“PROYECTO”

“DOCUMENTACION”

Presentado por:

JOHN EDWARD ARIAS CARDOZO
2º de Desarrollo De Aplicaciones WEB

MODULO DE PROYECTO

I.E.S Albarregas

TÍTULO DEL PROYECTO

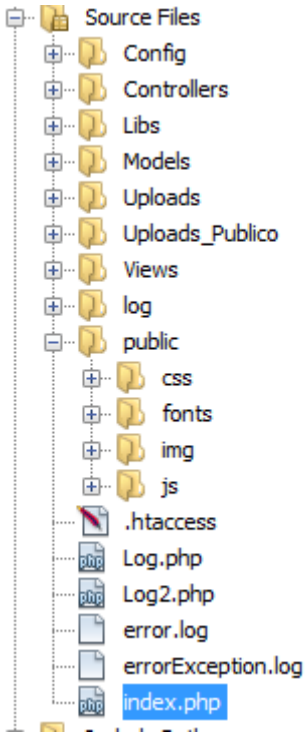
Nombre Del Proyecto:

Tablón de Anuncios Instituto

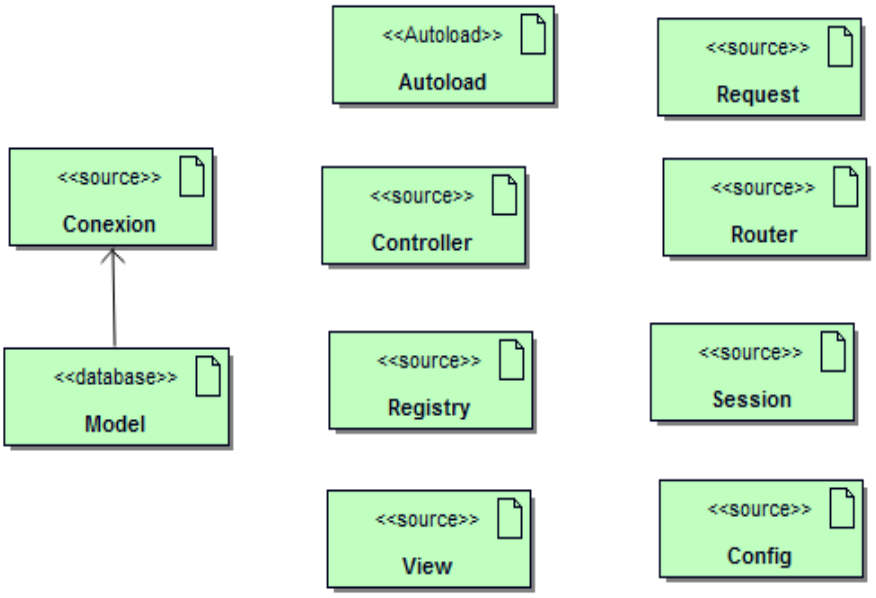
Tipo de proyecto:

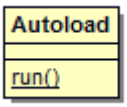
De innovación aplicada.

ESTRUTURA DEL PROYECTO



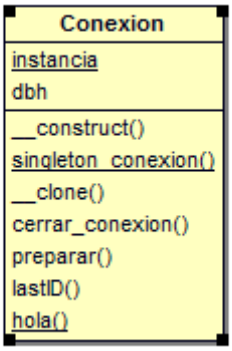
PAQUETE CONFIG





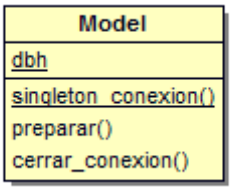
AUTOLOAD: registra cualquier número de autocargardores, posibilitando que las clases e interfaces sean cargadas automáticamente si no están definidas actualmente. Al registrar autocargardores, a PHP se le da una última oportunidad de cargar las clases o interfaces antes de que falle por un error

```
class Autoload {
    //put your code here
    public static function run() {
        spl_autoload_register(function($class) {
                $ruta = str_replace("\\", "/", $class) . ".php";
                include_once $ruta;
//                  echo $ruta;

        });
    }
}
```



