Descripción del proceso de desarrollo

Este proceso da inicio al dirigirnos a la escuela Diego Bustillos y detectar que en dicha escuela requieren de un sistema de carnetizacion.

Se realiza en análisis de requerimientos para el sistema donde encuentra que el sistema deberá registrar los estudiantes con sus respectivos representantes y los docentes estos últimos tendrán acceso al sistema para imprimir el carnet del estudiantes, el sistema deberá ser capaz de tomar la foto al estudiante e imprimir el carnet con todos sus datos, además deberá imprimir reportes pdf.

Elección del entorno de trabajo y tecnologías a usar:

Para este sistema se realiza una investigación tomando en cuenta sus requerimientos para lo cual se decide hacer uso de librerías y lenguajes que ayuden en el proceso de desarrollo ademas de un Entorno de desarrollo integrado (IDE).

Para este proyecto se elige el lenguaje php junto con la Librería CcMvc (Librería Mvc Orientada a Objetos php para el desarrollo de aplicaciones y sitios Web) se elige esta librería porque contiene clases y funciones que pueden dar agilidad y seguridad al proceso de desarrollo además de funciones como convertir HTML en pdf con DomPDF y uso de plantillas de diferentes formatos aunque para este proyecto se decide usar plantillas smarty por tener una curva de aprendizaje más corta, Smarty es un motor de plantillas php que mejora la separación del diseño lógico del visual, esto es del lado del servidor.

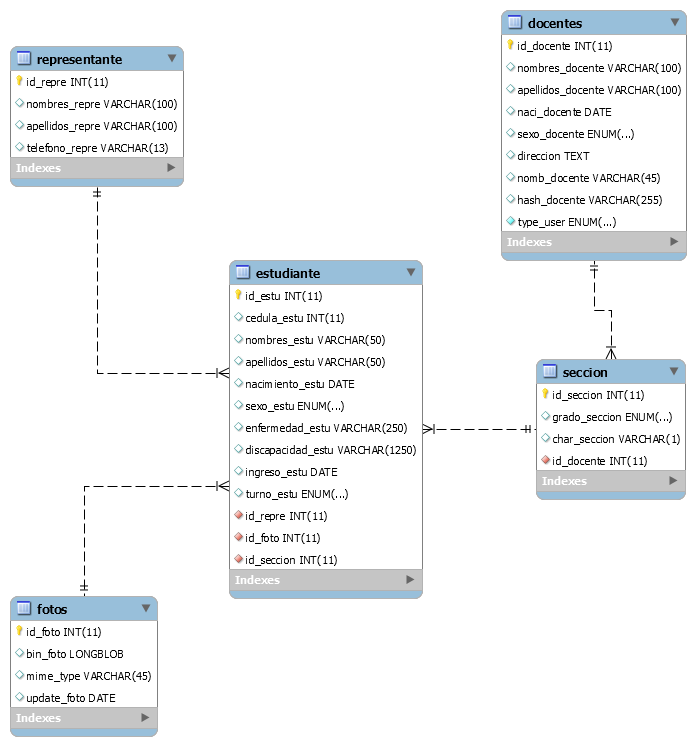
Del lado de Cliente (Navegador) se decide usar Bootstrap una librería Css que proporciona clases Css prediseñadas listas para usar la misma usa Jquery de JavaScript, para la acción de tomar fotos desde el sistema se elige la librería WebCamJs-Master la cual está programada en JavaScript y ofrece funciones fáciles de usar para tomar fotos con la cámara web.

Para la base de datos se elige Mysql ya que por estar basada en modelos de entidad relación.

Se elige usar el IDE NetBeans para la programación y Mysql WorkBench para la creación el modelo de entidad relación de la base de datos

Base de datos:

Ya realizado el análisis de requerimientos se procede a crear un modelo relacional de bases de datos quedando de la siguiente manera.



En esta base de datos encontramos 5 tablas que se describirán a continuación:

Tabla estudiante: Esta tabla contendrá los datos correspondientes a los alumnos esta tabla tiene relación con las tablas representante 1-N, con fotos 1-1, y con sección 1-N.

Tabla representantes: Esta tabla contendrá los datos de los representantes de cada estudiante la misma solo se relaciona con estudiante N-1.

Tabla sección: Esta tabla almacena los datos de la sección relacionándose con estudiante N-1 y con docentes 1-1.

Tabla docentes: Almacena la información de los docentes así como su nombre de usuario y hash de contraseña para el ingreso al sistema, esta tabla solo se relaciona con sección 1-1.

