UNIVERSIDAD AUTONOMA METROPOLITANA

AZCAPOTZALCO

Proyecto Terminal

SISTEMA DE GESTIÓN DE CURRICULA DE PERSONAL ACADÉMICO

David Israel Marroquín Magaña
Alumno

206309218
Matricula

M en C. Rafaela Blanca Silva López
Asesor

INDICE

Tabla de contenido

OBJETIVO GENERAL

Diseñar e implementar un sistema de gestión de currícula de personal académico así como sus actividades de vinculación, investigación, preservación y difusión de la cultura.

OBJETIVOS PARTICULARES

- Diseñar e implementar la base de datos del sistema.
- Diseñar e implementar el módulo de registros del personal académico así como sus actividades de vinculación, investigación, preservación y difusión de la cultura.
- Diseñar e implementar el módulo de consultas de actividades de vinculación, investigación, preservación y difusión de la cultura del personal académico.
- Diseñar e implementar el módulo de generación de reportes del sistema de gestión de currícula del personal académico.

INTRODUCCION

El objetivo de este proyecto tiene como finalidad dar solución a un problema especifico que esta presente en la UAM-A en el departamento de Sistemas de la División de Ciencias Básicas e Ingeniería mediante el desarrollo e implementación de un sistema web que lleva el control del perfil curricula del personal académico así como sus diversas actividades de vinculación, investigación, preservación y difusión de la cultura del departamento.

.En el departamento de sistemas de la UAM Azcapotzalco mucha de la información que se maneja del personal académico esta en hojas de papel lo que genera problemas para localizar rápidamente cierta información, generar estadísticas o encontrar puntos de intersección.

El proyecto terminal aquí descrito permite al departamento de sistemas de la UAM Azcapotzalco tener un control electrónico de la curricular de actividades del personal académico que labora en el departamento de sistemas, que se podrá accesar desde cualquier lugar.

Si se requiere resolver un problema, es complicado determinar rápidamente al personal ideal para llevarlo a cabo. La identificación de especialistas en ciertos temas es fundamental para la creación de nuevos colectivos de trabajo y nuevos proyectos de investigación.

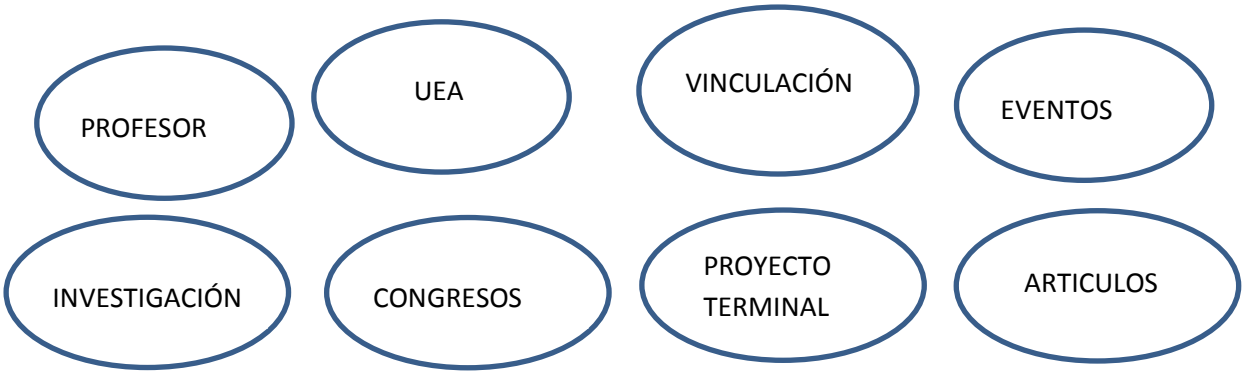
Otro problema que se presenta es identificar la idoneidad de la asignación de carga docente con base en su expertis, para que enriquezca el proceso enseñanza-aprendizaje.

Por ello, una de las formas de resolver estos problemas, fue con el desarrollo del sistema de gestión de currícula que se presenta a continuación, el cual contiene diferentes módulos que permiten guardar toda la información del personal docente.

Existen en la actualidad diversos sistemas de gestión de currícula, pero el objetivo de este proyecto es que se puede acoplar a las necesidades específicas de gestión del personal académico del departamento de sistemas de la UAM Azcapotzalco y que en un futuro se pueda implementar en los demás departamentos de la división.

Definición de la Estructura de la Base de Datos

ENTIDAD DE LA BASE DE DATOS:



En la Figura 1 Se muestra la tabla de Verbos del Diseño de la Base de Datos de Sistema

	PROFESOR	UEA	VINCULACIÓN	EVENTOS	INVEST.	CONGRESOS	ARTICULOS	P. T.
PROFESOR	X	Imparte	Realiza	participa	realiza	partica		asesora
UEA	Es impartida	X						pertenece
VINCULACION	realiza		X				realiza	
EVENTOS				X				
INVESTIGACIÓN	Es realizada	X			X			contiene
CONGRESOS	participa	X				X		
ARTICULOS	pertenece				contienen		X	
P. T	asesora	pertenece			contienen			X