CONEXIÓN: Instancia la conexión con la base de datos utilizando el patrón singlenton diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

```php
class Conexion {

    private static $instancia;
    private  $dbh;

    //usaremos $this-> dbh porque es un valor no statico.
    //usaremos self::$instancia para valores estaticos.

     private function __construct() {...35 lines }

    protected static function singleton_conexion() {...13 lines }

    //Evita que el objeto se puede clonar
    protected function __clone() {...3 lines }

    protected function cerrar_conexion() {...4 lines }

    protected function preparar($sql) {...3 lines }

    protected function lastID() {...3 lines }


}
?>
```



Model: Esta clase es la que hereda de la clase conexión y es la clase padre de las clases models

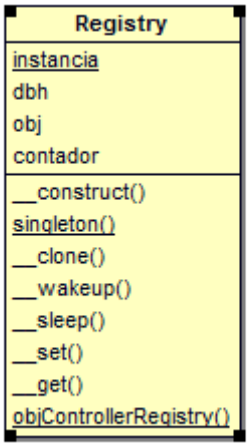
```
class Model extends Conexion{

    //put your code here
    protected static $dbh;




    protected static function singleton_conexion() {
         self::$dbh= parent::singleton_conexion();
    }


    protected function preparar($sql) {
        self::$dbh=parent::preparar($sql);
    }


    protected function cerrar_conexion() {
        self::$dbh= parent::cerrar_conexion();
    }


}
```

CONFIG: contiene todas las constantes que se utilizan mayormente en la aplicación.

```
define('BASE_URL',"http://localhost/tablonAnuncios15/");

define('DEFAULT_CONTROLLER', "index");
define('DEFAULT_TEMPLATE', "defecto");
define('PUBLIC_TEMPLATE', "public");


define('APP_NAME', "Mi fraweor");
define('APP_COMPANY', "www.edward.com");
define('APP_SLOGAN', "mi primer framework");

define('DB_HOST','localhost');
define('DB_USER','dwes');
define('DB_PASS','1234');
define('DB_NAME','tablon_anuncios');
define('DB_CHAR','utf8');
define('HASH_KEY','5946ef12beead');
```

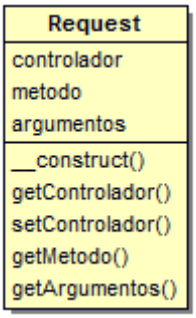
CONTROLLER: Esta clase es la clase padre del paquete controller contiene las funciones particulares que utiliza sus hijas al ser una clase abstrata no puede ser instanciada

```
abstract class Controller {

    //put your code here

    protected $view;
    protected $registry;

    public function __construct() {...5 lines }

    abstract public function index();

    protected function loadModel($modelo) {...20 lines }

    protected function objController($controller) {...23 lines }

    protected function getPost(array $clavePost, array $sinValor = null) {

    protected function getPostSingle($valor) {...12 lines }

    protected static function redireccionar($ruta = false) {...9 lines }

    protected function filtrarInt($id) {...8 lines }

protected static function getLibrary($libreria) {...8 lines }

protected function pintar($titulo, $metodo, $controllador = false, array $css = nu

protected static function getUrlImg($categoria) {...13 lines }

protected static function getUrLPdf($categoria) {...11 lines }

protected static function getUrlImgPublico($categoria) {...10 lines }

protected static function getUrLPdfPublico($categoria) {...10 lines }

protected static function compararFecha($fecha) {...7 lines }

protected static function definirPagina($registros, $pagina, $limite = null) {...

protected function convertSaveCategoriaPublica($categoria) {...10 lines }

protected function convertCategoriaPublica($categoria) {...10 lines }

protected function validarEmail($mail) {...6 lines }
```



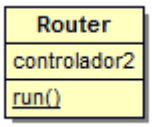
REGISTRY: Esta clase la donde se almacena las instancia de todos los objetos de aplicación utilizando singlenton diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Por lo que cuando se instancia un objeto si se vuele a llamar ese mismo objeto no se crea uno nuevo sino que se hace referencia al que ya ha sido creado con anterioridad

```php
class Registry {

    //put your code here

    private static $instancia;
    private $dbh;
    private $obj;

    private $contador;
    //No se puede instaciar la clase
    private function __construct() {
        $this->contador =0;
    }

    public static function singleton() {...13 lines }

    public function __clone() {...3 lines }

    public function __wakeup() {...3 lines }

    public function __sleep() {...3 lines }

    public function __set($name, $value) {...3 lines }

    public function __get($name) {...7 lines }

    public static function objControllerRegistry($obj) {...7 lines }

}
```