Tabla fotos: Esta tabla almacenara las fotos tomadas a los estudiantes y solo tiene relación con la tabla estudiante 1-1.

Diseño de la interface del sistema:

En esta etapa se procede a crear el diseño de la interface del sistema, este será realizado usando html5, css3 y bootstrap3 diseñando un modelo de caja para la aplicación creando así un diseño general para el sistema.



Programación:

Esta etapa comienza descargando de <https://github.com/ever23/CcMvcBasicProject> una Base proyecto CcMvc, esto proporciona la estructura de directorios para comenzar a desarrollar la aplicación usando CcMvc, antes de proseguir es necesario destacar que CcMvc provee un entorno de trabajo Modelo Vista Controlador Orientado A Objetos, por lo cual divide la programación en tres partes.

Los Modelos son la estructuras de datos que alimentan la aplicación estas son clases php que se almacenan en el directorio protected/model/.

Las Vistas representa todo el contenido que será mostrado al usuario las mismas se dividen en dos categorías, los layauts son el diseño general HTML de la aplicación así como menús links y imágenes que aparecen en todas las pantallas de la aplicación y se almacenan en protected/layaut/, los view son el diseño HTML especifico de cada pantalla y se almacenan en protected/view/.

Los controladores estos son clases php extendidas de la clase \Cc\Mvc\Controllers cuyos métodos son los encargados de realizar el proceso de obtención de datos de los Modelo y enviarlos a las vistas para ser mostrados al usuario, estas clases son instanciadas automáticamente por CcMvc según la petición del cliente (Navegador) y parámetros de los métodos serán resueltos y pasados automáticamente. Estas clases se almacenan en protected/controllers/.

Para más documentación sobre el funcionamiento de CcMvc visite <http://ccmvc.com.ve/documentacion>

Para empezar con la descripción del proceso de programación se modifica el archivo index.php donde encontramos el siguiente código:

<?php

include ("../CcMvc/CcMvc.php"); // CcMvc

$config = dirname(\_\_FILE\_\_) . "/protected/configuracion.php"; // configuracion

$app = CcMvc::Start($config, "Carnetizacion Safe Off");

$app->Run();

Donde incluimos la librería CcMvc luego con la clase CcMvc llamamos al método CcMvc::Start asignándolo a la variable $app donde pasamos por parámetro en nombre de archivo de configuración y el nombre del sistema este método retornara un objeto de la clase \Cc\Mvc, luego llamamos el método Run que dará inicio a la ejecución del sistema

Configurar CcMvc para el sistema:

Se define la configuración para CcMvc en el archivo protected/configuración.php donde se configura la librería para uso de especifico del sistema en este archivo se definen el acceso a bases de datos, autenticación y sesiones de usuario entre otros, este archivo retorna un array con la configuración

Para este proyecto definiremos el índice

'Response' => [

'Accept' => [

'text/html, application/xhtml+xml, application/xaml+xml, application/pdf' =>

[

'layaut' => 'main.tpl',

],

]

],

Para que se cargué por defecto el layaut protected/layaut/main.tpl cuando el tipo de respuesta sea text/html, application/xhtml+xml, application/xaml+xml o application/pdf.

Además de esto se configura el acceso a la base de datos con el índice

'DB' => include ('Mysqli.php')

Y se crea el archivo protected/Mysqli.php que retorna un array con la configuración de la base de datos esto incluye el nombre de la clase que manejara la base de datos que para este proyecto se usa la clase interna de CcMvc \Cc\Mvc\MySQLi y los parámetros de su constructor que son el nombre de host, el nombre de usuario, contraseña y nombre de base de datos.

Una vez configurado CcMvc se procede con los modelos de datos esto incluye formulario, sesiones y autenticación de usuarios:

Formularios: para crear los modelos de formularios se crean las clases \Cc\Mvc\ FormEstudiante, \Cc\Mvc\FormRepresentante, \Cc\Mvc\FormDocente, \Cc\Mvc\FormDocenteEditar, \Cc\Mvc\FormDocenteEditarContrasena dichas clases serán extendidas de la clase interna de CcMvc \Cc\Mvc\FormModel usada para validación de formularios, en cada una de estas clases se define dos métodos Campos y OnSubmit, el método Campos se usa para definir los campos que tendrá cada formulario y los parámetros para la validación automática de cada uno y el método OnSubmit se usar para validación manual de ser necesario.