Figura 1 Tabla de Verbos

Diagrama de Entidad-Relación

Figura 2 Diagrama Entidad Relación

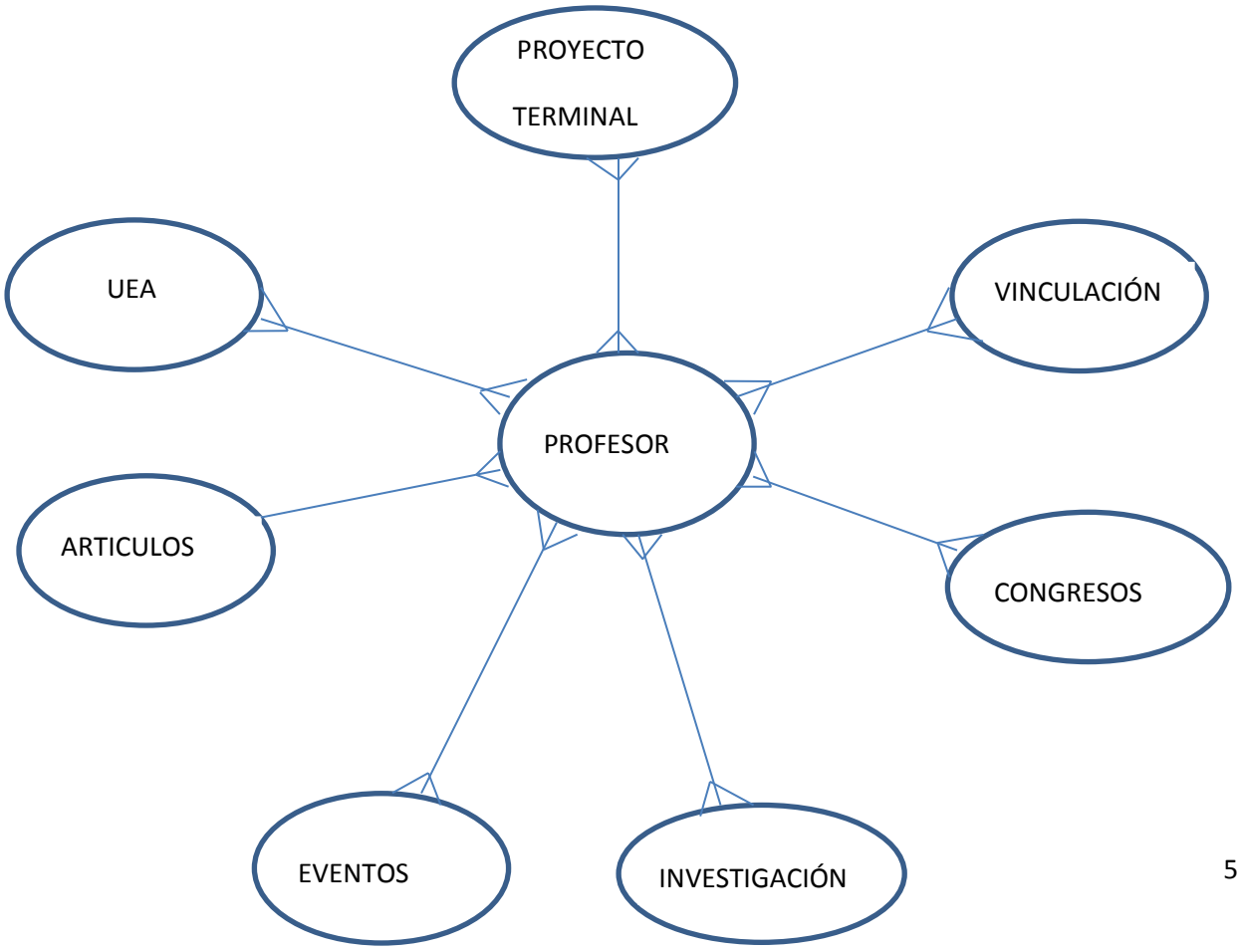


Diagrama de la Base de Datos

La Figura 3 Muestra el Diagrama Entidad-Relación Normalizado

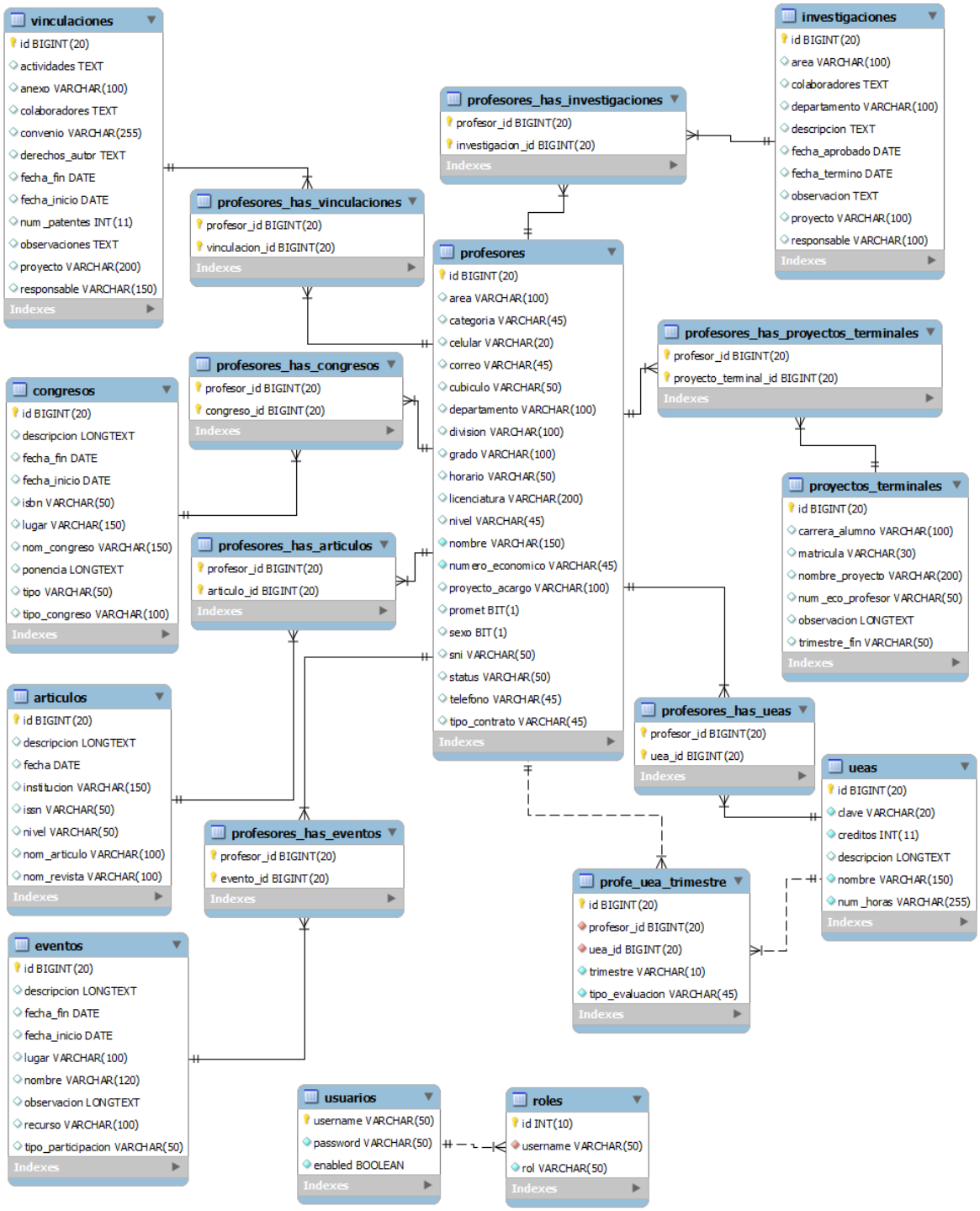


Figura 3 Diagrama Entidad Relacion Normalizada

ESQUEMA FÍSICO DE LA BASE DE DATOS

```
CREATE DATABASE  IF NOT EXISTS `profesores_db` /*!40100 DEFAULT
CHARACTER SET utf8 */;
USE `profesores_db`;
-- MySQL dump 10.13  Distrib 5.5.16, for Win32 (x86)
--
-- Host: localhost    Database: profesores_db
-- ------------------------------------------------------
-- Server version     5.5.29

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `articulos`
--

DROP TABLE IF EXISTS `articulos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `articulos` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `descripcion` longtext,
  `fecha` date DEFAULT NULL,
  `institucion` varchar(150) DEFAULT NULL,
  `issn` varchar(50) DEFAULT NULL,
  `nivel` varchar(50) DEFAULT NULL,
  `nom_articulo` varchar(100) DEFAULT NULL,
  `nom_revista` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `articulos`
--

LOCK TABLES `articulos` WRITE;
```

```
/*!40000 ALTER TABLE `articulos` DISABLE KEYS */;
/*!40000 ALTER TABLE `articulos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `congresos`
--

DROP TABLE IF EXISTS `congresos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `congresos` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `descripcion` longtext,
  `fecha_fin` date DEFAULT NULL,
  `fecha_inicio` date DEFAULT NULL,
  `isbn` varchar(50) DEFAULT NULL,
  `lugar` varchar(150) DEFAULT NULL,
  `nom_congreso` varchar(150) DEFAULT NULL,
  `ponencia` longtext,
  `tipo` varchar(50) DEFAULT NULL,
  `tipo_congreso` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `congresos`
--

LOCK TABLES `congresos` WRITE;
/*!40000 ALTER TABLE `congresos` DISABLE KEYS */;
/*!40000 ALTER TABLE `congresos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `eventos`
--

DROP TABLE IF EXISTS `eventos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `eventos` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `descripcion` longtext,
  `fecha_fin` date DEFAULT NULL,
  `fecha_inicio` date DEFAULT NULL,
  `lugar` varchar(100) DEFAULT NULL,
  `nombre` varchar(120) DEFAULT NULL,
  `observacion` longtext,
  `recurso` varchar(100) DEFAULT NULL,
```

```
  `tipo_participacion` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `eventos`
--

LOCK TABLES `eventos` WRITE;
/*!40000 ALTER TABLE `eventos` DISABLE KEYS */;
/*!40000 ALTER TABLE `eventos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `investigaciones`
--

DROP TABLE IF EXISTS `investigaciones`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `investigaciones` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `area` varchar(100) DEFAULT NULL,
  `colaboradores` longtext,
  `departamento` varchar(100) DEFAULT NULL,
  `descripcion` longtext,
  `fecha_aprobado` date DEFAULT NULL,
  `fecha_termino` date DEFAULT NULL,
  `observacion` longtext,
  `proyecto` varchar(100) DEFAULT NULL,
  `responsable` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `investigaciones`
--

LOCK TABLES `investigaciones` WRITE;
/*!40000 ALTER TABLE `investigaciones` DISABLE KEYS */;
/*!40000 ALTER TABLE `investigaciones` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profe_uea_trimestre`
--

DROP TABLE IF EXISTS `profe_uea_trimestre`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profe_uea_trimestre` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `tipo_evaluacion` varchar(45) NOT NULL,
  `trimestre` varchar(10) NOT NULL,
  `profesor_id` bigint(20) NOT NULL,
  `uea_id` bigint(20) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`),
  KEY `FKA282E99689F2E50D` (`uea_id`),
  KEY `FKA282E9963DBA0067` (`profesor_id`),
  CONSTRAINT `FKA282E9963DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FKA282E99689F2E50D` FOREIGN KEY (`uea_id`) REFERENCES
`ueas` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profe_uea_trimestre`
--

LOCK TABLES `profe_uea_trimestre` WRITE;
/*!40000 ALTER TABLE `profe_uea_trimestre` DISABLE KEYS */;
/*!40000 ALTER TABLE `profe_uea_trimestre` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profesores`
--

DROP TABLE IF EXISTS `profesores`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `area` varchar(100) DEFAULT NULL,
  `categoria` varchar(45) DEFAULT NULL,
  `celular` varchar(20) DEFAULT NULL,
  `correo` varchar(45) DEFAULT NULL,
  `cubiculo` varchar(50) DEFAULT NULL,
  `departamento` varchar(100) DEFAULT NULL,
  `division` varchar(100) DEFAULT NULL,
  `grado` varchar(100) DEFAULT NULL,
  `horario` varchar(50) DEFAULT NULL,
  `licenciatura` varchar(200) DEFAULT NULL,
  `nivel` varchar(45) DEFAULT NULL,
  `nombre` varchar(150) NOT NULL,
  `numero_economico` varchar(45) NOT NULL,
  `promet` bit(1) DEFAULT NULL,
  `proyecto_acargo` varchar(100) DEFAULT NULL,
```

```
  `sexo` bit(1) DEFAULT NULL,
  `sni` varchar(50) DEFAULT NULL,
  `status` varchar(50) DEFAULT NULL,
  `telefono` varchar(45) DEFAULT NULL,
  `tipo_contrato` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`),
  UNIQUE KEY `numero_economico` (`numero_economico`),
  UNIQUE KEY `numero_economico_2` (`numero_economico`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profesores`
--

LOCK TABLES `profesores` WRITE;
/*!40000 ALTER TABLE `profesores` DISABLE KEYS */;
/*!40000 ALTER TABLE `profesores` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profesores_has_articulos`
--

DROP TABLE IF EXISTS `profesores_has_articulos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores_has_articulos` (
  `profesor_id` bigint(20) NOT NULL,
  `articulo_id` bigint(20) NOT NULL,
  PRIMARY KEY (`articulo_id`,`profesor_id`),
  KEY `FK394C33AC2D3ECEC7` (`articulo_id`),
  KEY `FK394C33AC3DBA0067` (`profesor_id`),
  CONSTRAINT `FK394C33AC3DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FK394C33AC2D3ECEC7` FOREIGN KEY (`articulo_id`)
REFERENCES `articulos` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profesores_has_articulos`
--

LOCK TABLES `profesores_has_articulos` WRITE;
/*!40000 ALTER TABLE `profesores_has_articulos` DISABLE KEYS */;
/*!40000 ALTER TABLE `profesores_has_articulos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profesores_has_congresos`
```

```
--

DROP TABLE IF EXISTS `profesores_has_congresos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores_has_congresos` (
  `profesor_id` bigint(20) NOT NULL,
  `congreso_id` bigint(20) NOT NULL,
  PRIMARY KEY (`congreso_id`,`profesor_id`),
  KEY `FKEA2EAB33561E7C27` (`congreso_id`),
  KEY `FKEA2EAB333DBA0067` (`profesor_id`),
  CONSTRAINT `FKEA2EAB333DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FKEA2EAB33561E7C27` FOREIGN KEY (`congreso_id`)
REFERENCES `congresos` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profesores_has_congresos`
--

LOCK TABLES `profesores_has_congresos` WRITE;
/*!40000 ALTER TABLE `profesores_has_congresos` DISABLE KEYS */;
/*!40000 ALTER TABLE `profesores_has_congresos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profesores_has_eventos`
--

DROP TABLE IF EXISTS `profesores_has_eventos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores_has_eventos` (
  `profesor_id` bigint(20) NOT NULL,
  `evento_id` bigint(20) NOT NULL,
  PRIMARY KEY (`evento_id`,`profesor_id`),
  KEY `FK71C66BB2B61283C7` (`evento_id`),
  KEY `FK71C66BB23DBA0067` (`profesor_id`),
  CONSTRAINT `FK71C66BB23DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FK71C66BB2B61283C7` FOREIGN KEY (`evento_id`) REFERENCES
`eventos` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profesores_has_eventos`
--

LOCK TABLES `profesores_has_eventos` WRITE;
```

```
/*!40000 ALTER TABLE `profesores_has_eventos` DISABLE KEYS */;
/*!40000 ALTER TABLE `profesores_has_eventos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profesores_has_investigaciones`
--

DROP TABLE IF EXISTS `profesores_has_investigaciones`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores_has_investigaciones` (
  `profesor_id` bigint(20) NOT NULL,
  `investigacion_id` bigint(20) NOT NULL,
  PRIMARY KEY (`investigacion_id`,`profesor_id`),
  KEY `FKD353CA55DE28CE0D` (`investigacion_id`),
  KEY `FKD353CA553DBA0067` (`profesor_id`),
  CONSTRAINT `FKD353CA553DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FKD353CA55DE28CE0D` FOREIGN KEY (`investigacion_id`)
REFERENCES `investigaciones` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profesores_has_investigaciones`
--

LOCK TABLES `profesores_has_investigaciones` WRITE;
/*!40000 ALTER TABLE `profesores_has_investigaciones` DISABLE KEYS */;
/*!40000 ALTER TABLE `profesores_has_investigaciones` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profesores_has_proyectos_terminales`
--

DROP TABLE IF EXISTS `profesores_has_proyectos_terminales`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores_has_proyectos_terminales` (
  `proyecto_terminal_id` bigint(20) NOT NULL,
  `profesor_id` bigint(20) NOT NULL,
  PRIMARY KEY (`profesor_id`,`proyecto_terminal_id`),
  KEY `FK840E3C76C222212` (`proyecto_terminal_id`),
  KEY `FK840E3C73DBA0067` (`profesor_id`),
  CONSTRAINT `FK840E3C73DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FK840E3C76C222212` FOREIGN KEY (`proyecto_terminal_id`)
REFERENCES `proyectos_terminales` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
-- Dumping data for table `profesores_has_proyectos_terminales`
--

LOCK TABLES `profesores_has_proyectos_terminales` WRITE;
/*!40000 ALTER TABLE `profesores_has_proyectos_terminales` DISABLE
KEYS */;
/*!40000 ALTER TABLE `profesores_has_proyectos_terminales` ENABLE KEYS
*/;
UNLOCK TABLES;

--
-- Table structure for table `profesores_has_ueas`
--

DROP TABLE IF EXISTS `profesores_has_ueas`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores_has_ueas` (
  `uea_id` bigint(20) NOT NULL,
  `profesor_id` bigint(20) NOT NULL,
  PRIMARY KEY (`profesor_id`,`uea_id`),
  KEY `FKA868098E89F2E50D` (`uea_id`),
  KEY `FKA868098E3DBA0067` (`profesor_id`),
  CONSTRAINT `FKA868098E3DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FKA868098E89F2E50D` FOREIGN KEY (`uea_id`) REFERENCES
`ueas` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profesores_has_ueas`
--

LOCK TABLES `profesores_has_ueas` WRITE;
/*!40000 ALTER TABLE `profesores_has_ueas` DISABLE KEYS */;
/*!40000 ALTER TABLE `profesores_has_ueas` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `profesores_has_vinculaciones`
--

DROP TABLE IF EXISTS `profesores_has_vinculaciones`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `profesores_has_vinculaciones` (
  `vinculacion_id` bigint(20) NOT NULL,
  `profesor_id` bigint(20) NOT NULL,
  PRIMARY KEY (`profesor_id`,`vinculacion_id`),
```

```
  KEY `FKB0C19692F48364D` (`vinculacion_id`),
  KEY `FKB0C19693DBA0067` (`profesor_id`),
  CONSTRAINT `FKB0C19693DBA0067` FOREIGN KEY (`profesor_id`)
REFERENCES `profesores` (`id`),
  CONSTRAINT `FKB0C19692F48364D` FOREIGN KEY (`vinculacion_id`)
REFERENCES `vinculaciones` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `profesores_has_vinculaciones`
--

LOCK TABLES `profesores_has_vinculaciones` WRITE;
/*!40000 ALTER TABLE `profesores_has_vinculaciones` DISABLE KEYS */;
/*!40000 ALTER TABLE `profesores_has_vinculaciones` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `proyectos_terminales`
--

DROP TABLE IF EXISTS `proyectos_terminales`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `proyectos_terminales` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `carrera_alumno` varchar(100) DEFAULT NULL,
  `matricula` varchar(30) DEFAULT NULL,
  `nombre_proyecto` varchar(200) DEFAULT NULL,
  `num_eco_profesor` varchar(50) DEFAULT NULL,
  `observacion` longtext,
  `trimestre_fin` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `proyectos_terminales`
--

LOCK TABLES `proyectos_terminales` WRITE;
/*!40000 ALTER TABLE `proyectos_terminales` DISABLE KEYS */;
/*!40000 ALTER TABLE `proyectos_terminales` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `ueas`
--

DROP TABLE IF EXISTS `ueas`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `ueas` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `clave` varchar(20) NOT NULL,
  `creditos` int(11) NOT NULL,
  `descripcion` longtext,
  `nombre` varchar(150) NOT NULL,
  `num_horas` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `ueas`
--

LOCK TABLES `ueas` WRITE;
/*!40000 ALTER TABLE `ueas` DISABLE KEYS */;
/*!40000 ALTER TABLE `ueas` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `vinculaciones`
--

DROP TABLE IF EXISTS `vinculaciones`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `vinculaciones` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `actividades` longtext,
  `anexo` varchar(100) DEFAULT NULL,
  `colaboradores` longtext,
  `convenio` varchar(255) DEFAULT NULL,
  `derechos_autor` longtext,
  `fecha_fin` date DEFAULT NULL,
  `fecha_inicio` date DEFAULT NULL,
  `num_patentes` int(11) DEFAULT NULL,
  `observaciones` varchar(255) DEFAULT NULL,
  `proyecto` varchar(200) DEFAULT NULL,
  `responsable` varchar(150) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `vinculaciones`
--
```

```
LOCK TABLES `vinculaciones` WRITE;
/*!40000 ALTER TABLE `vinculaciones` DISABLE KEYS */;
/*!40000 ALTER TABLE `vinculaciones` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2013-01-07  0:34:56
```

Clases de Sistema

En la Figura 4. se muestra las clases principales del sistema con sus respectivos atributos que contiene el sistema.

