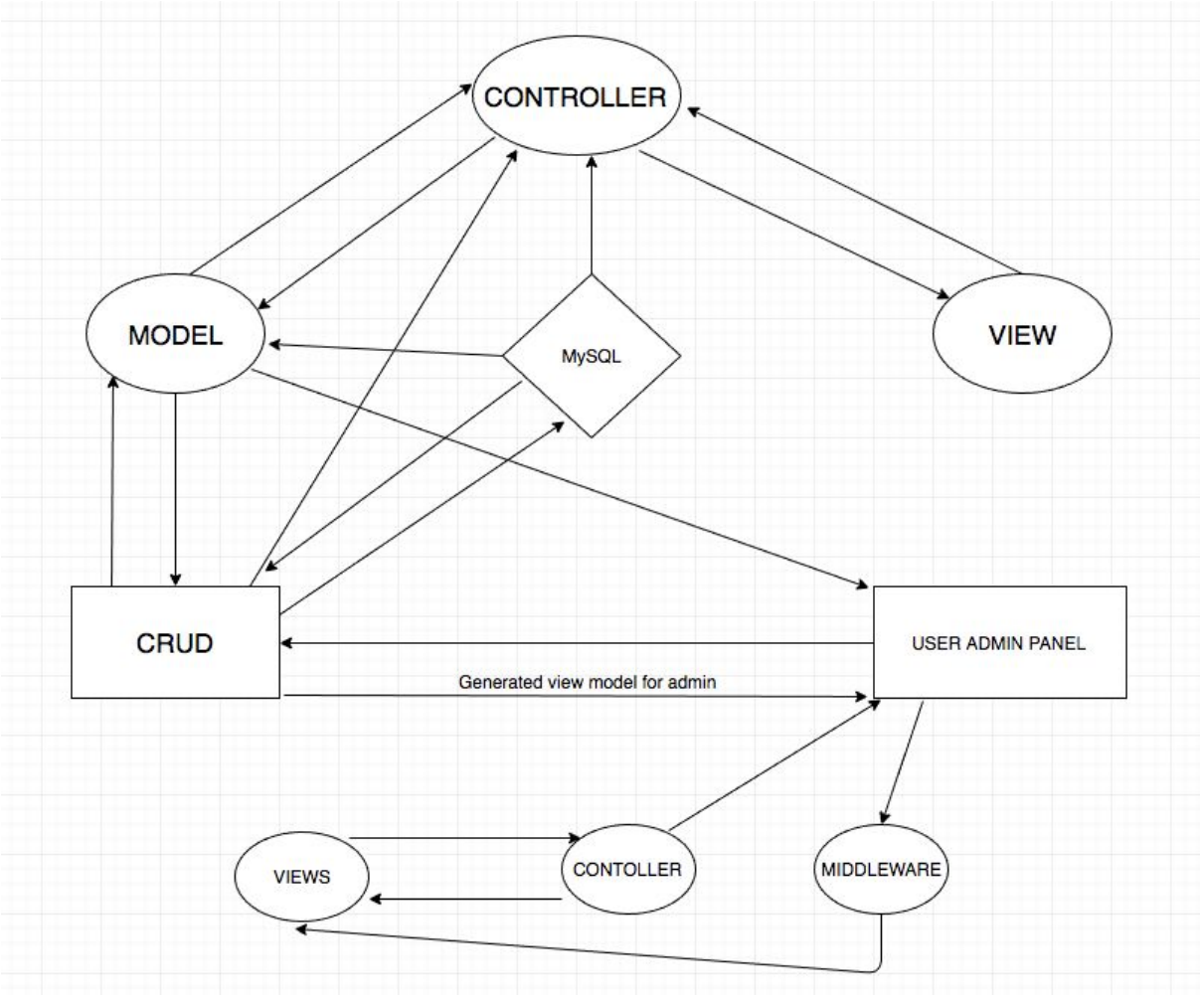
Technical Overview

- Disadvantages of usual frameworks
    1) In the usual frameworks there are base models which are include MVC model and database model.
    2) The main issue in real frameworks is that if we want to do something else which is not in the functionality domain of those frameworks, then we have to change the work principles of these frameworks and in most cases if it is not difficult then it is impossible.
    3) Usual frameworks do not include prepared user administration panel for websites and you can not extend it or change base work principles.