Sesiones y autenticación de usuarios: Para este propósito se crea la clase \Cc\Mvc\Autenticacion extendida de la clase interna de CcMvc \Cc\Mvc\AuteticateUserDB que ofrece métodos para autenticación de usuarios con base de datos bien seguros, en esta clase se definirán dos métodos InfoUserDB y OnFailed, el método InfoUserDB se usa para dar información a la clase sobre la tabla en la base de datos que almacena los usuarios ademas de los campos de dicha tabla que guardan el nombre de usuario, el hash de contraseña y el tipo de usuario, en el método OnFailed se definen las acciones que se realizara cuando la autenticación falle, usando el método interno IsFailed se puede obtener el tipo de falla en este caso cuando la autenticación falle por que no exista un usuario en la sesión se redireccionara a el controlador Cuser::index.

Para que esta clase tenga algún efecto sobre el sistema se modifica el archivo de configuración protected/configuración.php y se agrega el indice

'Autenticate' =>

[

'class' => '\\Cc\\Mvc\\Autenticacion',

'SessionName' => 'CarnetizacionSO',

'SessionCookie' =>

[

'cahe' => 'nocache, private',

'time' => '+6 hour',

'httponly' => true,

],

]

En el cual se define el nombre de la clase que manejara la autenticación, el nombre se la sesión y los parámetros de las cookies de sesión una vez hecho esto solo bastara que los controladores que necesiten control de acceso implementen la interface \Cc\Mvc\AccessUserController.

Creando plantillas Smarty para los layauts y views:

Primero se crea el layaut /protected/layaut/main.tpl el cual contiene etiquetas html y etiquetas Smarty para imprimir el contenido generado por los controladores y demás variables pasadas al layaut

Los views también son archivos .tpl que se almacenan en protected/view/ también contiene etiquetas html y Smarty para imprimir las variables pasadas al view los view se irán creando según se necesiten en los controladores

Definiendo Controladores:

Clase base controladora: para este proyecto se crea una clase base controladora de la cual heredaran todas las demás clases esta clase es \Cc\Mvc\ExtendsController definida en protected/model/ExtendsControler .php que a su vez hereda de la clase interna de CcMvc \Cc\Mvc\Controller en esta clase definimos en constructor el cual tiene como objeto mandar a los layauts y view las variables de sesión y si existe una cookie que se llame mensaje la pasara también y luego la borrara

Cindex: definido en protected/controllers/Cindex.php Este controlador en principio solo tiene un método

Index: este método solo será responsable de mostrar el view /protected/view/index.tpl el cual contiene el diseño del menú principal.



Cestudiante: definido en protected/controllers/Cestudiante.php. Este controlador será encargado realizar los procesos relacionados con los estudiantes, cuenta con los métodos:

Index: encargado de obtener de la base de datos la información del estudiante usando la clase \Cc\Mvc\DBtabla declarándola como parámetro y el nombre del parámetro será el nombre de la tabla a obtener CcMvc resuelve esto y pasa el objeto correspondiente se realiza un select o una búsqueda avanzada según existan en get las variables ‘q’ o ‘id\_estu’ luego se pasa el resultado al view /protected/view/estudiante/index.tpl



Carnet: Este método es el encargado de mostrar el carnet obteniendo de la base de datos la información del estudiante así como la foto creando el carnet con la clase \Cc\ImageGD para crear la imagen del carnet, luego se pasa al view protected/view/estudiante/carnet.tpl

Info: Este método obtendrá la información de un estudiante y su representante de la base de datos y la pasara a view protected/view/estudiante/info.tpl.

Reportes: Obtiene la información de los estudiantes de la base de datos y luego pasándola al view protected/view/estudiante/reporte.tpl. El contenido generado por este método puede ser transformado por CcMvc automáticamente a Pdf si el cliente (Navegador) así lo requiere, esto se logra haciendo que la clase controladora implemente la interface \Cc\Mvc\ExtByController y definiendo el método ExtAccept en el cual se define que métodos será transformable el contenido y en que formato.



Insertar: Este método se encargara de mostrar el formulario de insertar estudiante y si recibe los datos los inserta en la tabla, aquí se usa u modelo de formulario creado para este proyecto el cual es la clase \Cc\Mvc\FormEstudiante y está definida en protected/model/ ModelosDeFormularios.php esta clase se utiliza para validar la información recibida del formulario e imprimir el formulario luego se pasa el objeto de formulario al view protected/view/estudiante/form.tpl para que este sea impreso en protected/view/estudiante/form.tpl se definen funciones JavaScript que usando la librería WebCamjs-master hacen posible el tomar la foto con la cámara Web si la foto es tomada el formulario será enviado por Ajax a método guardar si no será enviado al el mismo.