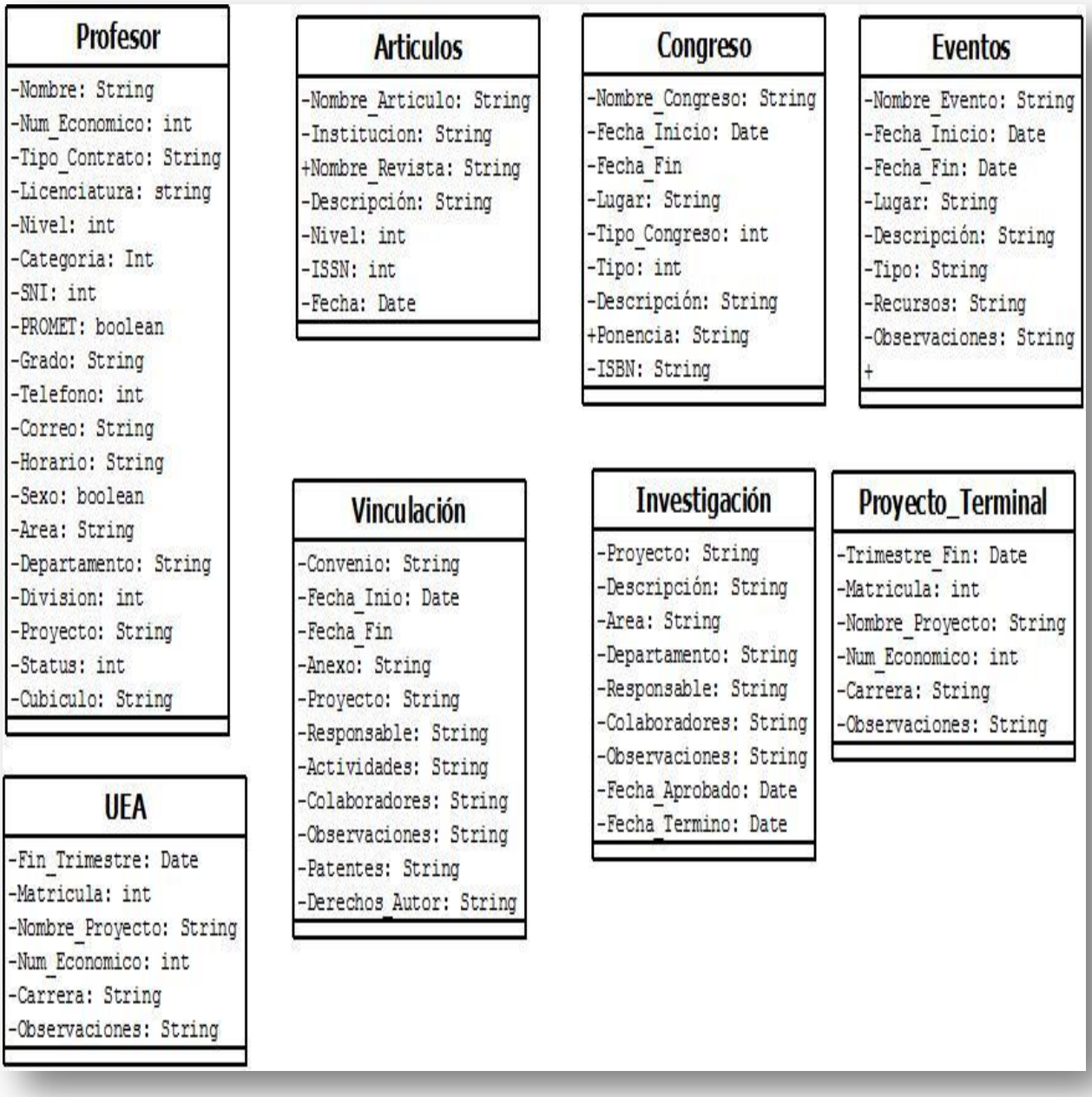


Figura 4 Diagrama de Clases

En la Figura 5 se muestra el diagrama de relacion entre clases dentro del sistema

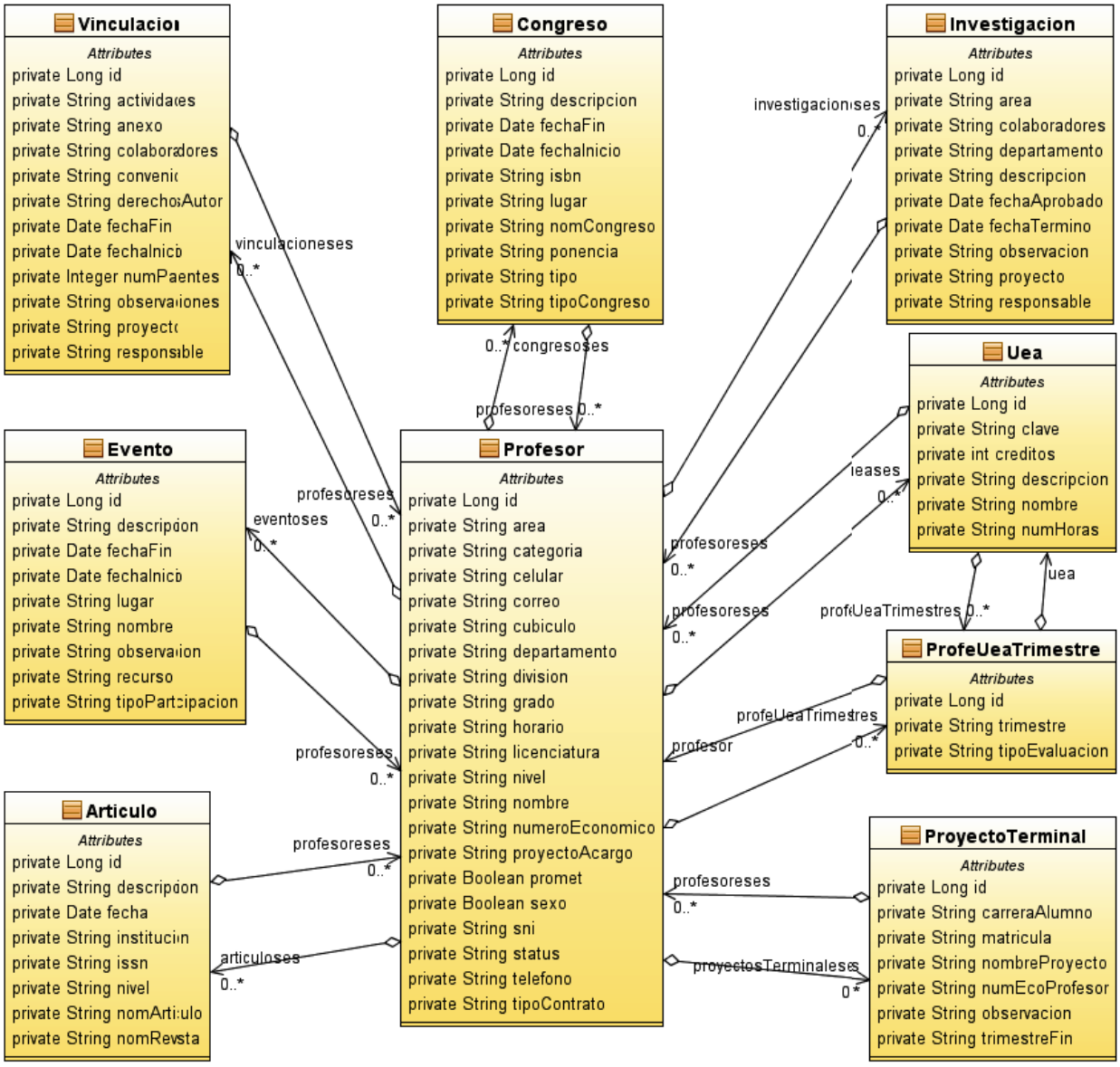


Figura 5 Diagrama de Asociacion de Clases

Diseño de la Interface

El diseño de la Interfaz de usuario se divide en 2 partes la parte principal en la cual el usuario ingresa al sistema mediante un Usuario y Contraseña proporcionada por el Administrador.

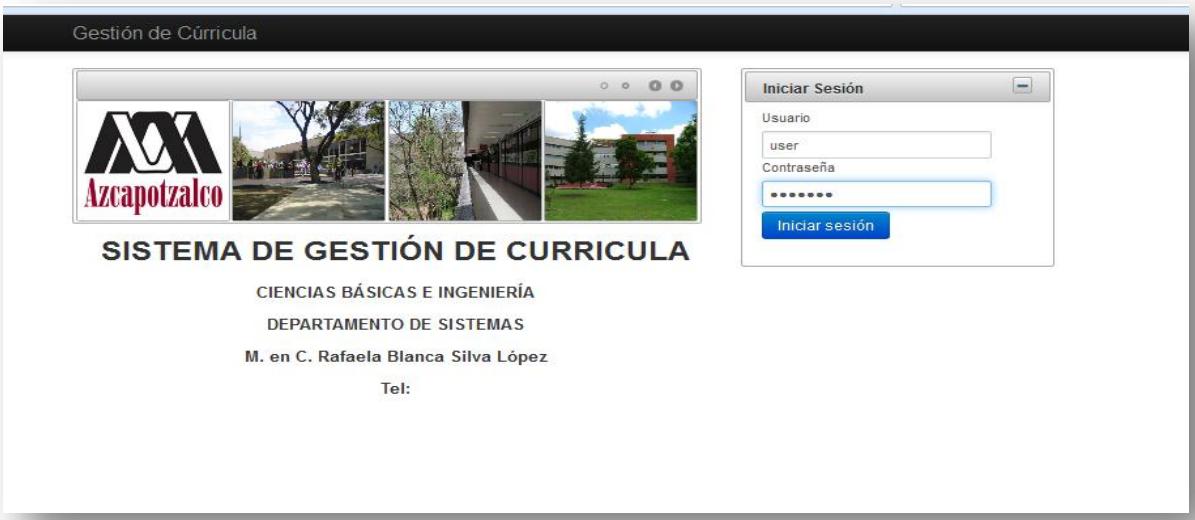


Figura 6 Interfaz de Inicio

En la figura 7se muestra como el usuario al ingresar al sistema podrá interactuar con los 8 diferentes módulos que contiene el sistema como se muestra en la Figura

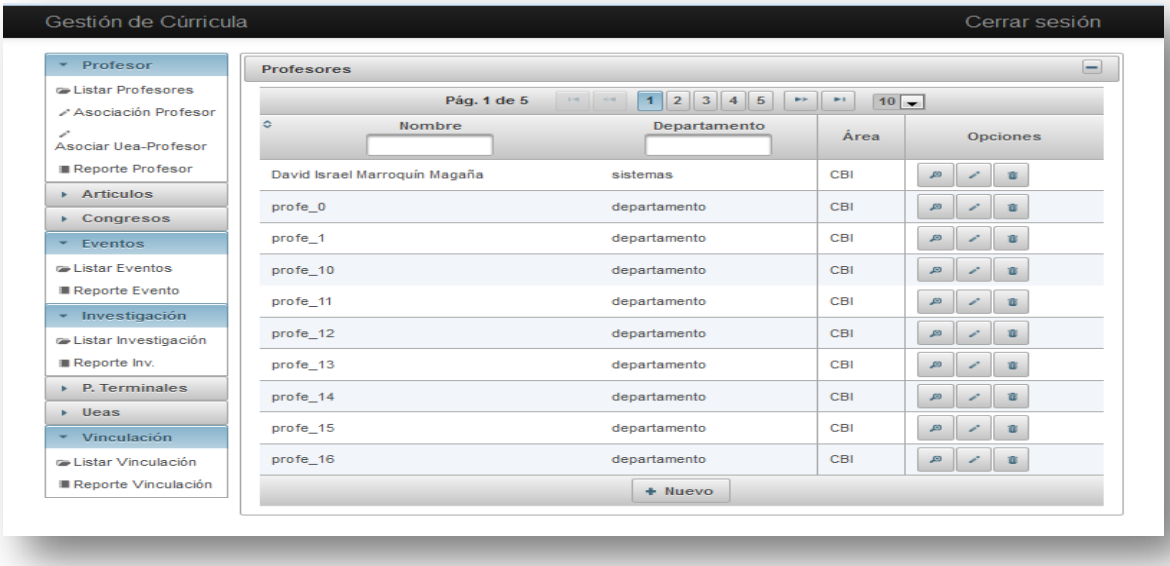


Figura 7 Interfaz Principal de Sistema

Arquitectura del Sistema

En la Figura 8 se muestra la Arquitectura MVC (Modelo Vista Controlador) que se utilizo para el diseño desarrollo e implementación del sistema.

