MANUAL DE TÉCNICO

REPOSITORIO DIGITAL DEL GOBIERNO AUTÓNOMO DEL CATÓN RIOBAMBA

Manual de Técnico

Para:

Sistema REDIGAD

Departamento de Gestión Documental

GADM RIOBAMBA

Versión 1.0

Preparado por:

Mayra Fernanda Ausay Espinoza **mausay.fis@unach.edu.ec**

Wilmer Estuardo Valle Padilla **wvalle.fis@unach.edu.ec**

Fecha: Riobamba - Marzo 2019

Contenido

TABLA DE ILUSTRACIONES

1. INTRODUCCIÓN

El Sistema REDIGAD Repositorio Digital del Gobierno Autónomo Descentralizado del Cantón Riobamba está compuesto de forma que permita que la información contenida pueda ser agregada, editada y/o modificada por el personal encargado del departamento.
Para alcanzar estos propósitos se ha hecho uso de PHP que es un lenguaje que se adecua a las nuevas necesidades de las aplicaciones web actuales, sin la necesidad de actualizaciones, a través de esta tecnología la administración y seguridad de la información se dará de forma centralizada y segura de datos, al mismo tiempo facilita la actualización eficiente de dicha información.
En cuanto a este manual se ha considerado incluir todos los aspectos técnicos necesarios para el manejo y control del sitio web para el Departamento de Gestión Documental.

2. OBJETIVO DE ESTE MANUAL

El objetivo primordial de este Manual es ayudar y guiar al técnico a informarse y utilizar herramientas para que el Sistema REDIGAD del Departamento de Gestión Documental entre en producción (ejecución), para de esa manera poder hacer uso de la información deseada para poder despejar todas las dudas existentes y para poder comprender:

- Guía para gestión de herramientas para poner en funcionamiento el sistema para el Departamento de Gestión Documental de Riobamba.
- Conocer cómo utilizar el sistema, mediante una descripción detallada e ilustrada de las opciones.
- Conocer el alcance de toda la información por medio de una explicación detallada e ilustrada de cada una de las páginas que lo conforman el manual técnico.

3. DIRIGIDO

Este manual está orientado a los técnicos u otros tipos de personal encargado del Departamento de Gestión Documental de Riobamba. Solamente dichas personas están autorizadas a realizar modificaciones en el sistema.
Una vez finalizado el proyecto, el Departamento de Tecnologías de la Información (Sistemas) del GADM Riobamba está encargado de definir políticas, normas, etc. para la administración del sistema REDIGAD.
También a través de este manual el personal podrá estar en la capacidad de supervisar el cumplimiento de políticas, normas, etc. que permitan el correcto funcionamiento de los Sistemas.
Definir conjuntamente con los departamentos pertinentes, los contenidos o cambios para el o los sistema para que también de igual forma puedan ser capacitados en herramientas necesarias para el mantenimiento y ejecución

4. CONOCIMIENTOS PREVIOS

Los conocimientos mínimos que deben tener las personas que operarán las páginas y deberán utilizar este manual son:

- Conocimientos básicos acerca de Programas Utilitarios
- Conocimientos básicos en adobe acrobat.
- Conocimiento básico de Internet
- Conocimiento básico de Windows

4.1. PASOS PARA ENCENDER LA COMPUTADORA

Encienda el C.P.U. presionar el botón Power
Encienda el monitor presionar el botón Power

Ilustración 1 Botón de Encendido

Fuente: http://www.cca.org.mx/profesores/abc/imagenes/m1/u2/t1/encendido.jpg

Espere mientras carga el Sistema Operativo. La apariencia de la pantalla mientras se carga el sistema es de un color negro y se aprecia la frase de iniciar Windows.

Automáticamente aparecerá la pantalla de Windows. La pantalla de Windows puede ser de varios tipos o diseño.

5. ESPECIFICACIONES TÉCNICAS

Para la implementación del Sistema REDIGAD para la preservación y validar la información obtenida en el Departamento de Gestión Documental del Municipio de Riobamba para lo cual requerimos lo siguiente:

5.1. HARDWARE

Cliente Requerido

El Software soporta Internet Explore8, Chrome, Mozilla, NetBeans 8.1., Xampp 1.8.2.0 VC9, y las siguientes versiones.