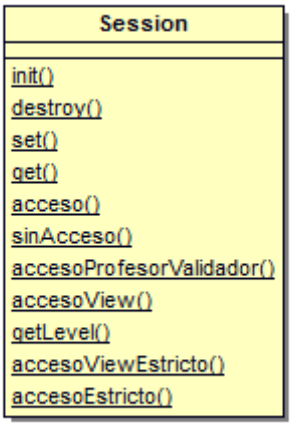
REQUEST: Esta clase es la encargada de recibir las peticiones por la URL y se la pasa a la Clase Router.

```php
class Request {

    private $controlador;
    private $metodo;
    private $argumentos;

    public function __construct() {...32 lines }

    public function getControlador() {...3 lines }

    public function setControlador($controlador) {...3 lines }


    public function getMetodo() {...3 lines }

    public function getArgumentos() {
        return $this->argumentos;
    }

}
```



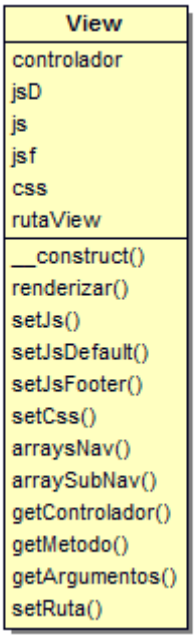
ROUTER: Esta clase es la encargada de instanciar la clase del paquete controllers que fue solicitada por el Request.

```php
class Router {

    private $controlador2;

    //put your code here

    public static function run(Request $request) {

      $controlador = $request->getControlador() . 'Controller';
        $rutaController = ROOT . 'Controllers' . DS . $controlador . '.php';

        $metodo = 'index';
        $argumentos = $request->getArgumentos();


        if (is_readable($rutaController)) {

            $controlador = "Controllers\\" . $controlador;
            $controlador = Registry::objControllerRegistry($controlador);



            if (is_callable(array($controlador, $request->getMetodo()))) {

                if (strpos($request->getMetodo(), '_') !== 0) {

                    $metodo = $request->getMetodo();
                }

             if (isset($argumentos)) {
                 call_user_func_array(array($controlador, $metodo), $argumentos);
             } else {
                 call_user_func(array($controlador, $metodo));
             }
         } else {
             header('location:' . BASE_URL."error");
         }
     }

}
```



SESSION: Esta clase es la encargada de memorizar la variables que se necesitan para la el funcionamiento de validaciones de usuario, peticiones de datos mantener la información propia de usuario que está conectado.

```
class Session {

    //put your code here
    public static function init() {...3 lines }

    public static function destroy($clave = false) {...17 lines }

    public static function set($clave, $valor) {...6 lines }

    public static function get($clave) {...4 lines }

    public static function acceso($tipo) {...10 lines }

    public static function sinAcceso() {...6 lines }

    public static function accesoProfesorValidador() {...6 lines }

    public static function accesoView($level) {...9 lines }

    private static function getLevel($tipo) {...13 lines }

    public static function accesoViewEstricto(array $level, $noAdmin = false) {...21

    public static function accesoEstricto(array $level, $noAdmin = false) {...23 line

}
```