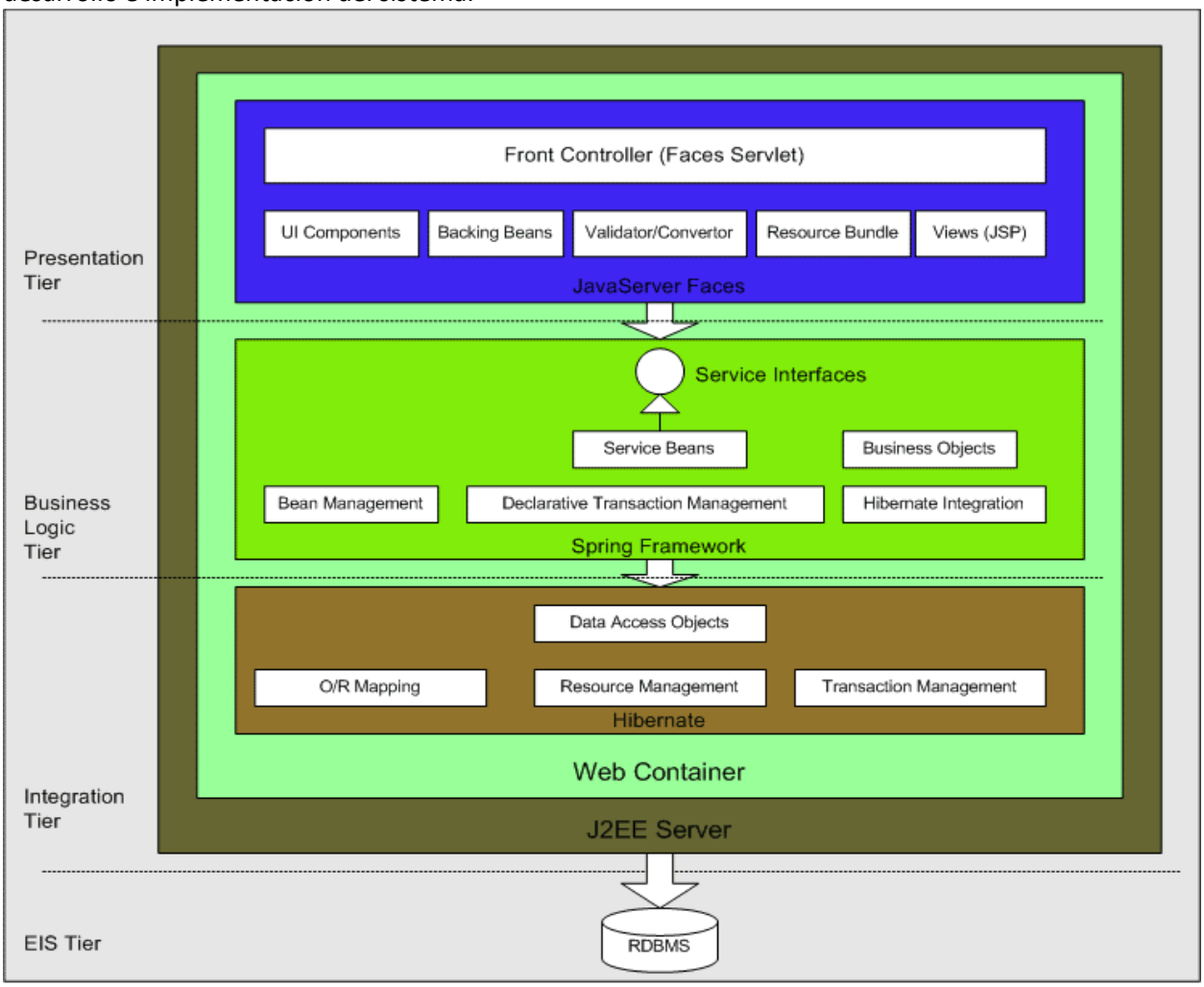


Figura 8

Capa Vista: En esta capa se encuentra la interface de usuario del sistema.

Capa Modelo: En estaa capa el modelo se limita a lo relativo de la *vista* y su *controlador* facilitando las presentaciones visuales.

Capa Controlador: En esta capa se encarga de responder peticiones del sistema generalmente realizadas por el usuario

Diagrama de Navegación

En la sig. Figura se muestra el diagrama de navegación del sistema donde se muestra como el usuario interactúa con el sistema.

Como primer paso el usuario tendrá acceso al sistema mediante un usuario y contraseña después podrá seleccionar entre los diferentes módulos con los que cuenta el sistema para poder registrar usuarios, realizar consultas o generar reportes respectivamente.

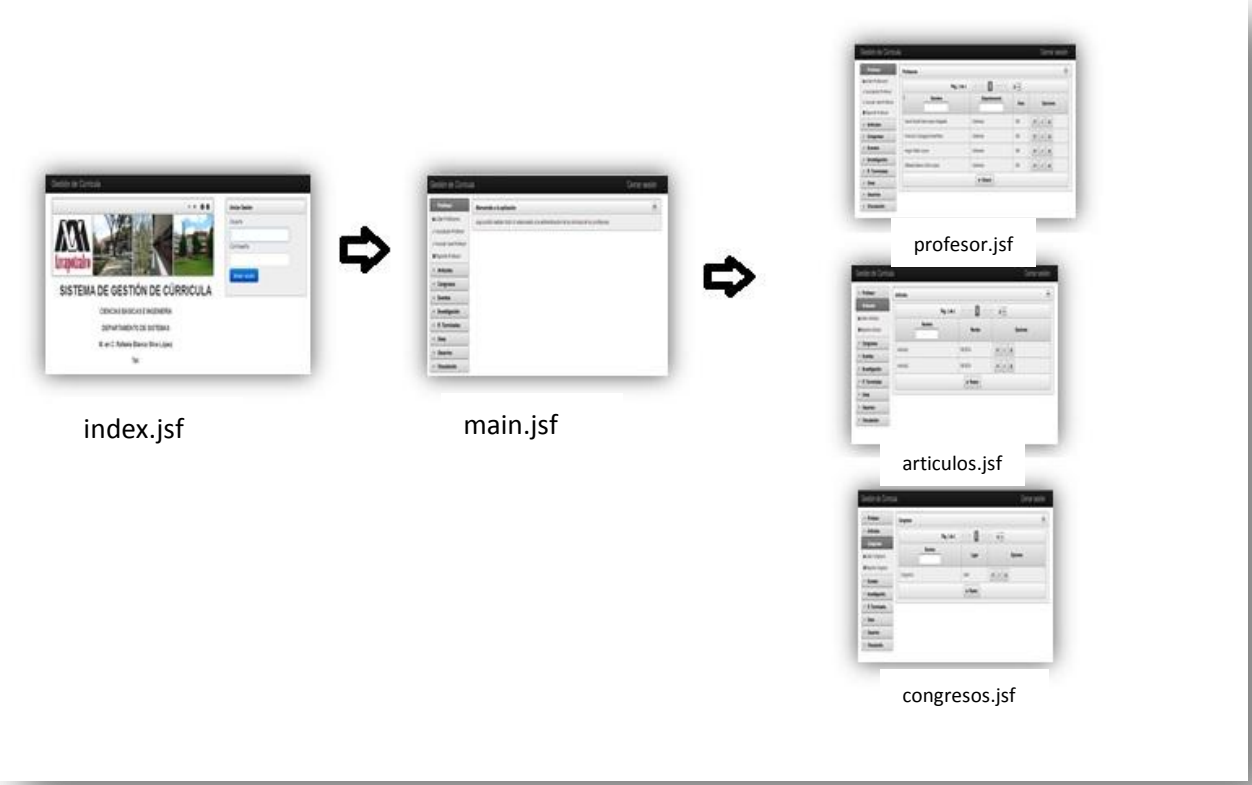


Figura 9

MODULO PROFESOR

En este módulo se realiza el registro de los datos de los profesores que laboran en el departamento, así como consultas por nombre o departamento para tener acceso a la información del personal docente

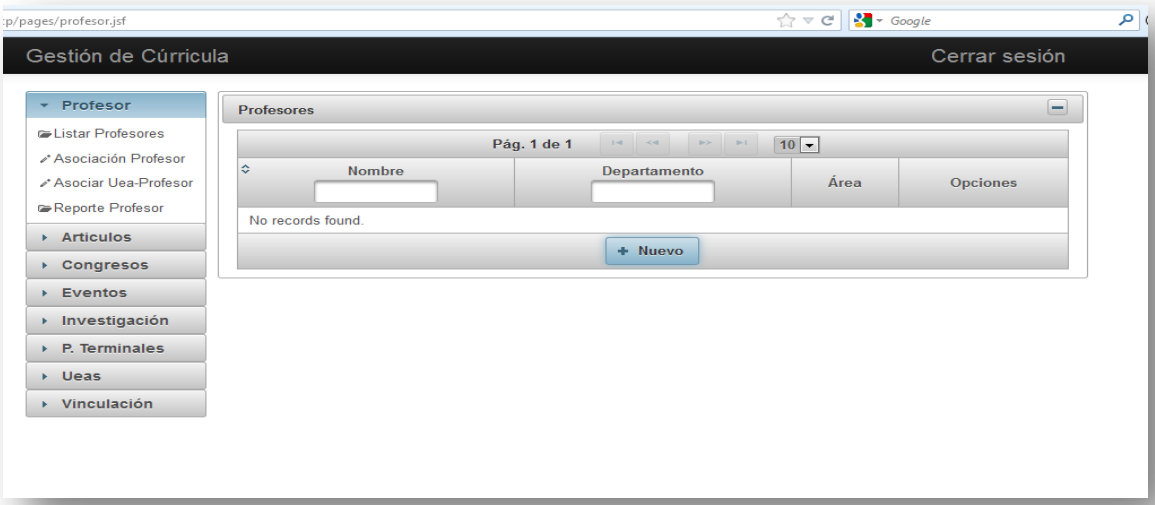


Figura 10 Lista Profesores

En este apartado se capturan los datos personales de cada profesor

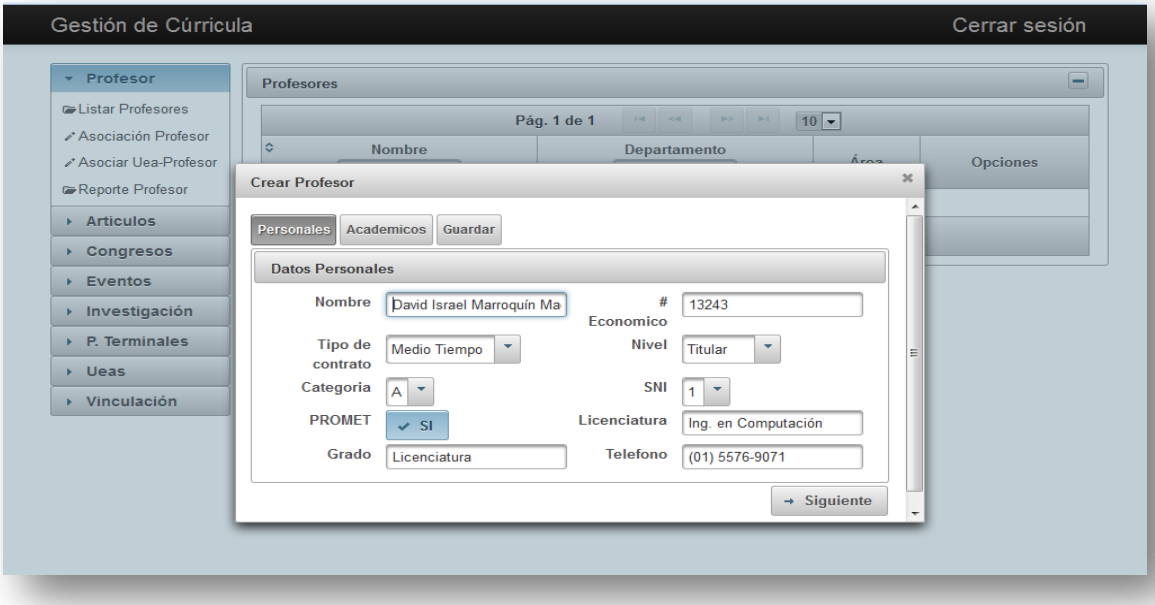


Figura 11 Interfaz de Registro de Profesor

Despues de haber capturado los datos del profesor se le da click en guardar los campos para que puedan guardarse en el sistema.