Servidor Windows

El sistema requiere básicamente todo lo que una instalación de Dominio Server requiere.

Se recomienda que se utilicen los requerimientos expuestos anteriormente para la mejor funcionalidad del Sistema REDIGAD.

5.2. SOFTWARE

- La Base de Datos requerida para el sistema es Domino – Notes
- El Software del Servidor es Domino Server, el sistema se puede administrar a través de la Web.

6. DESCRIPCIÓN DEL SISTEMA REDIGAD

El Sistema constará de toda la información preservada que se maneja en el Departamento de Gestión Documental, tales como: Resoluciones de Consejo, etc., contendrá una interfaz atractiva para los usuarios de este sistema, la cual será aprobada por el Departamento de Sistemas y el Departamento de Gestión Documental a cargo de la Lic. Victoria Muñoz, Directora del Departamento. En el menú superior encontrará las siguientes opciones:

Menú de Información:

- ✓ Mensajeria
- ✓ Notificaciones

- ✓ Tareas
- ✓ Configuración de Usuario
- ✓ Configuración General

Menú Lateral

- ✓ Cliente
- ✓ Usuario
- ✓ Administrador

7. DESCRIPCIÓN DE COTENIDO DEL SISTEMA REDIGAD

Para la realización de este Sistema REDIGAD se basó fundamentalmente en las necesidades que tenía el Departamento de Gestión Documental de Riobamba, aquí pueden preservar toda la información relacionada con su funcionamiento y descripción en general.
Una de los requerimientos del Sistema REDIGAD es que sea fácil de navegar para el usuario a través de su interfaz., es por este motivo que este sitio web tiene las siguientes características.

- ✓ Plataforma NETBEANS 8.1.
  - o Imágenes
  - o Módulos
  - o Plugins

- ✓ XAMPP

A continuación se tiene toda la información fundamental de cada uno de estos aspectos.

7.1. NETBEANS 8.1

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo, la plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

7.2. XAMPP

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl.
El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y MacOS X.

8. FUNCIONES PRINCIPALES DEL SISTEMA REDIGAD

Se requiere instalar los dos programas bases para la ejecución del sistema REDIGAD, entre ellos tenemos a NetBeans 8.1. y XAMPP 8.1.

Ilustración 2 Plataforma y servidor local para el desarrollo

9. CREACIÓN DEL SISTEMA EN NETBEANS.

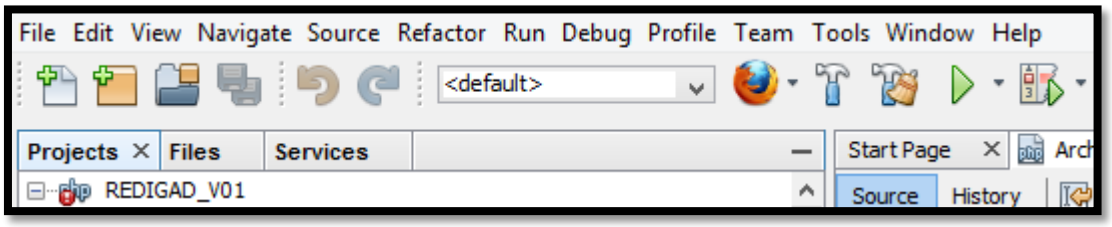

Ilustración 3 Creación del Proyecto en la plataforma NetBeans

10. CONEXIÓN DE BASE DE DATOS.

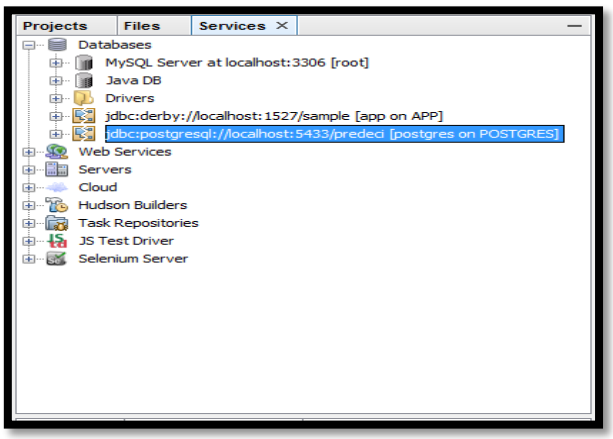

Ilustración 4 Conexión con la base de datos utilizada

11. CREACION DEL PATRÓN DEL MODELO VISTA CONTROLADOR

El Sistema REDIGAD está estructurado con un patrón de diseño de software utilizado para implementar sistemas donde se requiere el uso de interfaces de usuario. Se potencia la facilidad de mantenimiento, reutilización del código y la separación de conceptos.
Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad:

- Modelo.
- Vista.
- Controlador.