Our aim is to build framework that will allow us at least to change the base work principles and include user administration panel which is very useful tool for users.
In our framework we extend it with user administration panel, CRUD model, we used smarty technology which is very useful and convenient to connect html pages with php code parts, utilities, messages, forms and etc. The picture below weakly depicts our framework that will be described well further.



- First we will describe our framework main folders.
    - Controllers - includes controller classes
    - Core - includes CRUD model classes(all database and view generations functions) and Smarty php classes.
    - Forms - includes form classes, for example text forms, search forms, checkbox forms, radio box forms, login forms, registration forms, user profile forms, lost password forms, feedback forms and ect.
    - Imagemanager - this is prepared php tools which allow to us cut image sizes just by giving appropriate url for it, for instance "/hotel-reservation/imageresizer/resize/50/70//Applications/XAMPP/xamppfiles/htdocs/hotel-reservation/public/gallery/7.jpg" - which cuts given image 50px-70px.
    - Libs - includes php tools functionality such as "kcaptcha", "phpmailer", "phpthumb".
    - Messages - includes text files which includes all languages that included to the view part of our website, i.e. for multi language(such as english, italian), we can add as many languages as we want.
    - Middleware - include Middleware class which allow to us to get access to all useful data accessible in all pages of view part of our website.
    - Models - includes all model classes.
    - Modules - includes local MVC(views, controllers, extended models) model for user administrator part of our website.
    - Public - includes all files(images, doc files, pdf files and etc.) which is accessible both in user administration part and user view part in our website.
    - Static - includes all css, javascript, jquery, images(icons, png, jpeg) for user website part.
    - Templates_c - this is an extended smarty folder which keeps smarty generated templates(in its turn which is increases the work speed of our website because it keeps prepared html and php files).
    - Views - includes all pages of our user website part and the extension of these files is smarty template.

- Description of index.php file.
    - Includes global settings for our website
        - The settings for connecting to the database.
        - Application url - which is global url it is very useful in the case when we want to change our application web server.
        - Admin name - the user user administrator panel url name.
        - Includes controllers path roots.
        - Includes models path roots.
        - Includes how many and which languages we will have in user view part and user administrator panel side.
        - Predefined image extension settings.
        - Allowed extensions for accessing which type of files in future we will able to upload to our public folder.
        - Models setting which means which models will be visible in user administration panel side.
        - Debug settings.
        - We also can add new settings and get access to these settings in all php files because they are global.