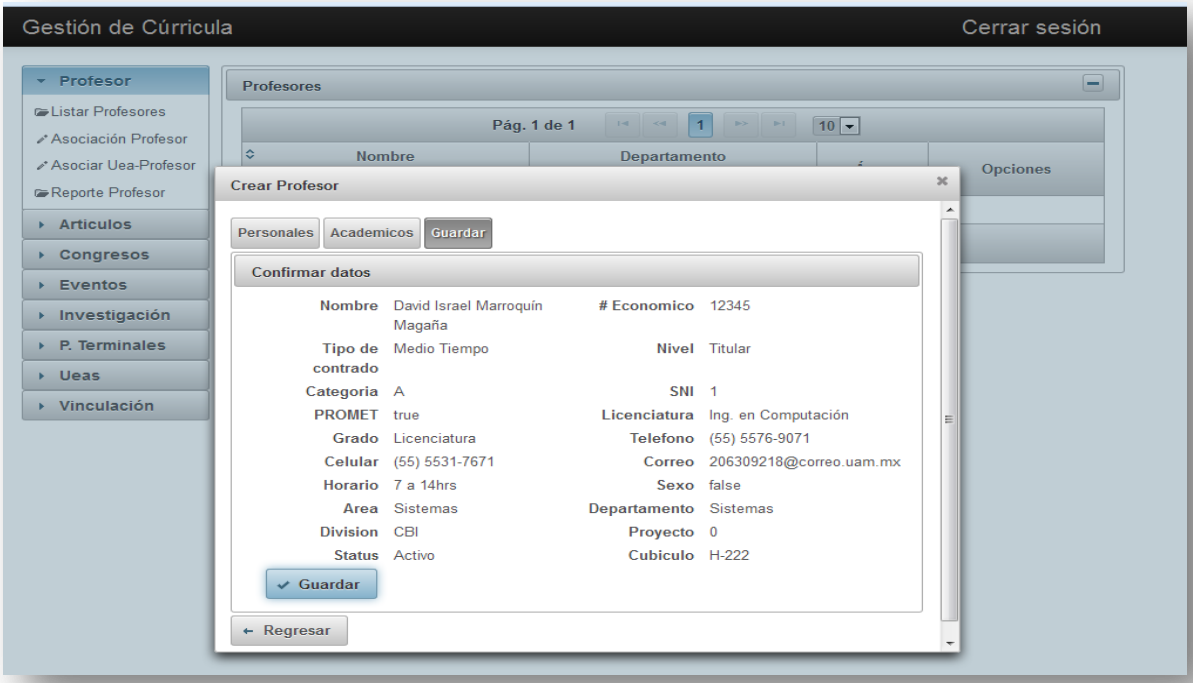


Figura 12 Interfaz de Confirmacion de Datos

En el Modulo de Profesores se pueden realizar busquedas por Nombre y Departamento lo que facilita el acceso a la informacion de los profesores ayudando a usuario a tener un mejor control del personal docente del departamento.

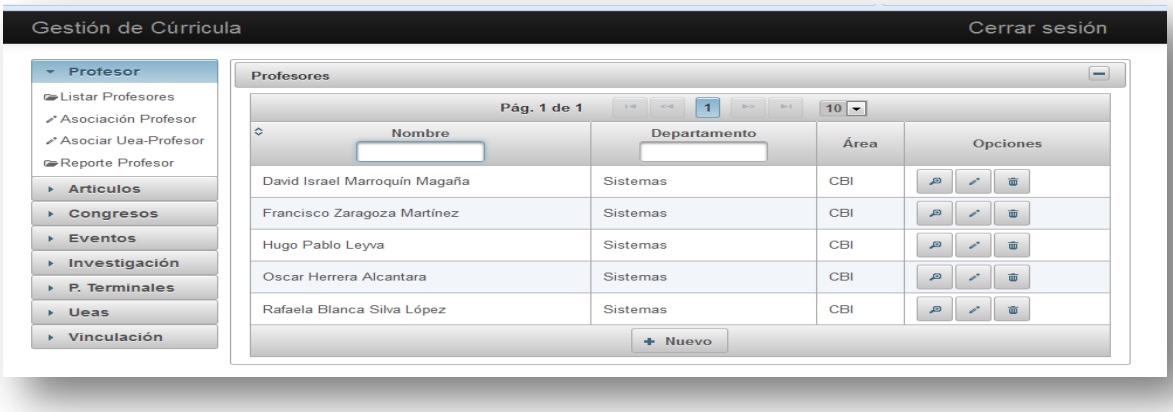


Figura 13 Interfaz de Consultas de Profesor

El sistema tiene en la parte de Opciones el poder ver Detalles el cual muestra los Detalles de los datos del Profesor

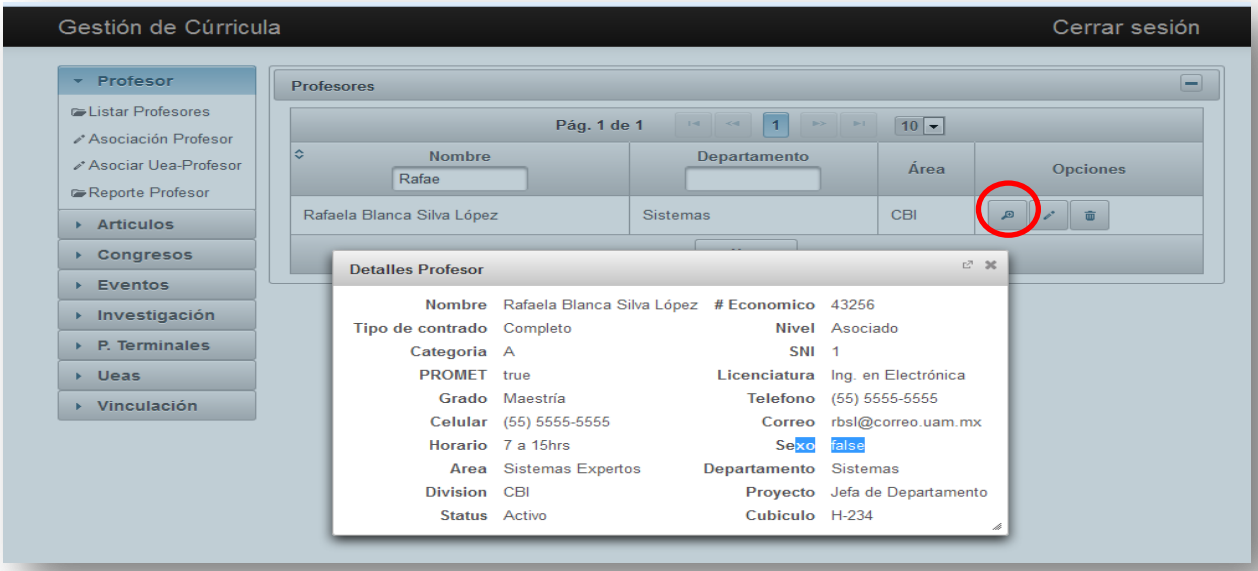


Figura 14 Detalles Profesor

El sistema en la parte de opciones puede editar cualquier campo de los datos del profesor

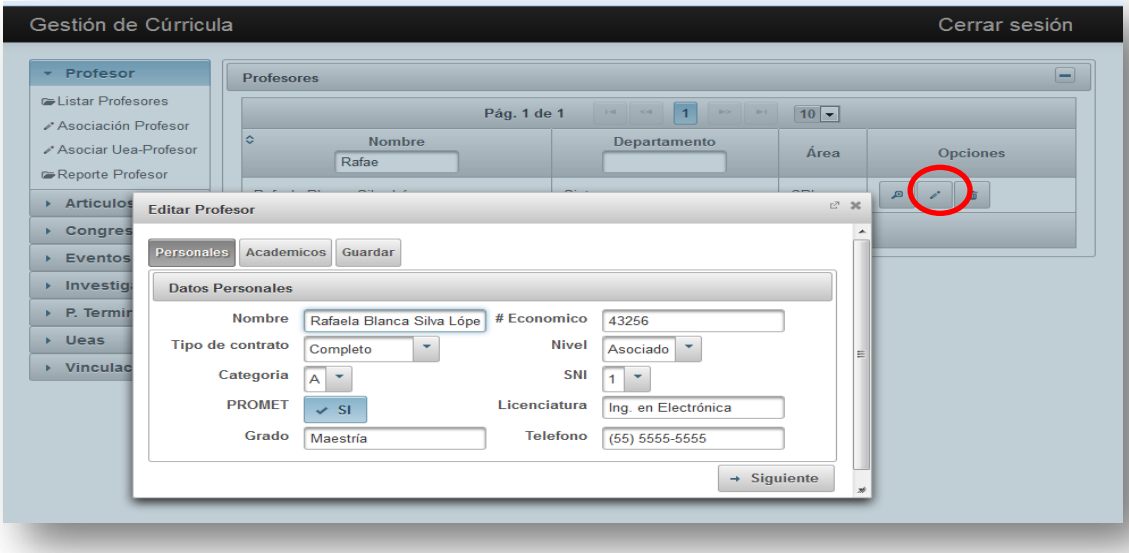


Figura 15 Edicion de Datos Profesor

El sistema en la parte de opciones puede eliminar el registro del profesor dando clik

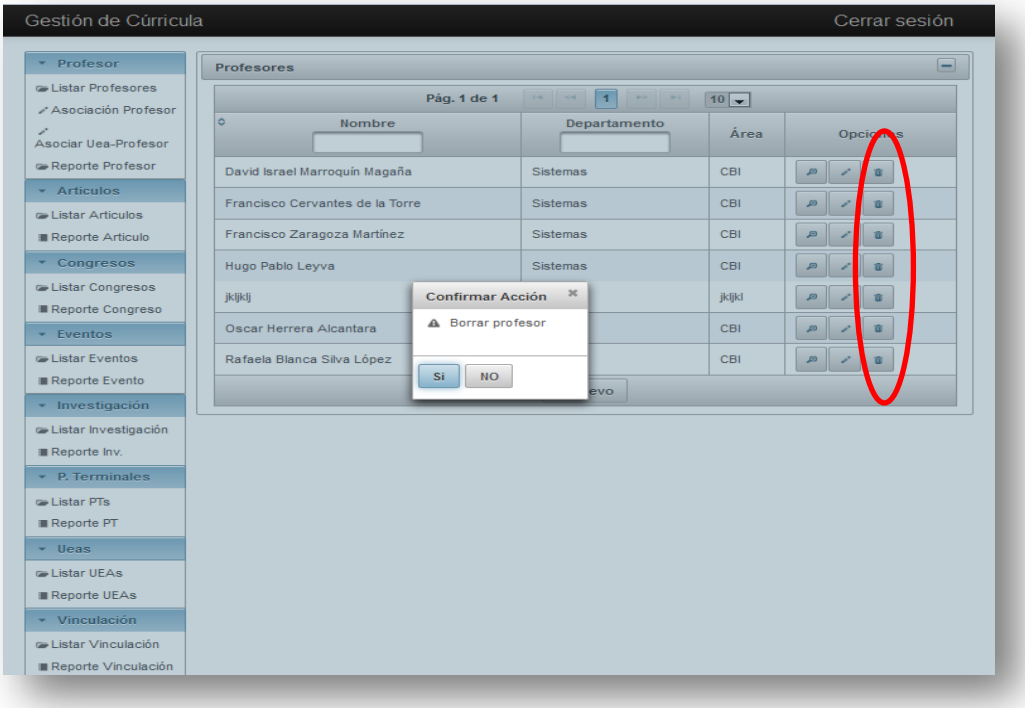


Figura 16 Eliminar Profesor

ASOCIACION PROFESOR

En esta seccion el usuario podra asociar a cada profesor con sus respectivas actividades curriculares dentro de su labor de docente y asi tener un mejor manejo control y administracion de su perfil y expertis.

