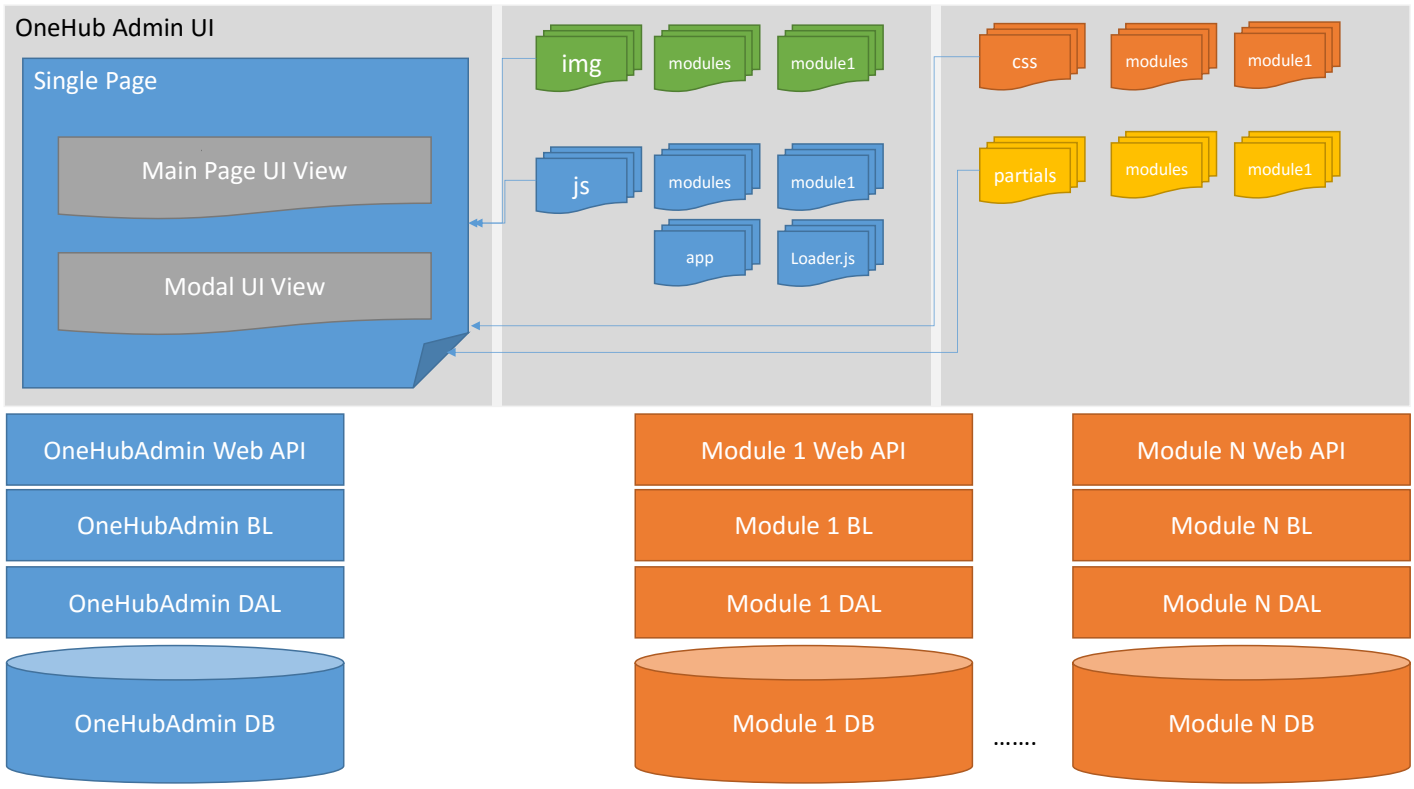
Solution Design



The solution is based on AngularJS UI Routing and composed into single page application (SPA).

The main admin project acts as UI container only, each module must implement the following:

- **Scripts:** including AngularJS controllers, custom scripts, etc. and all those components are placed inside the “js >> modules >> [Module Name]” folder, then using Require.JS the module must update the “modules.js” file with the required scripts to load before the AngularJS application initiation as the following example:

```
var modulesToLoad =
[
 // CMS Application Modules
 'cms/config',
 'cms/createDraftMemoController',
 'cms/viewDraftMemoController',
 'cms/memoActions',
 'cms/viewOutgoingMemoController',
 'cms/viewIncomingMemoController'
   ];
```

- All the registered AngularJS components (controllers, directives, etc.) must be registered under the main AngularJS module “OneHub360Admin”
- **Partials:** including all the required screens, popups and modals and they must be placed under the “partials >> modules >> [Module Name]” folder, keep in mind that all the paths must be relevant to the application root and start with “/”, the module developer has 2 options showing a specific page, one is to show the page within the main body content area or as modal.

- **UI Routes:** the application is designed to work with AngularJS UI routing, so the module developer must declare the state routes as the following example:

```
$stateProvider
    .state('viewdraftmemo', {
      url: "/dm/v/:id",
      templateUrl: "/partials/modules/cms/viewdraftmemo.html"
    });

$stateProvider.state("Modal.createdraftmemo", {
     views: {
       "modal": {
          templateUrl: "/partials/modules/cms/createdraftmemo.html"
       }
     }
  });
```

- **Images, CSS and other components:** the module developer can place other components like images and CSS inside their respective folders as shown in the above diagram, keep in mind not to alter or change the main theme.
- **Services:** the solution designed based on Web API services layer, the module developer must implement the required controllers for the module business logic.
- **Business Logic:** the module developer must implement the required Unit of Work (UOW) components which wraps the module business logic utilizing the framework comes with the solution, for instance the following example shows a UOF component:

```
public  class FeedWorker : AppWorkerBase
       {


       public FeedWorker(WorkerMode mode) : base(mode)
       {
        }

       public bool AddAction(UserAction action)
       {

         var userActionRepository = new UserActionRepository(Context);
         userActionRepository.Insert(action);

        return true;
        }
}
```

- **Data access layer:** the solution utilizing NHibernate as the data access layer framework and is using the XML (.hbm files) mapping method, the module developer must implement the required mapping files, entities and repositories.

Notes:

- Each module must register its start page in the “Modules” table in the “OneHubAdminDB”, the links defined will be used by the main application to construct the navigation and allow the user to access specific module administration.
- Each module then inside its main page can implement his own navigation based on the requirements.
- Each module can have whatever needed folder structure inside its respective module folder.
- Super admin account will be defined during the solution deployment and then can be used to assign modules admins.