```php
$__app = Array(
        'url' => $appUrl,
        'debug' => false,
        'url_converter_enabled' => false,
        'use_session' => true,
        'controllers_folder' => app_root . ds . 'controllers',
        'controllers_ext' => '.controller.php',
        'middleware_folder' => app_root . ds . 'middleware',
        'middleware_ext' => '.middleware.php',
        'models_folder' => app_root . ds . 'models',
        'templates_folder' => app_root . ds . 'views',
        'static_url' => $appUrl . '/static/',
        'static_files_ext' => Array('css', 'js'),
        'image_extensions' => Array('jpg', 'png', 'bmp'),
        'static_files' => Array(
            'jquery.js',
            'json.js',
            'template.js',
            'actions.js',
        ),
        'static_folder' => app_root . ds . 'static',
        'libs_folder' => app_root . ds . 'libs',
        'middleware_list' => Array(
            Array('Middleware','initializeApp'),
            Array('Middleware','forSmarty'),
            Array('Middleware','getSliderItems'),
            Array('Middleware','getLangsUrl'),
            Array('Middleware','getMenu'),
            Array('Middleware','getHoroscopeForMonth'),
            Array('Middleware','callTo'),
            Array('Middleware','getIbanners'),
        ),
        'smtp' => Array(
            'host' => 'mail.mega.az',
            'user' => 'notreply@mega.az',
            'password' => 'ylperton'
        ),
        'lpw' => Array(
            'from' => 'notreply@mega.az',
            'fromName' => 'agclub.az',
        ),
        'public_folder' => app_root . ds . 'public',
        'public_url' => $appUrl . '/public',
        'languages' => Array(
            'en' => 'English',
            'it' => 'Italian'
        ),
        'language_file_folder' => app_root . ds . 'messages',
        'page_count' => 10,
        'default_language' => 'en',
        'autoload_folders' => Array(
```

```php
        ),
        'name' => $_adminName,
        'middleware_list' => Array(
            Array('AdminMiddleware','initializeAdmin'),
            Array('AdminMiddleware','checkLoggedIn'),
            Array('AdminMiddleware','authorized'),
        ),
        'middleware_folder' => app_root . ds . 'modules' . ds . $_adminName . ds . 'middleware',
        'controllers_folder' => app_root . ds . 'modules' . ds . $_adminName . ds . 'controllers',
        'models' => Array(
            'FileManager',
            'MenuRelationsModel',
            'AdminUsersModel',
            'AdminUsersGroupModel',
            'SiteUsersModel',
            'RoomListViewModel',
            'GalleryModel',
            'NewsModel',
            'FeedbackModel',
            'BookingModel',
            //'NewsCommentsModel',
            //'ObjectsRelationsModel',
            //'CategoryRelationsModel',
            //'CallToModel',
            //'IBannerCategoriesModel',
            //'IBannerItemsModel',
            //'ServicesNetworkModel',
        ),
        'tree_models' => Array(
            1 => Array(
                'MenuRelationsModel',
                'MenuModel',
                '10'
            )
        ),
    )
);

$__template = Array(
        'debug' => $__app['debug'],
);

require core_path . ds . 'main.php';
```