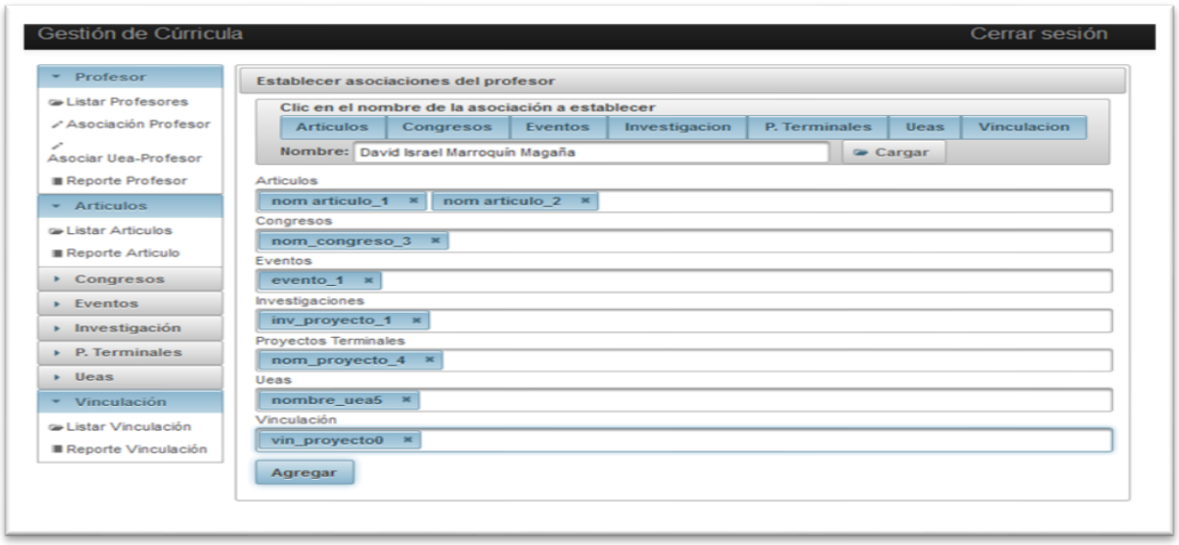


Figura 17 Establecer Asociacion de Profesor

En la Figura 16 se muestran los detalles de las Asociaciones de Profesor para después guardar las Asociaciones mostradas y crear el Vinculo entre ellas.

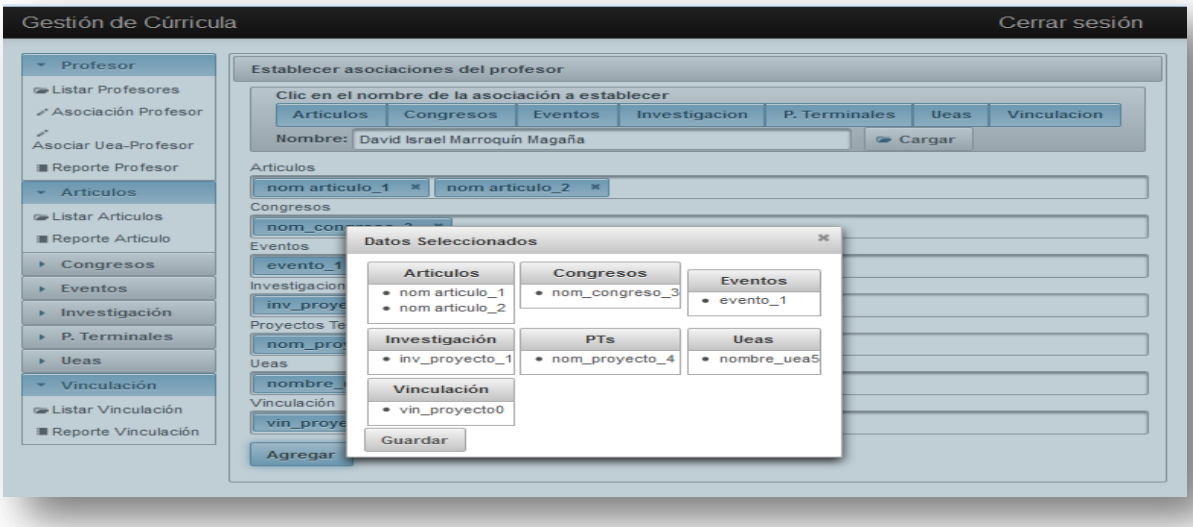


Figura 18 Confirmar Datos Asociacion

En la Figura 17 se muestra a detalle los datos del profesor con sus diferentes Asociaciones y la opcion de generacion de reportes del Profesor.

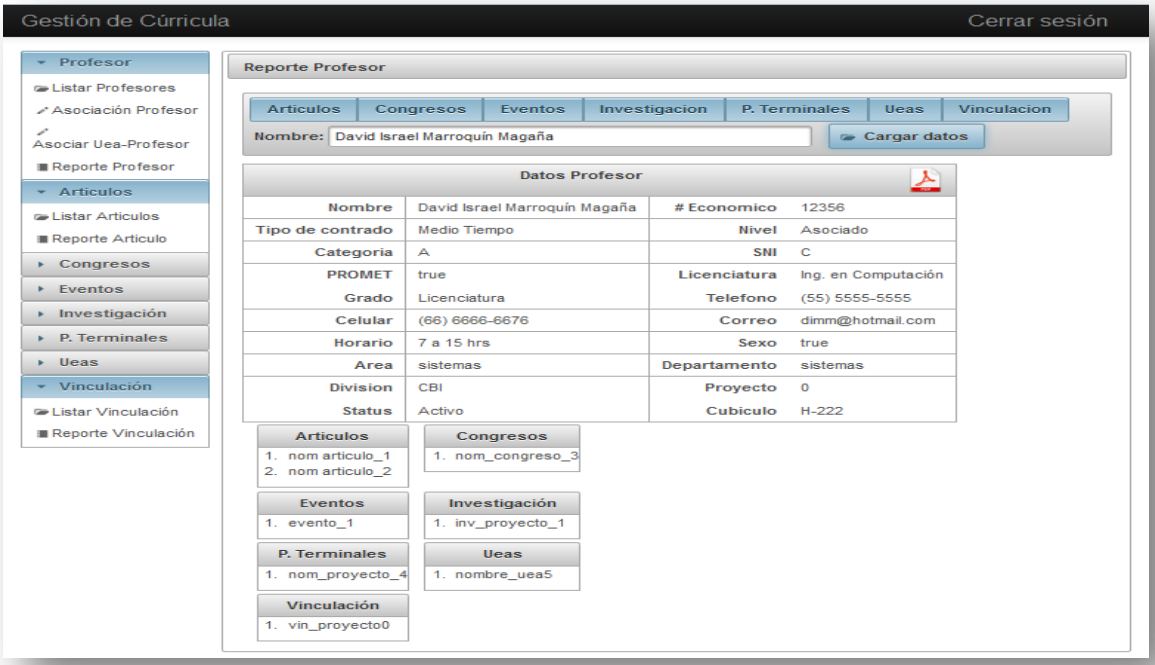


Figura 19 Generacion de Reportes Profesor

PROFESOR-UEA

La Figura 18 muestra la Asociacion de Profesor-UEA-Trimestre

Figura 20 Asociacion Profesor-UEA-Trimestre

MODULO ARTICULOS

En esta seccion el usuario puede registrar los datos de los Articulos realizados por el personal docente

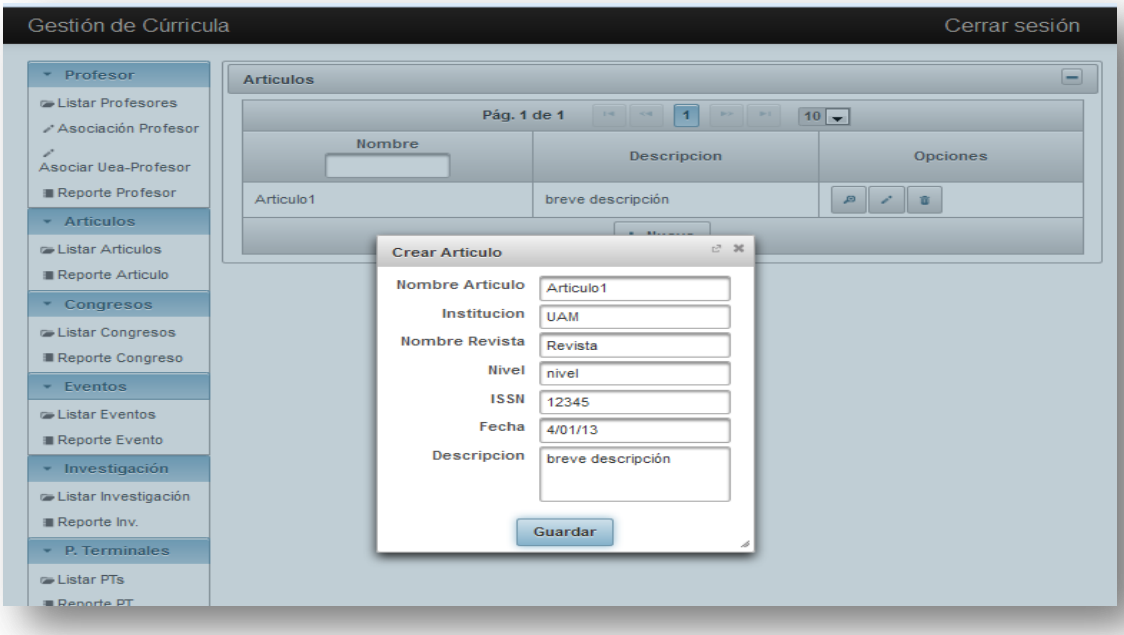


Figura 21 Interfaz Articulos

En la sección de Reporte Articulo se puede realizar consultas por fecha de publicación de los artículos teniendo como resultado los artículos realizados dentro de esas 2 fechas para después poder consultar los detalles y respectivos reportes.