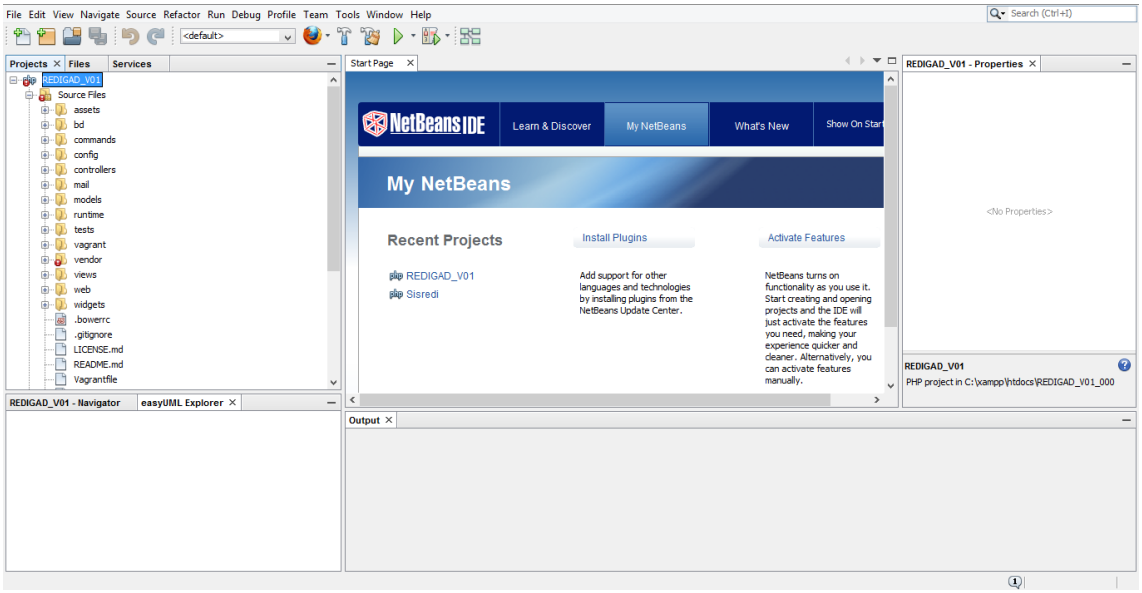

Ilustración 5 Estructura del MVC

12. CREACION DE CONTROLADORES

12.1. DEFINICIÓN:

Es el intermediario entre la Vista y el Modelo, se encarga de controlar las interacciones del usuario en la vista, contiene el código necesario para responder a las acciones que se solicitan en la aplicación.

Si se solicita desarrollar un sistema de administración de usuarios con un CRUD (Create, Read, Update, Delete).

- Pide los datos al modelo y los devuelve de nuevo a la vista para que ésta los muestre al usuario.
- Es decir las llamadas a clases y métodos, y los datos recibidos de formularios.
- La conexión entre la parte gráfica y los datos y los eventos que se producen cuando manejamos la aplicación.