- URL patterns
    - To avoid from recursion in order to find controllers(which takes a lot of time), their functions, to avoid from unsecure actions from users we developed url patterns which very useful and convenient method.
        - Url patterns in our model expresses with regular expression.
        - Url patterns include accessible and inaccessible parts in urls.
        - Url patterns include lang_id(if it is necessary), url name which will be unique, another additional fields if it is necessary, page id if the current page include page lists, controller name, the function name of the current controller name and the controller folder name(to avoid recursion in order to find that controller).

        - From this picture we can see the examples of url patterns.
        - For example the Room List View page as an url defined as room-list\/view/room_id/Controller name/ Controller function name/Controller path/.
        - Note that the invalid urls will redirect to the incorrect page(404) which will secure our website from undesirable actions.

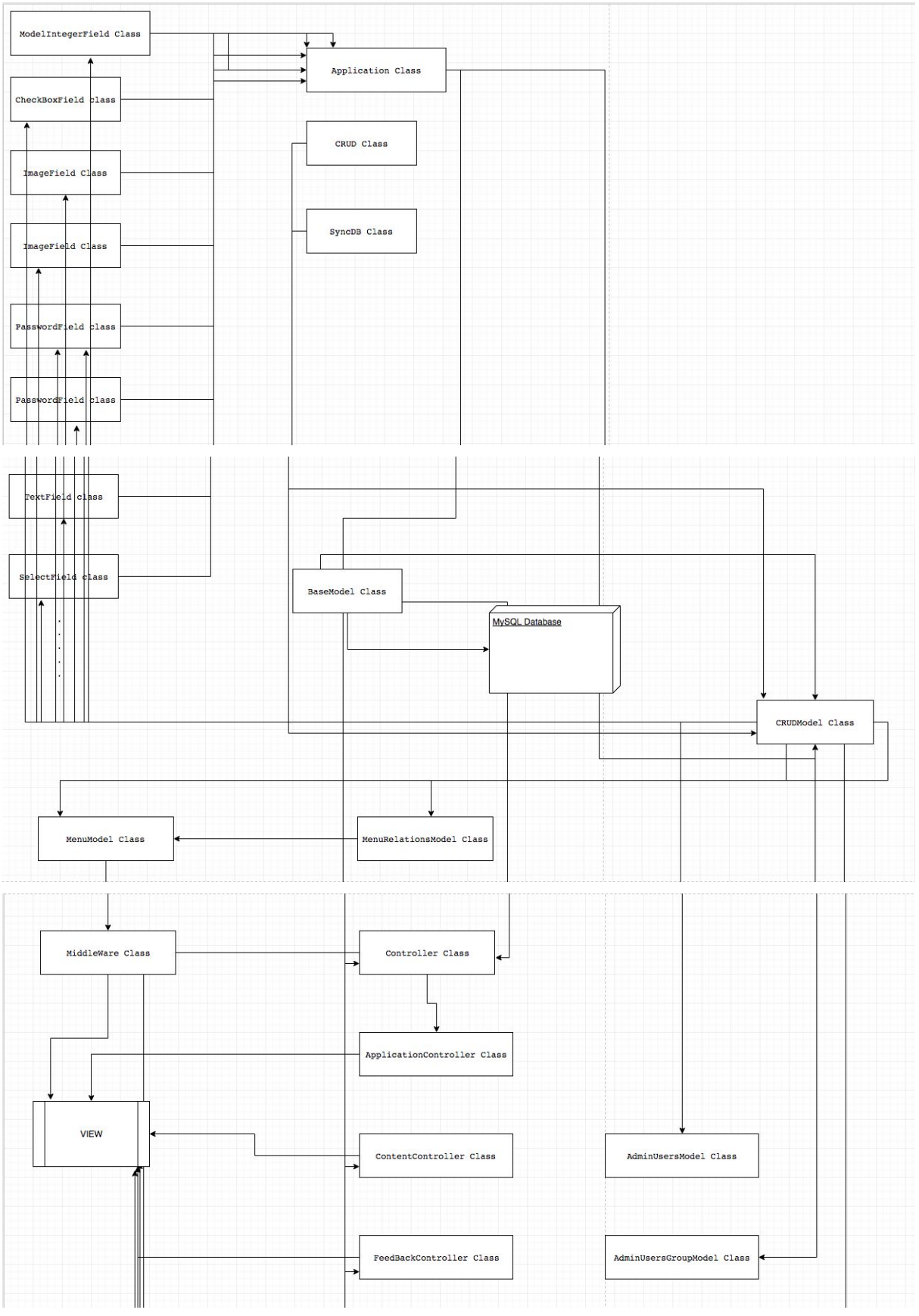
- How to create the new modules?.
    - In order to create content page in our website we just need to add new page in menu model in administration panel and fill content with some text. The content page does not include functionality, these type pages they are static, the only thing that changes in this type pages is their content.
    - In order to create module(functionality page) we have to pass seven step
        - First step is to create new folder in controller folder, create new folder with the same name in models folder, create new folder with the same name in views folder.
        - Second step is to create new controller class with some name in created new controller folder.