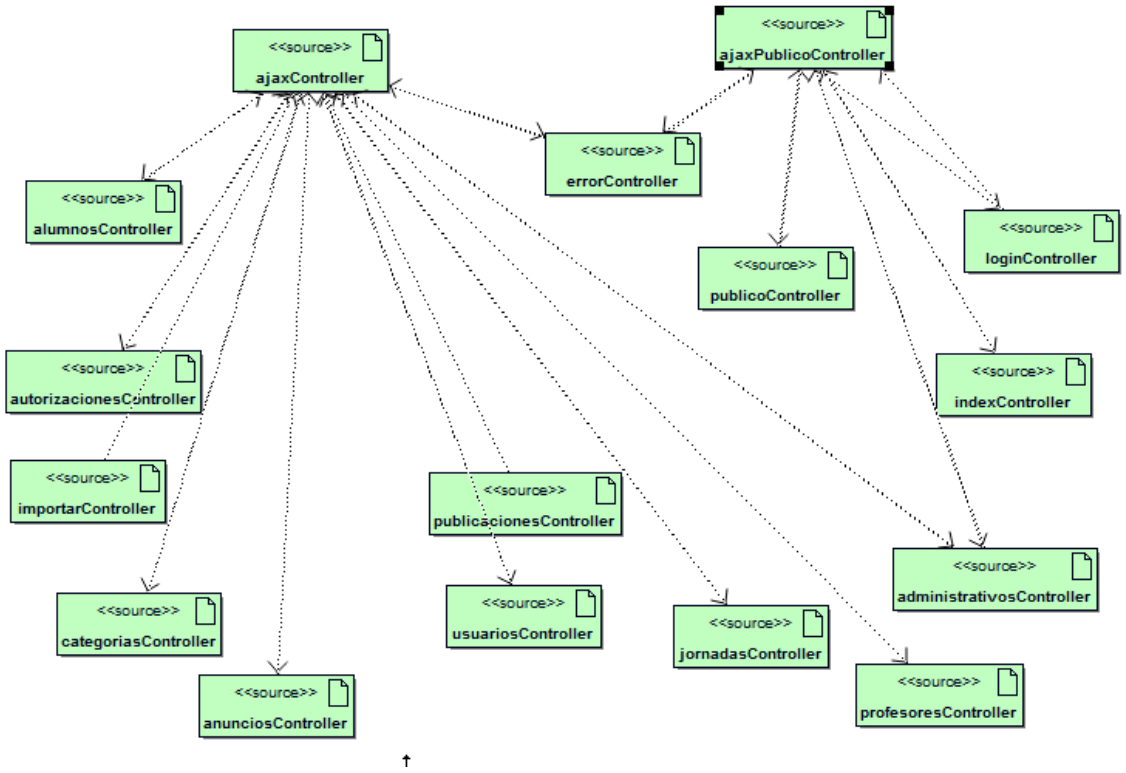


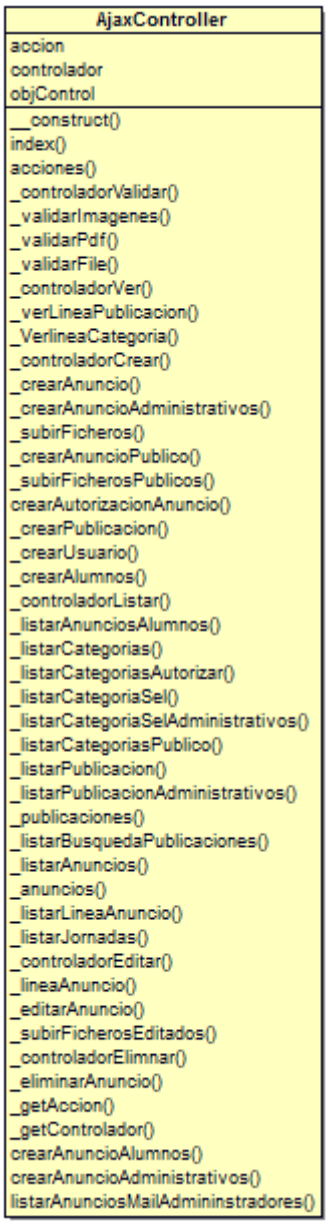
VIEW: Esta clase es la encargada de renderizar las vistas que se muestran en la web

```
class View {

    //put your code here
    private $controlador;
    private $jsD;
    private $js;
    private $jsf;
    private $css;
    private $rutaView;

    public function __construct(Request $request) {...10 lines }

    public function renderizar($vista, $controlador = false, $item = false) {...52 l

    public function setJs(array $js, $controlador = false) {...15 lines }

    public function setJsDefault(array $jsD) {...12 lines }

    public function setJsFooter(array $jsf, $controlador = false) {...15 lines }

    public function setCss(array $css, $controlador = false) {...15 lines }

    private function arraysNav() {...78 lines }

    private function arraySubNav() {...72 lines }

    public function getControlador() {...3 lines }
```