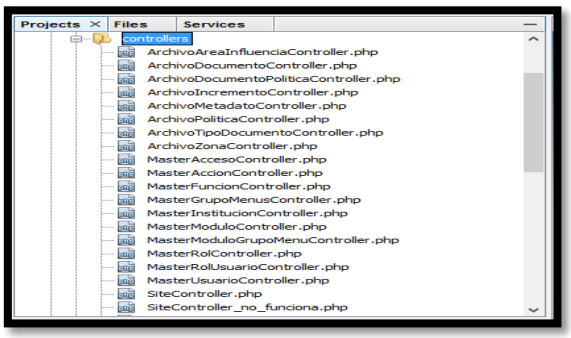

Ilustración 6 Controladores creados para el sistema REDIGAD

12.2. EJEMPLO DEL CONTROLADOR EN EL SISTEMA REDIGAD

Ilustración 7 Ejemplo del controlador de archivo documento

12.2.1. CÓDIGO:

```
<?php

namespace app\controllers;

use Yii;
use app\models\ArchivoDocumento;
use app\models\ArchivoDocumentoSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use app\models\MasterUsuario;
use yii\web\UploadedFile; //Para subir fotos

/**
 * ArchivoDocumentoController implements the CRUD actions for ArchivoDocumento model.
 */
class ArchivoDocumentoController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all ArchivoDocumento models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new ArchivoDocumentoSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single ArchivoDocumento model.
     * @param string $id
     * @return mixed
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new ArchivoDocumento model.
     * If creation is successful, the browser will be redirected to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new ArchivoDocumento();


         if (Yii::$app->request->isPost) {
```

```
            $model->load(Yii::$app->request->post());
            $model->url = UploadedFile::getInstance($model, 'url');


            if ($model->fecha_registro == '')
                $model->fecha_registro = date('Y-m-d H:i:s');

            if ($model->url && $model->validate()) {
                if ($model->url->saveAs('documentos/' . $model->url->baseName . '.' . $model->url->extension)) {
                    if ($model->save()) {
                        return $this->redirect(['view', 'id' => $model->id]);
                    } else {
                        echo 'Error! No puede redireccionar comuniquese con el administrador';
                    }
                } else {
                    echo 'Error! No puede guardar el documento';
                }
            } else {
                echo 'Error! El documento ' . $model->url . ' no se valida correctamente';
            }
        } else {
            ////////////////////////
            return $this->render('create', [
                    'model' => $model,
            ]);
            ////////////////////////
        }
//        if ($model->load(Yii::$app->request->post()) && $model->save()) {
//            return $this->redirect(['view', 'id' => $model->id]);
//        } else {
//            return $this->render('create', [
//                'model' => $model,
//            ]);
//        }
    }

    /**
     * Updates an existing ArchivoDocumento model.
     * If update is successful, the browser will be redirected to the 'view' page.
     * @param string $id
     * @return mixed
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
        $aux_url_anterior = $model->url;
//        if ($model->load(Yii::$app->request->post()) && $model->save()) {
//            return $this->redirect(['view', 'id' => $model->id]);
//        } else {
//            return $this->render('update', [
//                'model' => $model,
//            ]);
//        }
        if (Yii::$app->request->isPost) {
            $model->load(Yii::$app->request->post());
            $model->url = UploadedFile::getInstance($model, 'url');

            if ($model->url && $model->validate()) {
                if ($model->url->saveAs('documentos/' . $model->url->baseName . '.' . $model->url->extension)) {
                    if ($model->save()) {
                        return $this->redirect(['view', 'id' => $model->id]);
                        //echo "<script>javascript:history.go(-2)</script>";
                    } else {
                        echo 'Error! No puede redireccionar comuniquese con el administrador';
                    }
                } else {
                    echo 'Error! No puede el documento';
                }
            } else {
                //  echo 'Error! El archivo ' . $model->imagen_producto . ' No se valida correctamente';
                //Se guarda de igual manera
                $model->url = $aux_url_anterior;
                if ($model->save()) {
                    return $this->redirect(['view', 'id' => $model->id]);
                    //echo "<script>javascript:history.go(-2)</script>";
                }
```

```
            }
        } else {
            ////////////////////////////
            return $this->render('update', [
                    'model' => $model,
            ]);
            ////////////////////////////
        }
    }

    /**
     * Deletes an existing ArchivoDocumento model.
     * If deletion is successful, the browser will be redirected to the 'index' page.
     * @param string $id
     * @return mixed
     */
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    /**
     * Finds the ArchivoDocumento model based on its primary key value.
     * If the model is not found, a 404 HTTP exception will be thrown.
     * @param string $id
     * @return ArchivoDocumento the loaded model
     * @throws NotFoundHttpException if the model cannot be found
     */
    protected function findModel($id)
    {
        if (($model = ArchivoDocumento::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }

}
```

13. INGRESO A MODELO

13.1 DEFINICIÓN

Contiene la lógica de la aplicación., son las clases y métodos que se comunican directamente con la base de datos, todas las funciones que accederán a las tablas y harán los correspondientes SELECT, UPDATE, INSERT, DELETE.