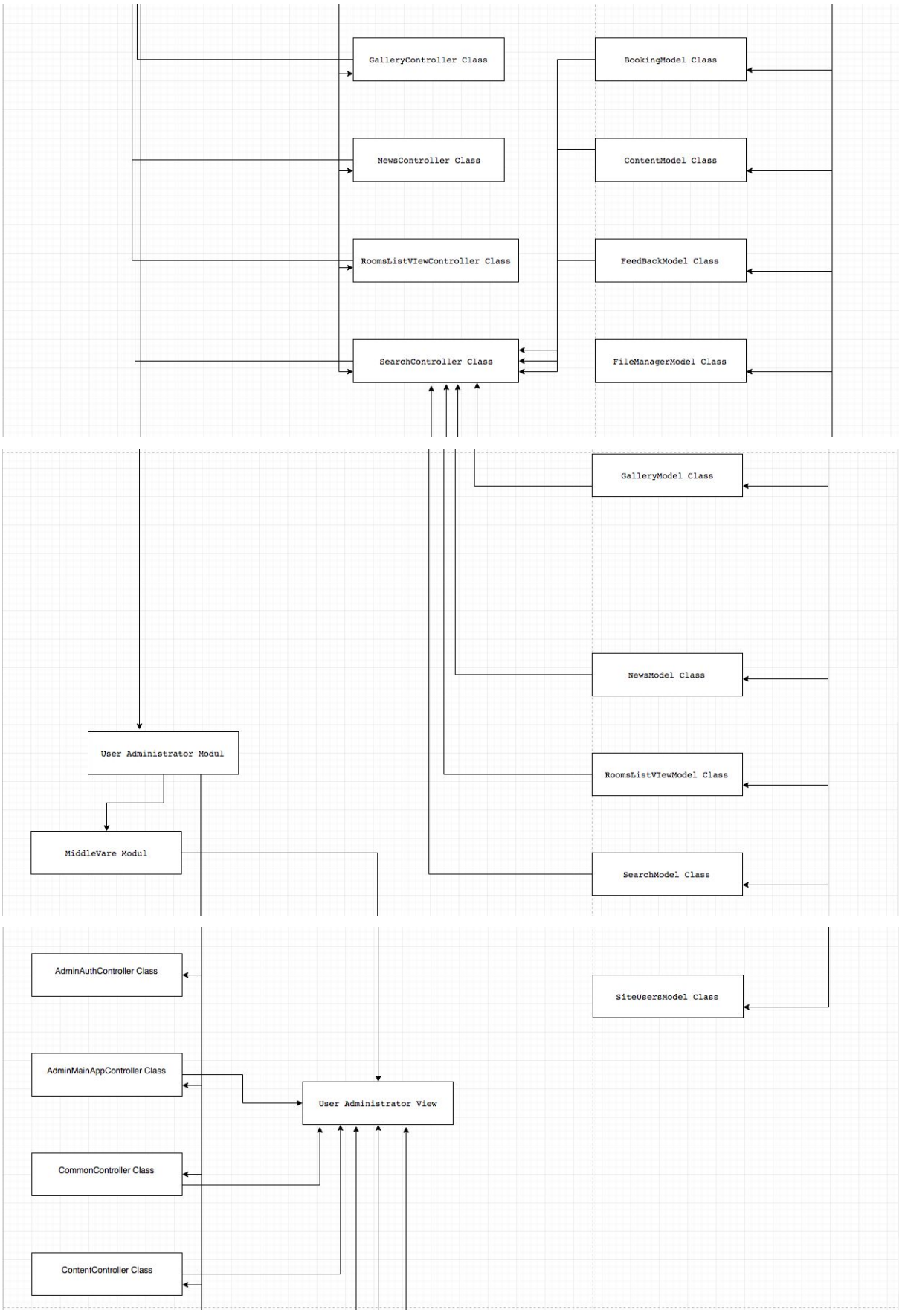
- Third step is to create new model class with the same name as created controller class name.
- Fourth step is to define the path roots in index php file(to avoid recursion).
- Fifth is to create new template view file with the same name as the controller class name.
- Sixth step is to define new url pattern in url patterns php file to refer to which controller and controller function will be send our url.
- Seventh step is to create new table in our database with the same name as controller class name.
- Note that we define the table name, controller class name, model class name the same, because it will allow to CRUD model to connect each other to share information with each other and will be automatically visible in user administration panel as a model.
- In our user administration panel the actions like delete, create, refresh, select, update will be automatically included due to the CRUD model which an incredible simplification in order to develop our model.
- The fields in our model will be automatically defined due to the table fields which will be visible in our model in its turn visible in user administration model.
- This is not quite strict rule to create controller class and model class with the same name we can create single model which will not be visible in user administration panel(or visible) and will play the utilite rule for our application also controllers can play the same role.
- This not necessary that our controller will fetch information from database due to the models, they can also fetch information by themselves(which is not preferable).
- This not necessary to create in one single folder one single controller class we can also create a several of them(it is preferable in some situations) also it relates to the models(it is preferable in some situations).
- The above things can allow us big variety in architecture of creating new pages or websites or some models or controllers as the utilities.