Guardar: Este método se encarga de insertar y editar un estudiante mediante Ajax esto es cuando se toma la foto en el formulario en este se usa la clase \Cc\Mvc\FormEstudiante solo para recibir y validar los datos una vez completada la acción se emite una respuesta en formato json.

Editar: Este método también usa la clase \Cc\Mvc\FormEstudiante para el formulario de editar y también el view protected/view/estudiante/form.tpl la diferencia entre este método y el método insertar es que se consulta el estudiante y se pasa los datos obtenidos al formulario mediante el método \Cc\Mvc\FormEstudiante::DefaultValue y si el formulario se recibe se procede a actualizar el registro en la base de datos.

Eliminar: este método elimina un estudiante de la base de datos si el estudiante no existe emite un mensaje si la eliminación fue exitosa redirige al método index

LoadFoto: este método es privado esto quiere decir que no es accesible para el cliente (navegador), es usado por los métodos insertar, editar y guardar para insertar o editar la foto de un estudiante en la base de datos

AccessUser: Este método es de la interface \Cc\Mvc\AccessUserController y no es accesible al cliente (navegador) define tipos de usuario tendrán acceso a los diferentes métodos

ExtAccept: Este método es de la interface \Cc\Mvc\ ExtByController y no es accesible al cliente (navegador) define que métodos será transformable el contenido y en que formato.

Crepresentante: definido en protected/controllers/Crepresentante.php. Este controlador será encargado realizar los procesos relacionados con los representantes, cuenta con los métodos:

Index: encargado de obtener de la base de datos la información del representante se realiza un select o una búsqueda avanzada según existan en get las variables ‘q’ o ‘id\_repre’ luego se pasa el resultado al view /protected/view/representante/index.tpl

Info: Este método obtendrá la información de un representante y sus representados de la base de datos y la pasara a view protected/view/representante/info.tpl.

Insertar: este método se encarga de mostrar el formulario de insertar representante y si recibe los datos los inserta en la tabla usa el modelo de formulario \Cc\Mvc\FormRepresentante definido en protected/model/ ModelosDeFormularios.php para recibir y validad el formulario luego lo pasa al view protected/view/representante/form.tpl.

Editar: Se encarga de editar un registro de la tabla representante usa el modelo de formulario \Cc\Mvc\FormRepresentante definido en protected/model/ ModelosDeFormularios.php para recibir y validad el formulario luego lo pasa al view protected/view/representante/form.tpl.

Eliminar: está encargado de eliminar un registro de representante si no existe emite un mensaje si la eliminación fue exitosa redirige al método index.

AccessUser: Este método es de la interface \Cc\Mvc\AccessUserController y no es accesible al cliente (navegador) define tipos de usuario tendrán acceso a los diferentes métodos

Cfotos: definido en protected/controllers/Cfotos.php. Este controlador tiene como único propósito mostrar la foto de un estudiante

Index: obtiene de la base de datos la foto de un estudiante y la envía al navegador

Cdocente: definido en protected/controllers/Cdocente.php. Este controlador será encargado realizar los procesos relacionados con los docentes, cuenta con los métodos:

Index: encargado de obtener de la base de datos la información de los docentes, se realiza un select o una búsqueda avanzada según existan en get las variables ‘q’ o ‘id\_docente’ luego se pasa el resultado al view /protected/view/docente/index.tpl

Info: Este método obtendrá la información de un docente y sus alumnos de la base de datos y la pasara a view protected/view/docente/info.tpl.

Insertar: este método se encarga de mostrar el formulario de insertar representante y si recibe los datos los inserta en la tabla

Insertar: este método se encarga de mostrar el formulario de insertar docentes y si recibe los datos los inserta en la tabla usa el modelo de formulario \Cc\Mvc\FormDocente definido en protected/model/ ModelosDeFormularios.php para recibir y validad el formulario también usa la clase \Cc\Mvc\Autenticacion definida en protected/model/Autenticacion.php para insertar en la tabla ya que se configuro la tabla docentes como tabla de usuarios y esta clase tiene métodos para insertar y editar usuarios además crea un hash automáticamente para contraseña bastante seguro. Luego pasa el formulario al view protected/view/docente/form.tpl.

Editar: Se encarga de editar un registro de la tabla docentes usa el modelo de formulario \Cc\Mvc\FormDocente definido en protected/model/ ModelosDeFormularios.php para recibir y validad el formulario luego lo pasa al view protected/view/docente/form.tpl.

