Lógica de la arquitectura en CuboSoft Alfrio

Para una correcta comprensión del siguiente documento es necesario que el lector se encuentre familiarizado con temas como:

MVC, JavaScript, HTML, CSS, JQuery, PHP

La aplicación busca ser construida bajo la arquitectura Modelo – Vista – Controlador o MVC, dicha arquitectura busca tener una separación entre las capas de la aplicación.

Para lograr tal fin se tienen tres directorios principales:

1. Controladores
2. Modelos
3. Vistas

Dicho patrón de diseño se implementó de la siguiente manera en CuboSoft Alfrio:

Los archivos contenidos en el directorio “vistas” contienen todos los elementos con los que interactúa el usuario, mientras que el directorio “controladores” contine los archivos de comunicación con la base de datos (modelos) y la lógica de negocio, todas ellas construidas en lenguaje PHP nativo.

Las capas de controladores y vistas se apoyan por dos directorios escritos en lenguaje JavaScript bajo el directorio js los cuales son “modelos” y “vistas”. Dentro del directorio modelos de js se encuentran archivos JavaScript que contienen funciones que interactúan a través de AJAX con los archivos de la capa controladores de PHP.

Para el trabajo de animaciones en las vistas se trabajó con la librería JQuery en su versión 3.4.1 según el documento ubicado en ./bower\_components/jquery/README.md

La trazabilidad de una petición tradicional que usa este modelo de capaz a través de la captura del evento “click” en un botón sería la siguiente:

1. A través del siguiente código se haría captura del evento “click” del botón haciendo uso de la librería JQuery así:

$(“#id\_boton”).on(“click”,function(){

Paso 2.

});

Ó

$(“#id\_boton”).click(function(){

Paso 2.

});

Donde: id\_boton sería el id del botón asignado en la vista.

1. La petición a un controlador PHP se haría por una estructura similar a la siguiente:

$.ajax({

url: "../controladores/CT\_am\_usuarios.php", -> controlador donde se va a realizar la petición

data: { 'caso': '4', 'datosAEnviar': datosAEnviar }, -> los datos enviados en la petición

type: "POST", -> el método usado para enviar los datos

success: function (respuesta) { -> la respuesta enviada por el controlador una vez procesada la información enviada

var obj = JSON.parse(respuesta);

-- logica a realizar con los datos enviados por el controlador --

},

error: function (jqXHR, textStatus, errorThrown) { -> en caso de falla, muestra una alerta

alert("Error en function retornarAMUsuarios(opcion, data) {...\nError desde el servidor. Por favor informe a soporte");

}

});

1. Dentro del evento “success” del anterior modelo de función, se integra la lógica que será mostrada al usuario en la vista donde arrancó el evento “click” del primer paso.

Se ha buscado que cada archivo contenido en los directorios js/modelos/, modelos y controladores sean similares a los nombres de las tablas de la tablas le la base de datos de manera que sea fácilmente comprensible la lógica a aplicar en cada caso.

Para el caso de las vistas, los nombres de los archivos contenidos en js/vistas tienen los mismos nombres de las vistas de manera que cada archivo js interactúe con su par.

Para la llamada de una función JavaScript usando JQuery dentro de uno de los archivos js del directorio vistas al cargar la página se usa la siguiente estructura:

$(document).ready(function () { -> carga predeterminada usada por JQuery para el momento en que la vista se renderiza

//función llamada una vez la carga de la vista

function nombreFunction(opcion, parametros);

});

Directorios de la aplicación:

.vscode -> directorio usado por VSCode para la sincronización de los archivos de local al hosting a través de FTP

Adicionales-> contiene los archivos apoyan la lógica en las vistas, por ejemplo, el archivo varios.php contiene funciones con código PHP que muestra código HTML en las vistas donde las funciones son llamadas.

Bower\_components -> contiene buena parte de los archivos usados por la plantilla usada para la aplicación

Cls -> por favor remitirse a Ricardo

Controladores -> contiene todos los archivos PHP que responden a las peticiones AJAX enviadas desde js/modelos o desde js/vistas

Cotizaciones -> en este directorio se encuentran las cotizaciones generadas por el sistema en formato .pdf

Css -> como su nombre lo indica contiene los archivos .css de la aplicación, algunos personalizados y otros de la plantilla usada.

Ctr -> por favor remitirse a Ricardo

Db-> estructura de la base de datos

Dist -> archivos de la plantilla

Documentacion -> este documento

Imágenes -> las imágenes de la aplicación y otras agregadas por necesidad, como por ejemplo las imágenes de los usuarios.

Img\_inve -> las imágenes de los repuestos

Js -> los archivos JavaScript que usa la aplicación para la lógica.

Modelos -> documentos en formato PHP que tienen las consultas ejecutadas por la aplicación a través de clases que emulan el nombre de las tablas de la base de datos.

Nbproject -> directorio generado por NetBeans al inicio del proyecto

Pdf -> contiene los archivos que generan las cotizaciones para los repuestos y los equipos además de los que hacen el envío a través de correo electrónico.

PHPMailer -> librería PHP para envío de correos.

Plugins -> archivos de la plantilla.

Tcpdf -> librería para la generación de archivos de tipo pdf

Vistas -> ya comentada con anterioridad.