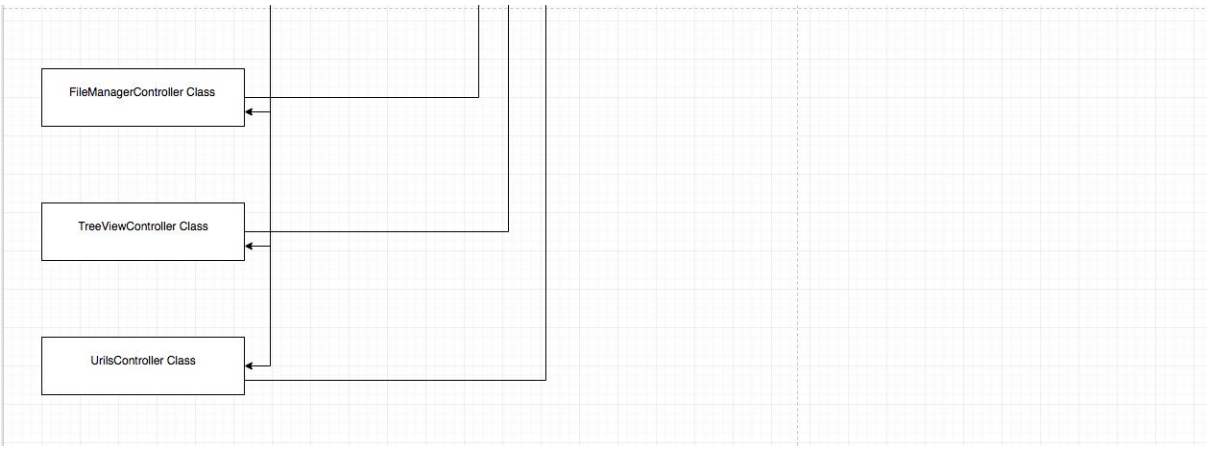
- Menu model.
    - The menu model one of the most difficult model in our application.
    - The main difficulty was for us how to avoid from recursion(because recursion decreases the speed of our frame framework). For this problem we used the power of mysql.
    - We create two table for the menu model (MenuModel_table and MenuRelationsModel_table).
    - The MenuModel_table includes fields such as type(content menu, model menu, link menu, page menu), menuItemTitle, id, link(if it exists), modelId(if the menu is functional then it includes some model), treetreeItemId(which define the parent menu id of this sub menu), content(if the current menu not functional menu but content menu), r_id(which define the current language of this menu), lang_id(which defines also the language id in the text form), visible(which define is this menu visible or not) and pageid(if it is a page id).
    - As we mentioned before the treeItemId field defines to which parent id it belongs if the current menu has parent id(in other case it may be the parent menu itself) which in its turn allow to us avoid from recursive request in the view part and user administration part in our application.
    - Our user administration panel works with ajax(javascript) which allows to users work in a real time which in its turn very convenient to our users.
    - Also our menu system in the view part(user administration part) works with ajax(javascript), we can easily drag and drop menu items and we can do other manipulations also.

- Smarty model.
    - Smarty model is the predefined php script.
    - Smarty allows to us easily fetch data from php and insert it to the view part.
    - Smarty has block separate technique which we used in our application. This method allowed to us to separate view parts such as separate our website footer block, header block, content block, news block, search block and etc. It increased our application speed because it allowed do not download the same block again, they will included if it will be necessary.
    - Smarty template technology collects generated templates in templates_s folder which will not regenerate them for a long time which in its turn increase the speed of our application.
    - We can easily distinguish the smarty code parts in templates files(html).

- Overall overview of connections between classe




GalleryController Class
BookingModel Class
NewsController Class
ContentModel Class
RoomsListVIewController Class
FeedBackModel Class
SearchController Class
FileManagerModel Class
GalleryModel Class
NewsModel Class
User Administrator Modul
RoomsListVIewModel Class
MiddleVare Modul
SearchModel Class
AdminAuthController Class
SiteUsersModel Class
AdminMainAppController Class
User Administrator View
CommonController Class
ContentController Class


FileManagerController Class
TreeViewController Class
UrilsController Class