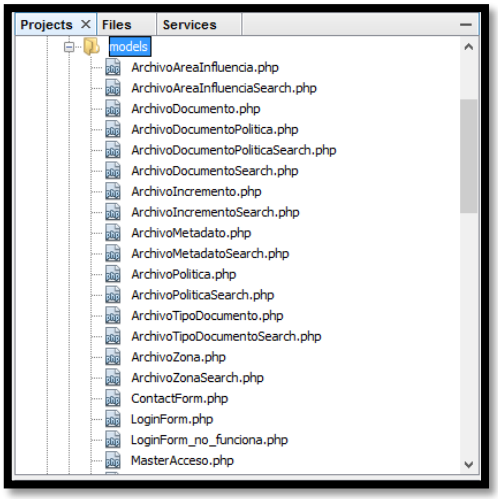

Ilustración 8 Modelos del sistema REDIGAD

13.2. EJEMPLO DEL MODELO EN EL SISTEMA REDIGAD

Ilustración 9 Ejemplo dl modelo del sistema REDIGAD

13.2.1. CÓDIGO:

```
<?php

namespace app\models;

use Yii;

/**
 * This is the model class for table "archivo_documento".
 *
 * @property string $id
 * @property string $id_tipo_documento
 * @property string $id_area_influencia
 * @property string $titulo
 * @property string $descripcion
 * @property string $palabra_clave
 * @property string $editor
 * @property string $lenguaje
 * @property string $id_documento_padre
 * @property string $autor_creador
 * @property string $url
 * @property string $fecha_registro
 * @property string $fecha_validado
 * @property integer $validado
 * @property integer $estado
 *
 * @property ArchivoTipoDocumento $idTipoDocumento
 * @property ArchivoAreaInfluencia $idAreaInfluencia
 * @property ArchivoDocumentoPolitica[] $archivoDocumentoPoliticas
 * @property ArchivoIncremento[] $archivoIncrementos
 * @property ArchivoMetadato[] $archivoMetadatos
 */
class ArchivoDocumento extends \yii\db\ActiveRecord
{
    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'archivo_documento';
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['id_tipo_documento', 'id_area_influencia'], 'required'],
            [['id_tipo_documento', 'id_area_influencia', 'id_documento_padre', 'validado', 'estado'], 'integer'],
            [['fecha_registro', 'fecha_validado'], 'safe'],
            [['titulo'], 'string', 'max' => 300],
             // Para subir archivo
            //  [['imagen_producto'], 'file', 'skipOnEmpty' => false, 'extensions' => 'pdf, png, jpg, rar, xlsx, docx'],
            //[['imagen_producto'], 'file', 'extensions' => 'png, jpg, jpeg, gif'],
            [['url'], 'file'],
            //Fin para subir archivo
            [['descripcion'], 'string', 'max' => 5000],
            [['palabra_clave', 'autor_creador'], 'string', 'max' => 100],
            [['editor'], 'string', 'max' => 200],
            [['lenguaje'], 'string', 'max' => 2],
            [['id_tipo_documento'], 'exist', 'skipOnError' => true, 'targetClass' => ArchivoTipoDocumento::className(),
'targetAttribute' => ['id_tipo_documento' => 'id']],
```

```
            [['id_area_influencia'], 'exist', 'skipOnError' => true, 'targetClass' => ArchivoAreaInfluencia::className(),
'targetAttribute' => ['id_area_influencia' => 'id']],
        ];
    }

    /**
     * @inheritdoc
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'id_tipo_documento' => 'Archivo de tipo',
            'id_area_influencia' => 'Area de influencia',
            'titulo' => 'Título',
            'descripcion' => 'Descripción',
            'palabra_clave' => 'Palabra clave',
            'editor' => 'Editor',
            'lenguaje' => 'Lenguaje',
            'id_documento_padre' => 'Documento padre',
            'autor_creador' => 'Autor/creador',
            'url' => 'URL',
            'fecha_registro' => 'Fecha de registro',
            'fecha_validado' => 'Fecha que fue validado',
            'validado' => 'Validado',
            'estado' => 'Estado',
        ];
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getIdTipoDocumento()
    {
        return $this->hasOne(ArchivoTipoDocumento::className(), ['id' => 'id_tipo_documento']);
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getIdAreaInfluencia()
    {
        return $this->hasOne(ArchivoAreaInfluencia::className(), ['id' => 'id_area_influencia']);
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getArchivoDocumentoPoliticas()
    {
        return $this->hasMany(ArchivoDocumentoPolitica::className(), ['id_documento' => 'id']);
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getArchivoIncrementos()
    {
        return $this->hasMany(ArchivoIncremento::className(), ['id_documento' => 'id']);
    }

    /**
     * @return \yii\db\ActiveQuery
     */
    public function getArchivoMetadatos()
    {
        return $this->hasMany(ArchivoMetadato::className(), ['id_documento' => 'id']);
    }
}
```