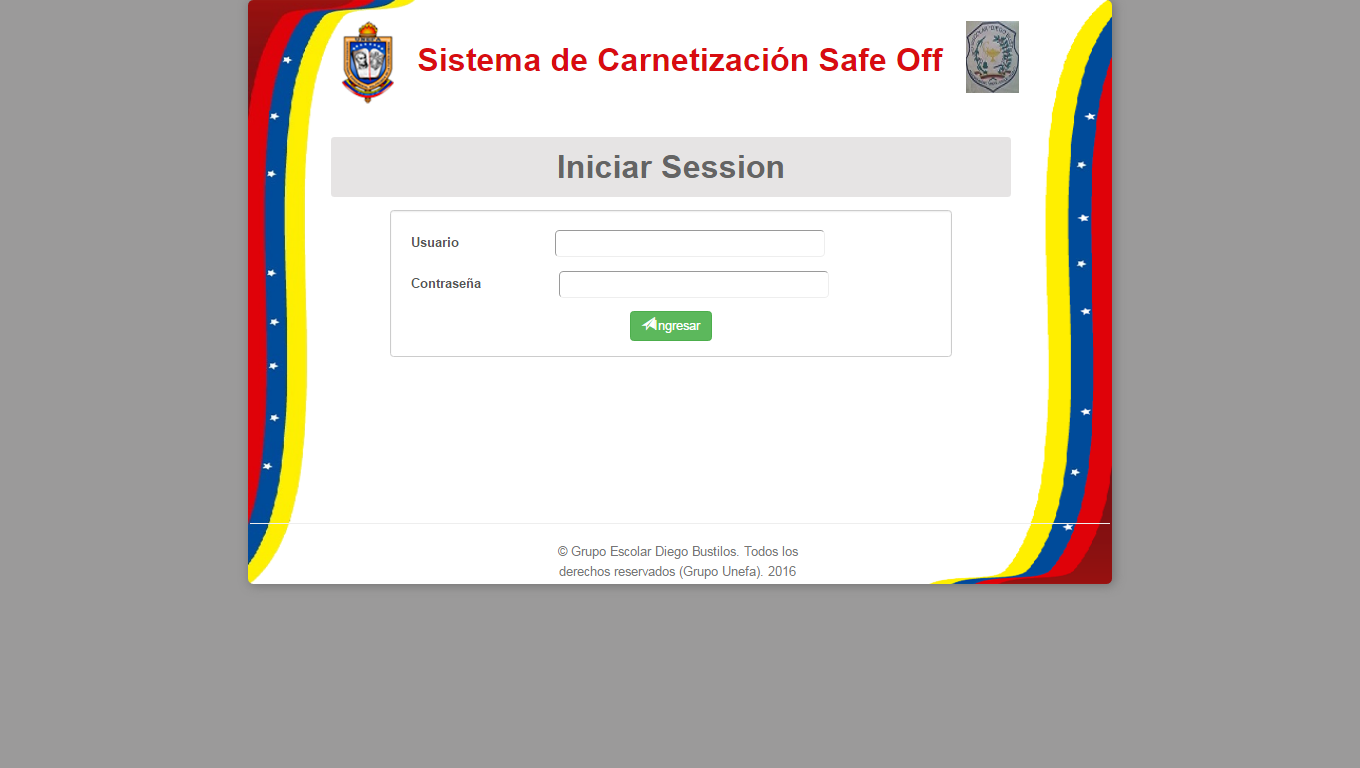
Eliminar: está encargado de eliminar un registro de docentes si no existe emite un mensaje si la eliminación fue exitosa redirige al método index.

CambiarContrasena: este método se encarga de mostrar el formulario de cambio de contraseña y si se reciben los datos se cambia la contraseña usando la clase \Cc\Mvc\Autenticacion definida en protected/model/Autenticacion.php y cargar el view protected/view/docente/ contrasena.tpl

AccessUser: Este método es de la interface \Cc\Mvc\AccessUserController y no es accesible al cliente (navegador) define tipos de usuario tendrán acceso a los diferentes métodos.

Cuser: definido en protected/controllers/Cuser.php. Este controlador se encarga de iniciar y cerrar las sesiones de usuario cuenta con los métodos:

Index: este método se encarga de mostrar el formulario de inicio de sesión y si recibe los datos procede a iniciar la sesión usando la clase \Cc\Mvc\Autenticacion definida en protected/model/Autenticacion.php también usa \Cc\Mvc\LoginFormModel clase predefinida de CcMvc para formularios de inicio de sesión, luego se pasa el formulario al view protected/view/user/index.tpl



Close: este método solo se encarga de cerrar la sesión de usuario usando la clase \Cc\Mvc\Autenticacion definida en protected/model/Autenticacion.php para este propósito