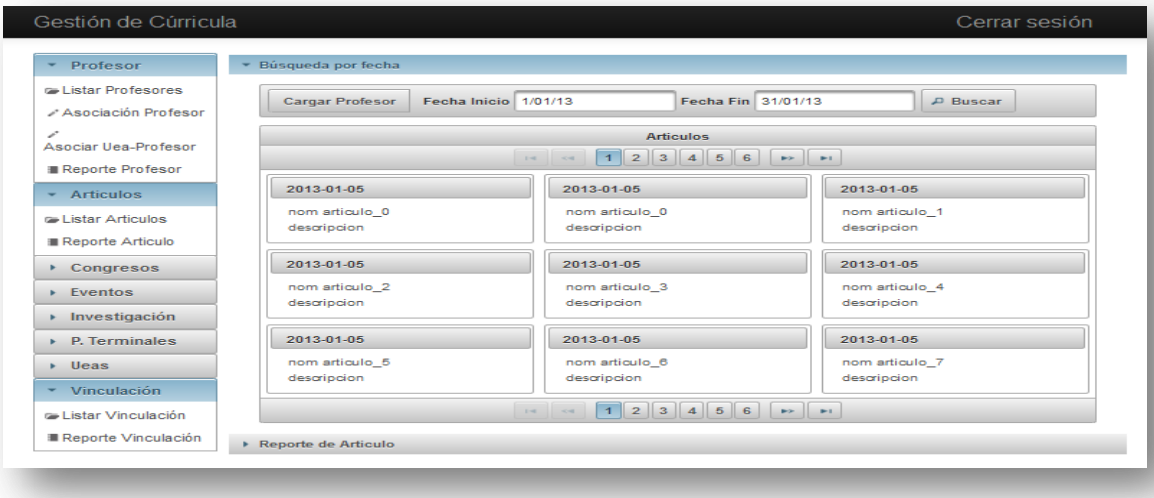


Figura 22 Consultas de Articulos

En la Figura 21 se muestra las consultas de Articulo y generación de Reportes

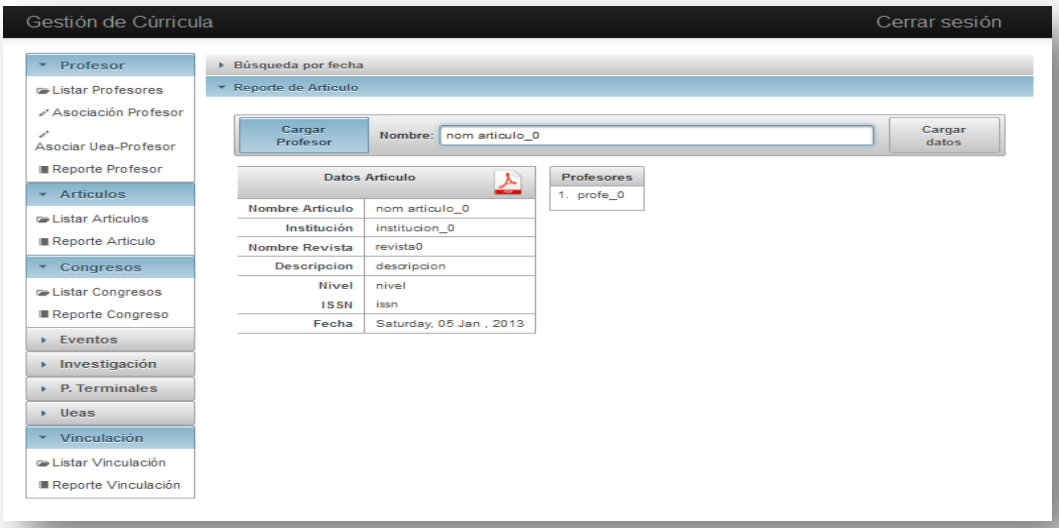


Figura 23 Reportes Articulos

MODULO CONGRESOS

En esta sección el usuario podrá capturar todos los registros de los diferentes congresos en los que hayan participado el personal docente del departamento.

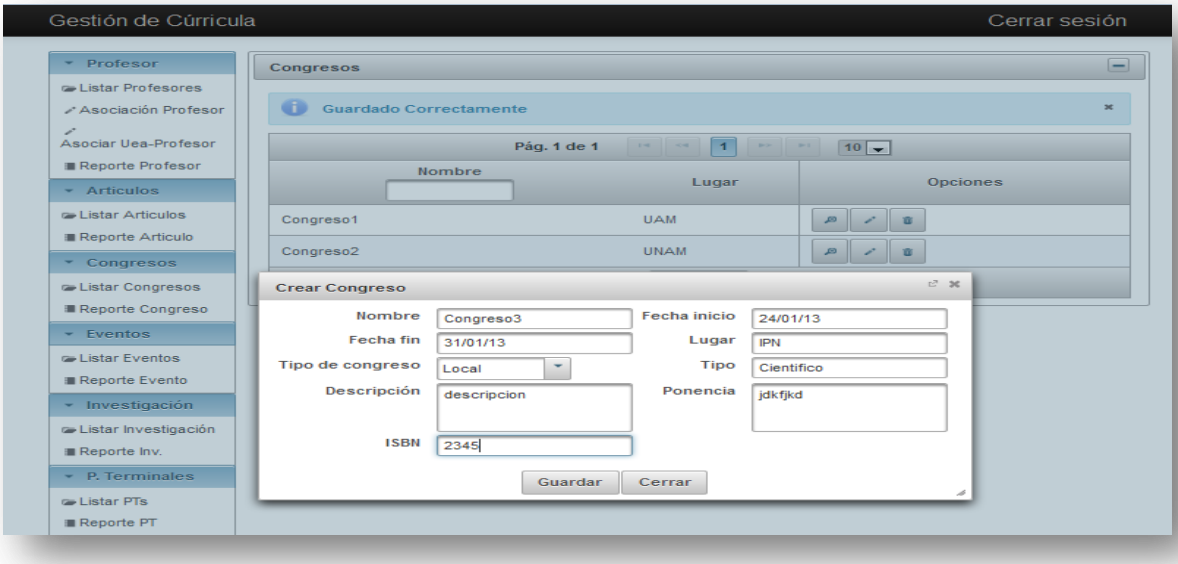


Figura 24 Interfaz Congresos

En la sección de Reporte Congreso se puede realizar consultas por fecha de publicación de los congresos teniendo como resultado los congresos realizados dentro de esas 2 fechas para después poder consultar los detalles de las participaciones del personal docente y poder generar reportes.

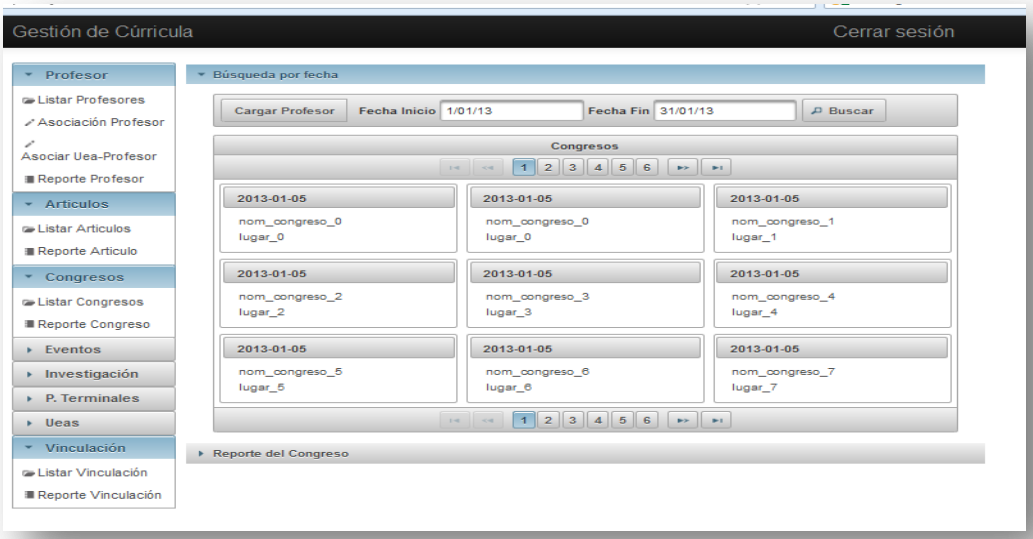


Figura 25 Consultas de Congresos

En la Figura 24 se muestra la Interfaz de genracion de Reportes por Congresos del Profesor

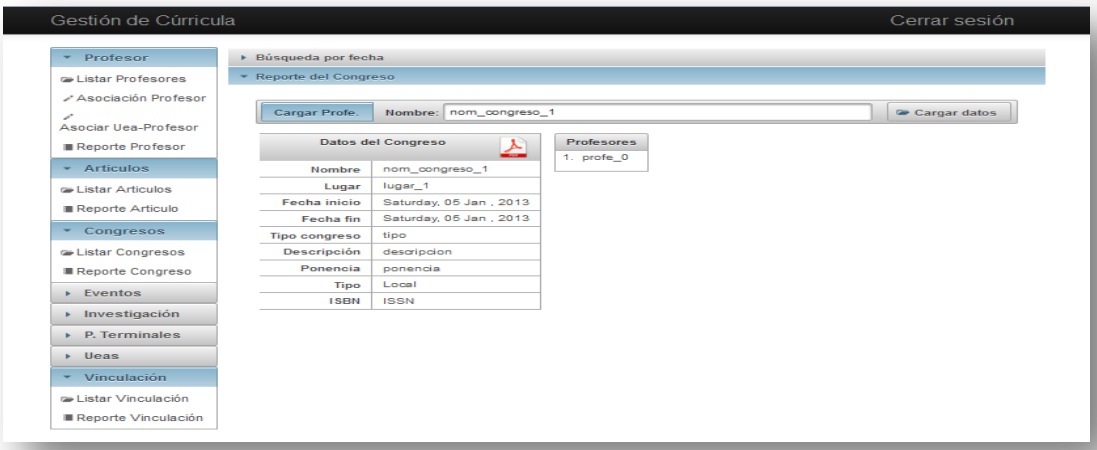


Figura 26 Reportes Congresos

MODULO EVENTOS

En esta sección el usuario podrá capturar todos los registros de los diferentes eventos en los que hayan participado el personal docente del departamento.

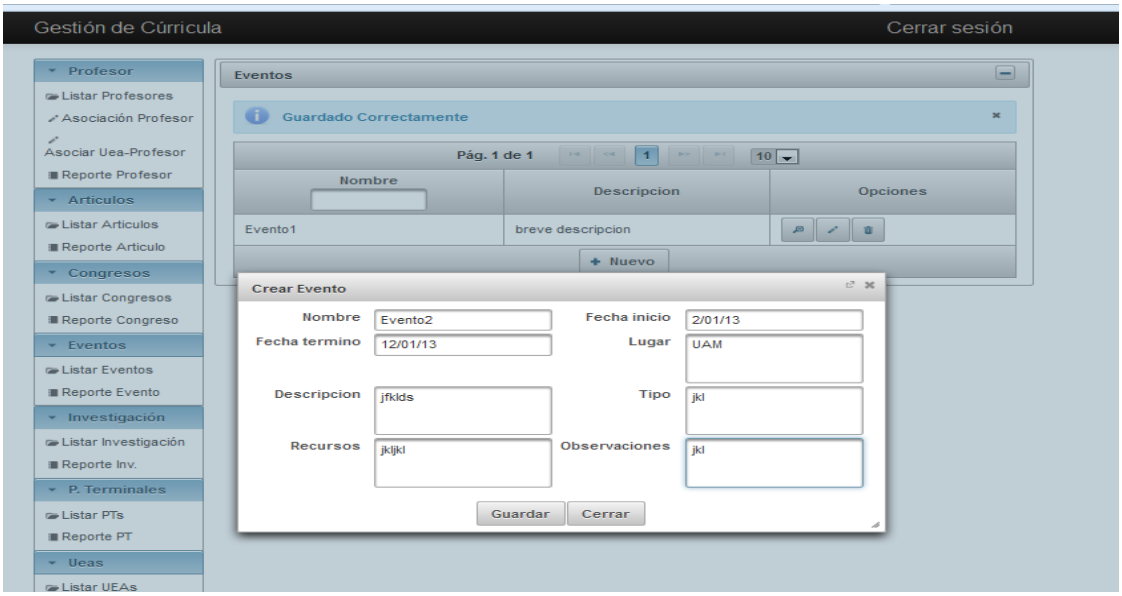


Figura 27 Registro Eventos

En la figura 26 se muestran las consultas por fechas de los diferntes eventos de los profesores