14. INGRESO A VISTAS

14.1. DEFINICIÓN

Programaremos lo relativo a la interfaz gráfica, muestra la información al usuario e interactúa con él. Las vistas requerirán los datos a los modelos y ellas se generarán las salidas, tal como nuestra aplicación requiera.

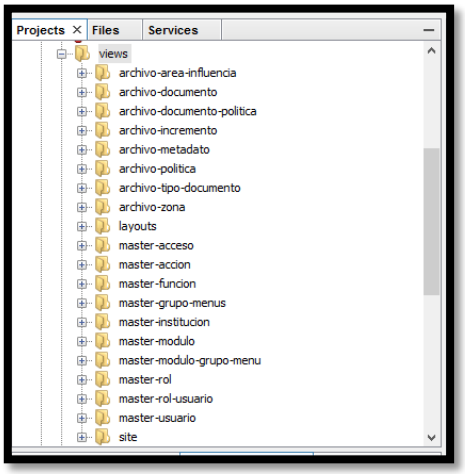

Ilustración 10 Vistas de sistema REDIGAD

14.2. EJEMPLO DE VISTA EN EL SISTEMA REDIGAD

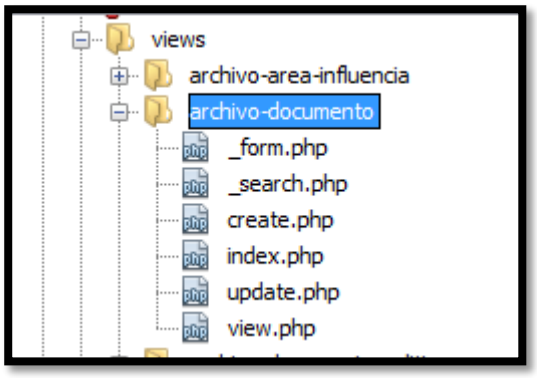

Ilustración 11 Ejemplo de vista del sistema REDIGAD

14.2.1. CÓDIGO DE INDEX:

```
<?php

use yii\helpers\Html;
use yii\grid\GridView;

/* @var $this yii\web\View */
/* @var $searchModel app\models\ArchivoDocumentoSearch */
/* @var $dataProvider yii\data\ActiveDataProvider */

$this->title = 'Documentos';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="archivo-documento-index">

    <h1><?= Html::encode($this->title) ?></h1>
    <?php // echo $this->render('_search', ['model' => $searchModel]); ?>

    <p>
```

```
        <?= Html::a('Nuevo Documento', ['create'], ['class' => 'btn btn-success']) ?>
    </p>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'filterModel' => $searchModel,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],

            'id',
            //'id_tipo_documento',
              [
                'label' => 'Tipo de documento',
                'attribute' => 'id_tipo_documento',
                'value' => 'idTipoDocumento.tipo_documento',
            ],
            //'id_area_influencia',
              [
                'label' => 'Área de influencia',
                'attribute' => 'id_area_influencia',
                'value' => 'idAreaInfluencia.area_influencia',
            ],
            'titulo',
            'descripcion',
            // 'palabra_clave',
            // 'editor',
            // 'lenguaje',
            // 'id_documento_padre',
            // 'autor_creador',
            // 'url:url',
            // 'fecha_registro',
            // 'fecha_validado',
            // 'validado',
            // 'estado',
            [
                'attribute' => 'estado',
                'format' => 'html',
                'label' => 'Estado',
                'value' => function ($data) {
                    return Html::img('@web/imagenes/' . $data['estado'] . '.png', ['width' => '40px',
                            'height' => '40px']);
                },
            ],
            ['class' => 'yii\grid\ActionColumn'],
        ],
    ]); ?>
</div>
```