PAQUETE CONTROLLERS

En este paquete se encuentra las clases que se encargar del negociado de la aplicación y son las que se comunican con las clases modelos y las vistas





AJAXCONTROLLER: Esta clase es la encargada de gestionar las peticiones que se hacen de Ajax en la parte privada por lo que de acuerdo a la solicitud llama al controlador que se necesita para obtener datos o presentar datos.

```
class AjaxController extends \Config\Controller {

//put your code here
    private $accion;
    private $controlador;

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

    public function index($argumentos = false) {...16 lines }

    /**
     * Funciones de accion
     */
    private function acciones() {...24 lines }

    /**
     * Funciones para validar
     */
    private function _controladorValidar() {...16 lines }

    private function _validarImagenes() {...6 lines }

    private function _validarPdf() {...7 lines }

    private function _validarFile() {...7 lines }


 /**
  * Funciones para ver
  */
 private function _controladorVer() {...10 lines }

 private function _verLineaPublicacion() {...20 lines }

 private function _VerlineaCategoria() {...10 lines }

 /**
  * Funciones para crear
  */
 private function _controladorCrear() {...32 lines }

 private function _crearAnuncio() {...18 lines }

 private function _crearAnuncioAdministrativos() {...12 lines }

 public function _subirFicheros() {...5 lines }

 private function _crearAnuncioPublico() {...10 lines }
```

```
private function _listarPublicacionAdministrativos() {...22 lines }

private function _publicaciones($array) {...13 lines }

private function _listarBusquedaPublicaciones() {...43 lines }

private function _listarAnuncios() {...7 lines }

private function _anuncios($array) {...12 lines }

private function _listarLineaAnuncio() {...9 lines }

private function _listarJornadas() {...9 lines }

 /**
 * Funciones para editar
 */
private function _controladorEditar() {...13 lines }

private function _lineaAnuncio() {...10 lines }

private function _editarAnuncio() {...19 lines }

private function _subirFicherosEditados() {...5 lines }

public function _subirFicherosPublicos() {...8 lines }

private function crearAutorizacionAnuncio() {...15 lines }

private function _crearPublicacion($autorizado, $id, $mail) {...6 lines }

private function _crearUsuario() {...8 lines }

private function _crearAlumnos() {...6 lines }

/**
 * Funciones para listar
 */
private function _controladorListar() {...43 lines }

private function _listarAnunciosAlumnos() {...9 lines }

private function _listarCategorias() {...10 lines }

private function _listarCategoriasAutorizar() {...10 lines }

private function _listarCategoriaSel() {...10 lines }

private function _listarCategoriaSelAdministrativos() {...11 lines }

private function _listarCategoriasPublico() {...8 lines }

private function _listarPublicacion() {...13 lines }
```

```
    /**
    * Funciones para eliminar
    */
    private function _controladorElimnar() {...10 lines }

    private function _eliminarAnuncio() {...8 lines }

    public function _getAccion() {...3 lines }

    public function _getControlador() {...3 lines }


}
```

ADMINISTRATIVOCONTROLLER: En esta clase se encuentra todos los métodos y funciones que utiliza el usuario administrativo

ALUMNOSCONTROLLER: En esta clase se encuentra todos los métodos y funciones que utiliza el usuario alumno

PROFESORESCONTROLLER: En esta clase se encuentra todos los métodos y funciones que utiliza el usuario profesor también el profesor que tiene privilegios de validación.

ANUNCIOSCONTROLLER: En esta clase se encuentra todos los métodos y funciones que utiliza los usuarios para crear los anuncios.

AUTORIZACIONESCONTROLLER: En esta clase se encuentra todos los métodos y funciones que utiliza el profesor validador para listar y validar los anuncios de los alumnos.

CATEGORIASCONTROLLER: En esta clase se encuentra todos los métodos y funciones que se utilizan para categorizar los distintos filtro que dispone los alumnos, profesores y alumnos validador para listar y validar los anuncios de los alumnos.

