwa pliku)***

```
class EgovappModuleExample extends EgovappModule implements EgovappModuleShow {
	public $modulelabel = 'Module Example Label';

	public function __construct() {
		parent::__construct();
	}
}
```

Exemplary methods of the EgovappModule class, allowing integration of created modules with the template page:

Modifying the appearance of the module displayed on the main page is done by the development of methods:

```
public function renderHomeView() {}
```

Modifying the appearance of the module displayed on a single page takes place through the development of methods:

```
public function renderSingleView() {}
```

Modifying the label for the module settings panel in the administrative panel.

```
public function renderOptionsTabView(){}
```

Modification of the module settings panel in the administrative panel.

```
public function renderOptionsPanelView(){}
```

Loading scripts and styles associated with the created module

```
public function enqueue_scripts() {}
```

Execution of operations during activation of plugin by the site administrator. It allows you to initialize the settings required to operate the module. By default, saves the settings of the module required for the proper integration of the template page, adds the created module to the list of modules available for notifications. Additionally, it creates a new table in the database based on the class name used to create a module that can be used to record the relationship between the notification and the user.

```
public function activate(){}
```

Execution of operations during deactivation (not remove!) of the module by the administrator. By default, it removes the module from the list of available modules in the template and notifications.

```
public function deactivate(){}
```

Defining text domain for the module. It is used to determine an individual text domain, to facilitate the creation of translation of the plugin.

```
function module_textdomain() {}
```

It downloads basic settings of the plugin, defined during its activation (label, name, prefix).

```
public function getModuleSettings(){}
```

It allows you to update the default settings of the plugin.

```
protected function updateModuleSettings( $settings ){}
```

Then, in the main module file, you should create a new object of the created class and register catches shares (hooks) for activation and deactivation of the module:

***egovapp-module-main-file.php (przykładowa nazwa pliku)***

```
if( class_exists('EgovappModuleExample') ){
    $module_egovapp_ex = new EgovappModuleExample();

    register_activation_hook( __FILE__, array( &$module_egovapp_ex,
'activate' ));
```

```
    register_deactivation_hook( __FILE__, array( &$module_egovapp_ex,
'deactivate' ));
}
```

EgovappModuleShow interface ensures proper integration of the module created with CMS Wordpress and template site (required method of activation and deactivation of the plugin).

## Hooks (catches shares)

List of custom catches shares (hooks) added to the application:

- Page template
  - ***egovapp_module_home_view*** – homepage, allows you to connect the module to the homepage.

```
do_action( 'egovapp_module_home_view', $settings_home );
```

The function used with *egovapp_module_home_view* hook has one argument - an array appearance settings of homepage. Hook used by the base module to "connect" rendering function view of the module on the homepage.

```
add_action( 'egovapp_module_home_view', array( &$this, 'renderHomeView' )
);
```

  - ***egovapp_module_single_view*** – view of a single notification, allows you to connect the created module to the notification.

```
do_action( 'egovapp_module_single_view' );
```

Used by the base module to "connect" function rendering view of the module on a single notification

```
add_action( 'egovapp_module_single_view', array( &$this, 'renderSingleView'
) )
```

  - ***egovapp_module_options_tab_view*** **–** the administrative panel for the template. A hook allows you to add a new label of the module settings panel.

```
do_action( 'egovapp_module_options_tab_view' );
```

Used by the base module to create a new card label for the module in the panel template settings.

```
add_action(     'egovapp_module_options_tab_view',     array(     &$this,
'renderOptionsTabView' ) );
```

  - ***egovapp_module_options_panel_view*** **–** administrative panel for the template. A hook allows you to add a new panel settings for the module.

```
do_action( 'egovapp_module_options_panel_view' );
```

Used by the base module to create a new card for the module in the panel template settings.

```
add_action(     'egovapp_module_options_panel_view',     array(     &$this,
'renderOptionsPanelView' ) );
```

- ***egovapp_panel_mailing_form*** - a hook allowing the addition of new settings card in the template settings panel before notification settings card (before the last card in the panel).

```
do_action( 'egovapp_panel_mailing_form' );
```

- egovapp_notify_add_new_egovpost – a hook allowing you to send a notification when you add a new application to the system. The function performed with a hook takes one argument - an array of information: id of added notification and id of user adding notification.

```
do_action( 'egovapp_notify_add_new_egovpost', array( 'post_id' => $egov_post_id, 'user_id' => get_current_user_id() ) )
```

Used by EgovNotifyManager class to call the function sending notification

```
add_action( 'egovapp_notify_add_new_egovpost', array( &$this, 'notifyEgovpostNew' ), 30 );
```

## API for administrative units

The site provides an API helping by downloading of applications. Access of units for notifications is possible only after having added to the list of authorized entities. The files in the \wp-content\themes\egovapp\egovdivisions\ are responsible for the API.

EgovDivision.php - EgovDivision class - is used to create and save settings for administrative units which are supposed to have access to the API. It uses the *_divisions* table for storage of data.

The structure of the table of administrative units (*_divisions*)

| Column | Type | Attributes | Null | Default | Description |
|---|---|---|---|---|---|
| id | int(10) | UNSIGNED | No | Lack | AUTO_INCREMENT |
| name | char(60) | | No | Lack | Administrative unit name |
| authcode | char(50) | | No | Lack | Authorisation code |
| email | char(50) | | Yes | NULL | Unit email |
| share_methods | char(100) | | Yes | NULL | Methods of data sharing (json) - ["json","email"] |
| types | text | | Yes | NULL | Types of notifications (json) |
| categories | text | | Yes | NULL | Categories of notifications (json) |
| api_url | char(150) | | Yes | NULL | API url adress for the unit |
| division_date | datetime | | No | Lack | Update date |

EgovDivisionApi.php – EgovDivisionJsonApi class - responsible for converting data from the system to json.

EgovDivisionManager.php - EgovDivisionManager class - manages units that have access to the API.

*Creating a table in the _ divisions database, launched during activation of page template*

```
function create_division_table(){}
```

Download all the divisions from the system

```
public static function getAllDivisions(){}
```

Currently in the system are prepared two methods of sharing data: email and json. The units that have an active method of email can receive email notifications for submissions (in the form of a link to the notification). Json method allows you to receive data about notifications in json format.

Example of receiving json data on the website:

```
<script type="text/javascript">
      (function($){
            $(document).ready(function(){

                  $.ajax({
                        dataType: "jsonp",
                        url:
'http://ADRES_WITRYNY_EGOVAPP/?appdata=json&token=TOKEN'
                  }).done(function(data){

      // DISPLAY (PROCESSING) OF DATA WITH CONTENT OF SUBMISSION TO WHICH
THE UNIT HAS GIVEN ACCESS (date)

                  });
            });
      })(jQuery);
</script>
```