15. DISTRIBUCIÓN DE ROLES DE USUARIO

Para la programación de este aplicativo se definieron 3 tipos de usuarios o roles donde cada uno interactúa con el sistema de diferentes modos. En la figura 1 se ilustra los roles y casos de uso que realiza cada usuario dentro del sistema.

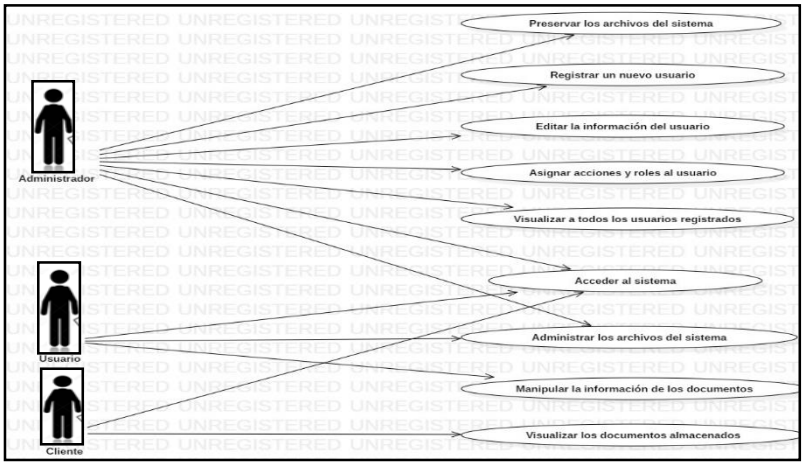

Ilustración 12 Diagrama General de Caso de Uso

En el sistema encontramos 3 tipos de usuarios con son: administrador, usuario y cliente.

Administrador: Este usuario realiza tareas de preservar los archivos del sistema, registrar un nuevo usuario, editar información del usuario, asignar acciones y roles al usuario, visualizar a todos los usuarios registrados, acceder al sistema, administrar los archivos del sistema

Usuario: Este usuario realiza tareas de acceder al sistema, administrar los archivos del sistema, manipula información de los documentos

Cliente: este usuario realiza tareas de acceder al sistema y visualiza los documentos almacenados**.**

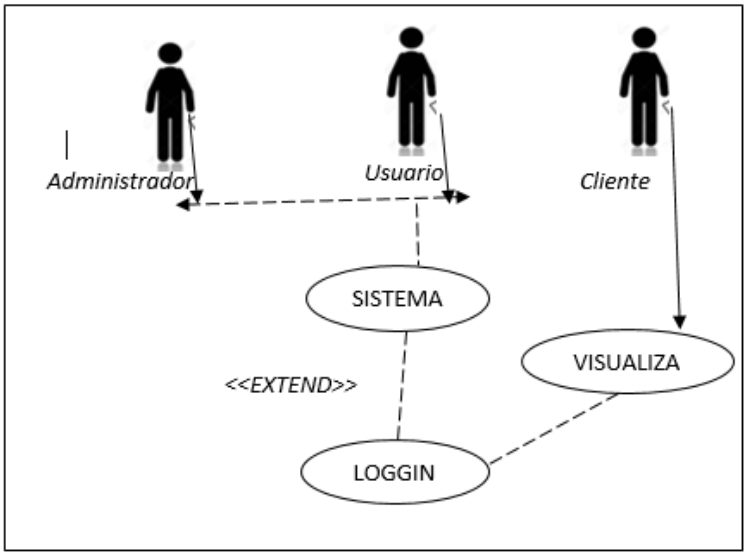

Ilustración 13 Ingresos ala sistema de acuerdo al rol