ERRORCONTROLLER: En esta clase se encuentra todos los métodos y funciones que se utilizan para mostrar la página de error 404 en caso de que una vista falle o un usuario intente ingresar una dirección URL no valida

IMPORTARCONTROLLER: En esta clase se encuentra todos los métodos y funciones que se utilizan para subir imágenes, pdf y listados de ficheros de alta de usuarios

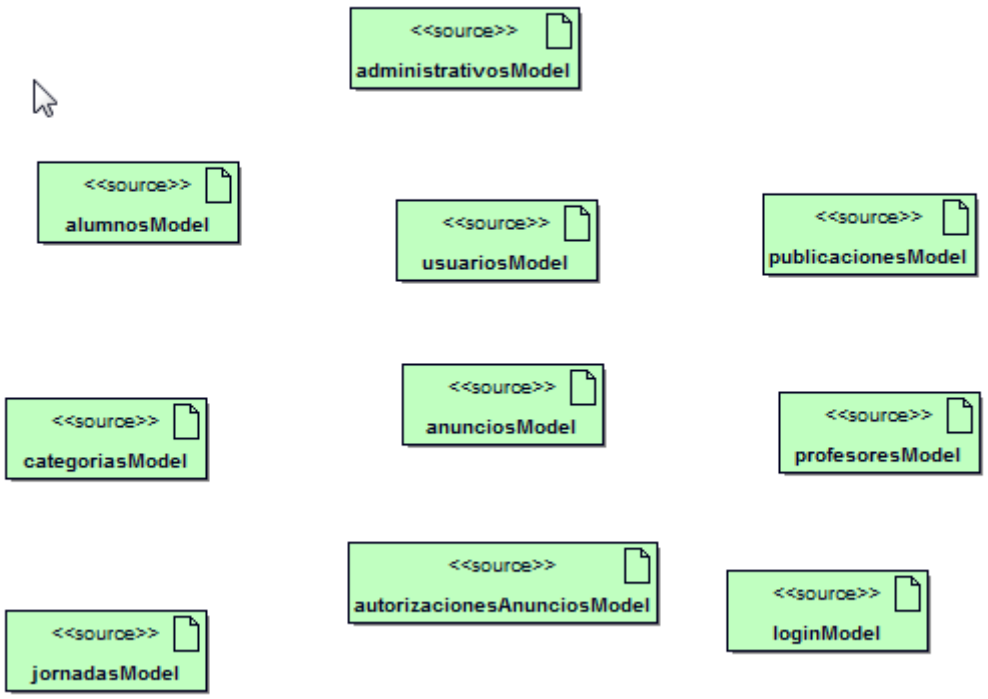
INDEXCONTROLLER: En esta clase se encuentra todos los métodos y funciones que se utilizan para el usuario anónimo y la portada que es visible para todos los usuarios públicos y privados

JORNADASCONTROLLER: En esta clase se encuentra todos los métodos y funciones que se utilizan para determinar la jornada del usuario conectado.

USUARIOSCONTROLLER: En esta clase se encuentra todos los métodos y funciones que son comunes en los datos de los usuarios que están registrados.

BLOQUE MODELS

En este bloque está la capa de datos todas clases que gestionan el acceso a los datos estas clases son las que heredan de la clase modelo que es en la cargada de gestionar la conectividad con la clase de conexión.



ADMINISTRATIVOMODELS: En esta clase se encuentra todas las funciones estáticas que utiliza el usuario administrativo para obtener los datos de la base de datos

ALUMNOSMODELS: En esta clase se encuentra todas las funciones estáticas que utiliza el usuario alumno para obtener los datos de la base de datos

PROFESORESMODELS: En esta clase se encuentra todas las funciones estáticas que utiliza el usuario profesor también el profesor que tiene privilegios de validación alumno para obtener los datos de la base de datos

ANUNCIOSMODELS: En esta clase se encuentra todas las funciones estáticas que utiliza para acceder a las tablas de la base de datos anuncios públicos y privados.

AUTORIZACIONESMODELS: En esta clase se encuentra todas las funciones estáticas que utiliza el profesor validador para obtener los datos de listar y validar los anuncios de los alumnos.

CATEGORIASMODELS: En esta clase se encuentra todos las funciones estáticas que obtiene los datos que se utilizan para categorizar los distintos filtro que dispone los alumnos, profesores y alumnos validador para listar y validar los anuncios de los alumnos.

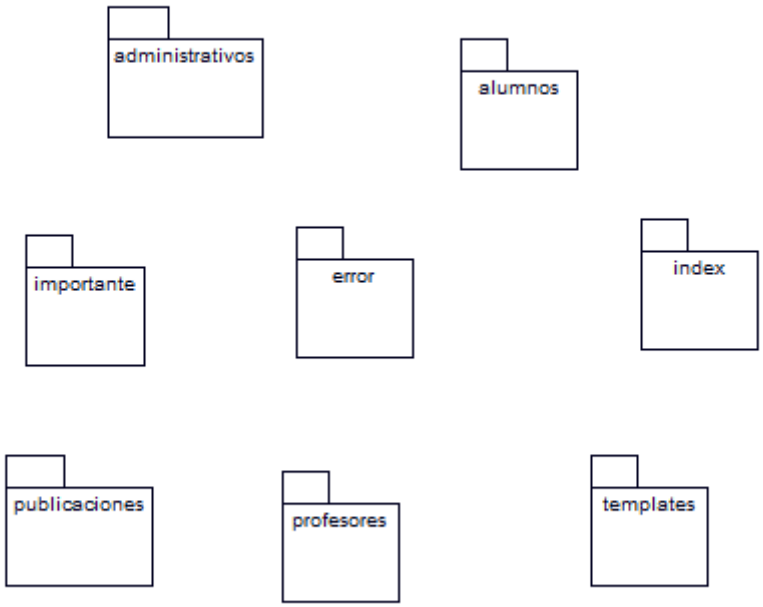
JORNADASMODELS: En esta clase se encuentra todas las funciones estáticas para acceder a la tabla de la base de datos jornadas.

USUARIOSMODELS: En esta clase se encuentra todas las funciones estáticas para acceder a la tabla de la base de datos usuarios.

PUBLICACIONESMODELS: En esta clase se encuentra todas las funciones estáticas para acceder a las tablas de la base de datos pública y privada.

PAQUETE WIEWS

En este paquete se encuentra todos los archivos de vista que se utilizan para visualizar la página web



ADMINISTRATIVOS: En esta carpeta están contenidas las vistas que visualiza el administrativo y que se comunican con el ajax.js

ALUMNOS: En esta carpeta están contenidas las vistas que visualiza el alumno y que se comunican con el ajax.js

PROFESORES: En esta carpeta están contenidas la vistas que visualiza los profesores y profesores validadores y que se comunican con el ajax.js

INDEX: En esta carpeta están contenida la vistas públicas que de la aplicación y que se comunica con el ajaxPublico.js.

PUBLICACIONES: En esta carpeta están contenidas las vistas de publicaciones de anuncio y que visualiza los usuarios privados con respecto a su tipo y categoría y que se comunica con el ajax.js.

ERROR: En esta carpeta están contenida la vista que visualiza el error 404

TEMPLATES: En esta carpeta están contenidas las plantillas que están por defecto para aplicación y que se mostrara aplicado a todas las páginas, también están los js que son comunes a las páginas de la aplicación

- Head: cabecera de las páginas de uso privado
- HeadPublico: cabecera de las páginas de uso publico
- Header: Es la parte donde se encuentra el logo, login y la opción de salir cuando se está logueado
- Nav: Es el menú que contiene los enlaces de uso privado
- NavPublico: : Es el menú que contiene los enlaces de uso publico
- Footer: Pie de página de la página web

PAQUETE LIBS

En este bloque están contenida las librerías que utiliza la aplicación

PAQUETE UPLOADS

En este bloque están contenidas todas las imágenes y pdf que contienen los anuncios de uso privado

PAQUETE UPLOADS PÚBLICO

En este bloque están contenida todas las imágenes y pdf que contienen los anuncios de uso publico

.HTACCESS

Este archivo es que se utiliza para aplicar las directivas y restricciones definidas antes de cursar la petición y, lógicamente, cancelar peticiones que se encuentren prohibidas dentro de este archivo, por lo que todas las variables que se envían vía get están condicionas por este archivo y no tenga conflicto cuando se envié la url