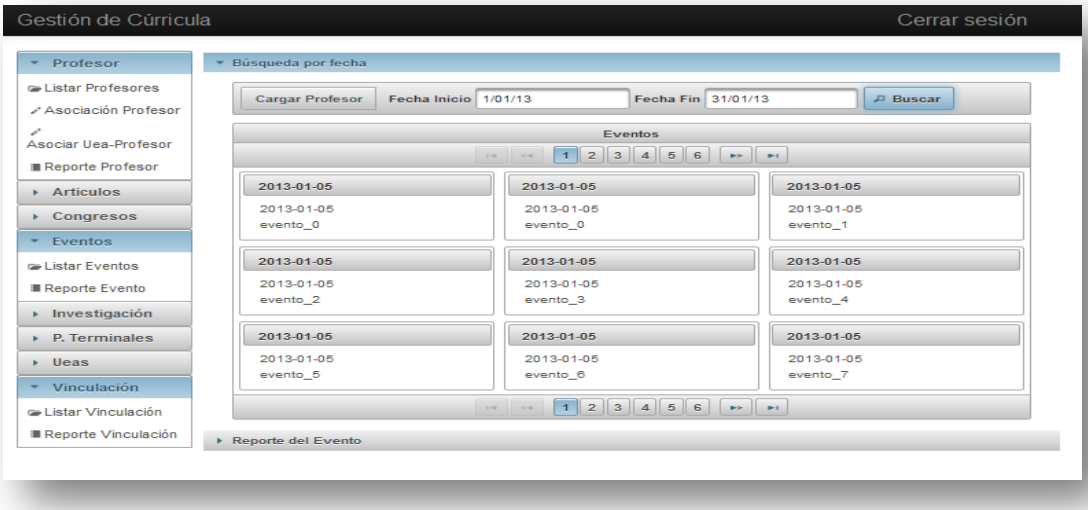


Figura 28 Consultas de Eventos

En la Figura27 se muestra la generacion de Reportes de Eventos Profesor

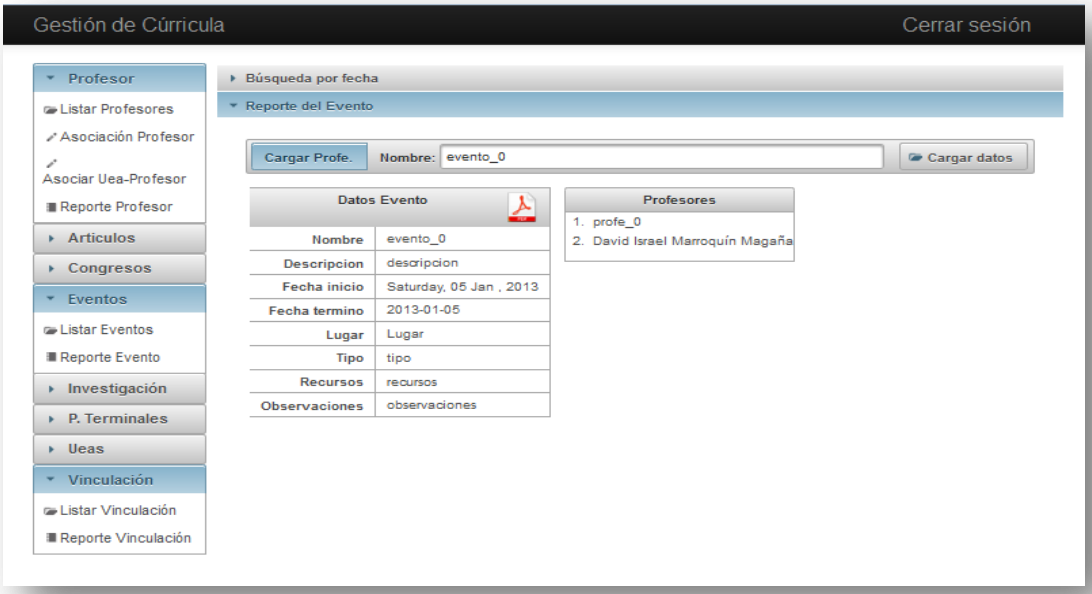


Figura 29 Reportes de Eventos

MODULO INVESTIGACIÓN

En esta sección el usuario podrá capturar todos los registros de las diversas investigaciones realizadas por el personal docente del departamento.

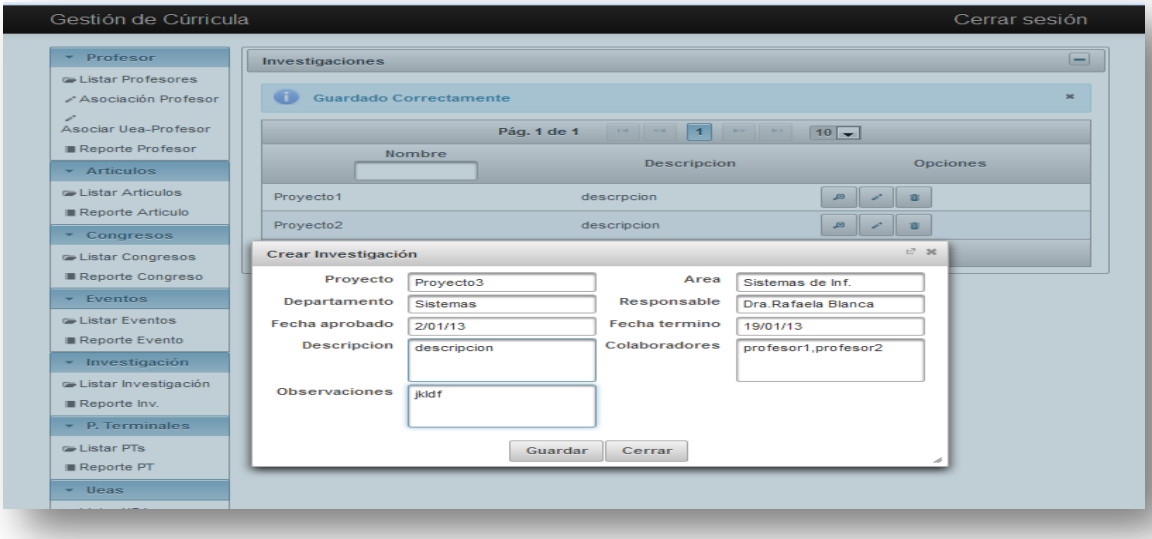


Figura 30 Registro de Investigacion

En la Figura 29 se muestra las consultas de Investigacion Profesor asi como la generacion de Reportes

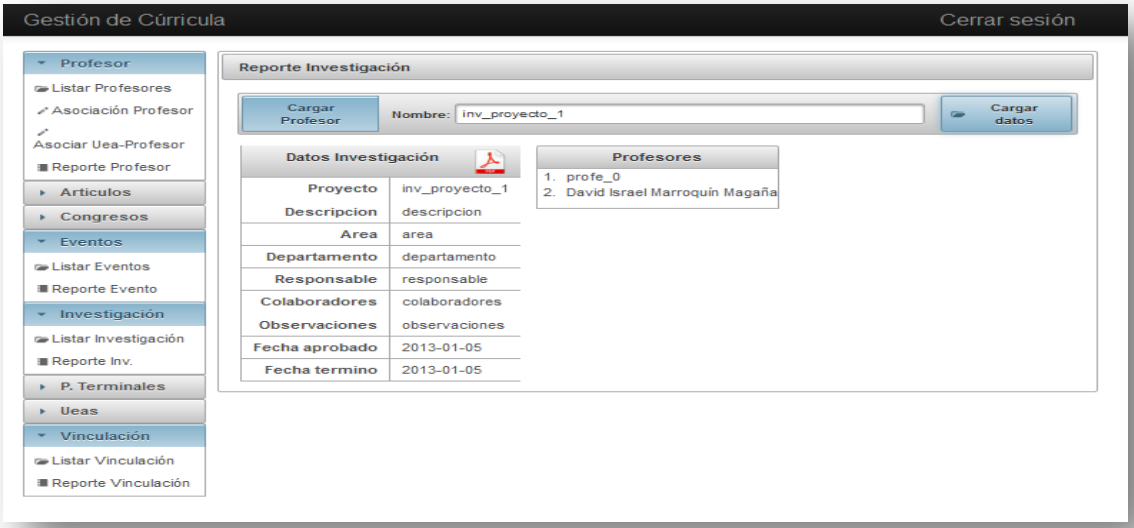


Figura 31 Reportes de Investigacion

MODULO PROYECTOS TERMINALES

En esta sección el usuario podrá capturar todos los registro de los diferentes Proyectos Terminales del departamento

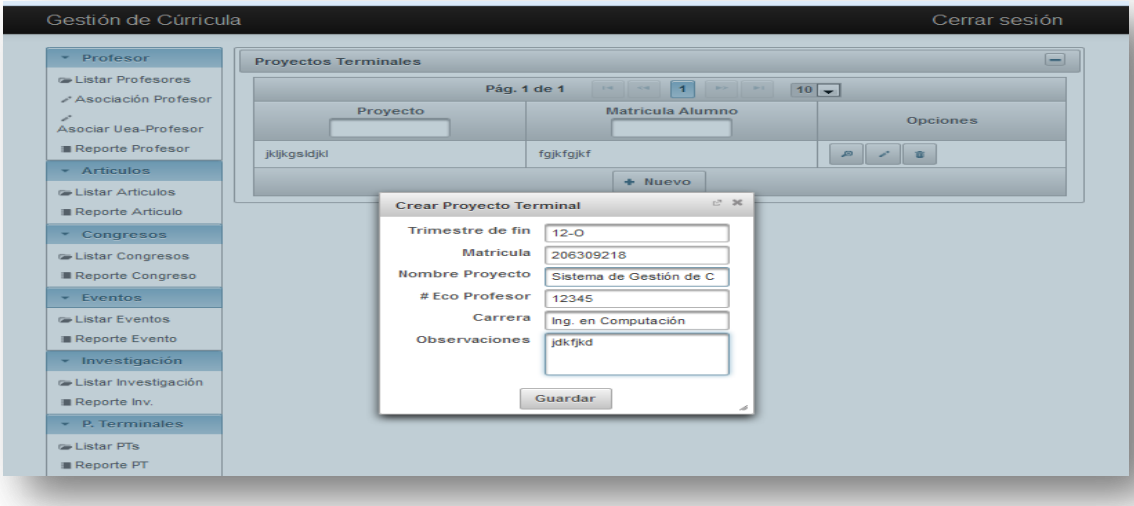
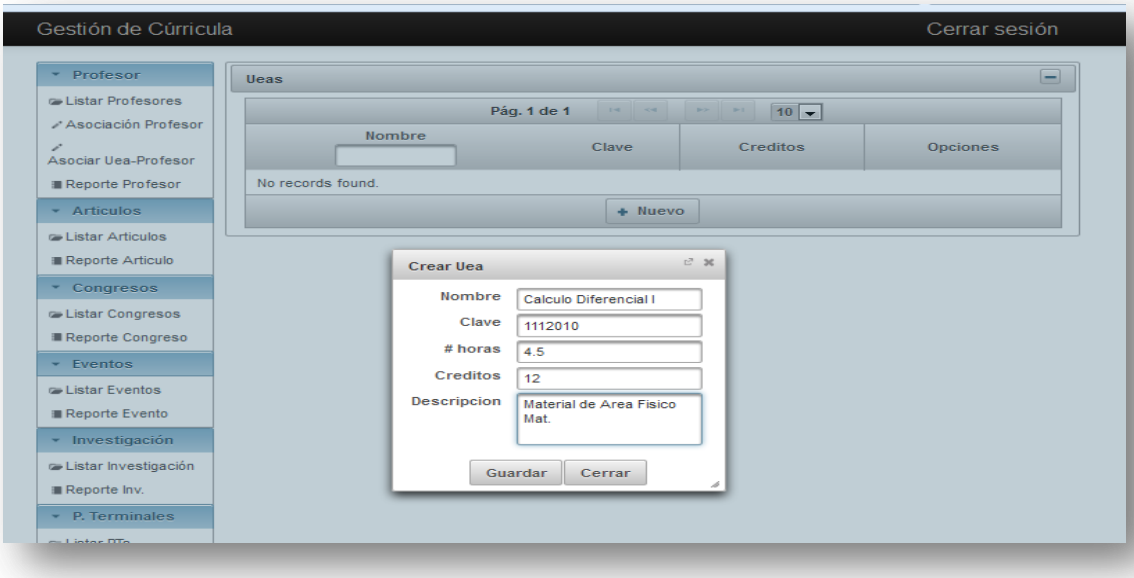


Figura 32 Registro Proyecto Terminal

MODULO UEA

En esta sección el usuario podrá capturar todos los registros de las diversas UEAS impartidas en el departamento.



MODULO DE VINCULACIÓN

En esta sección el usuario podrá capturar todos los registros relacionados con Vinculación del departamento y así poder tener un mejor manejo y control de la información relacionada vinculación-profesor.

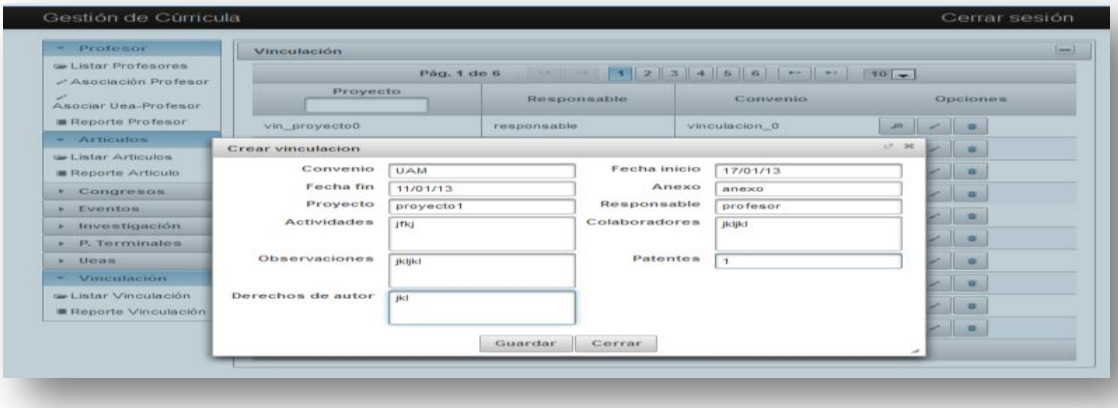


Figura 33 Registro Vinculación

CODIGO FUENTE

En el paquete com.marroquin.gestion.curricula.action se encuentran las clases ManagedBean que son los Bean para generar las vistas del JSF

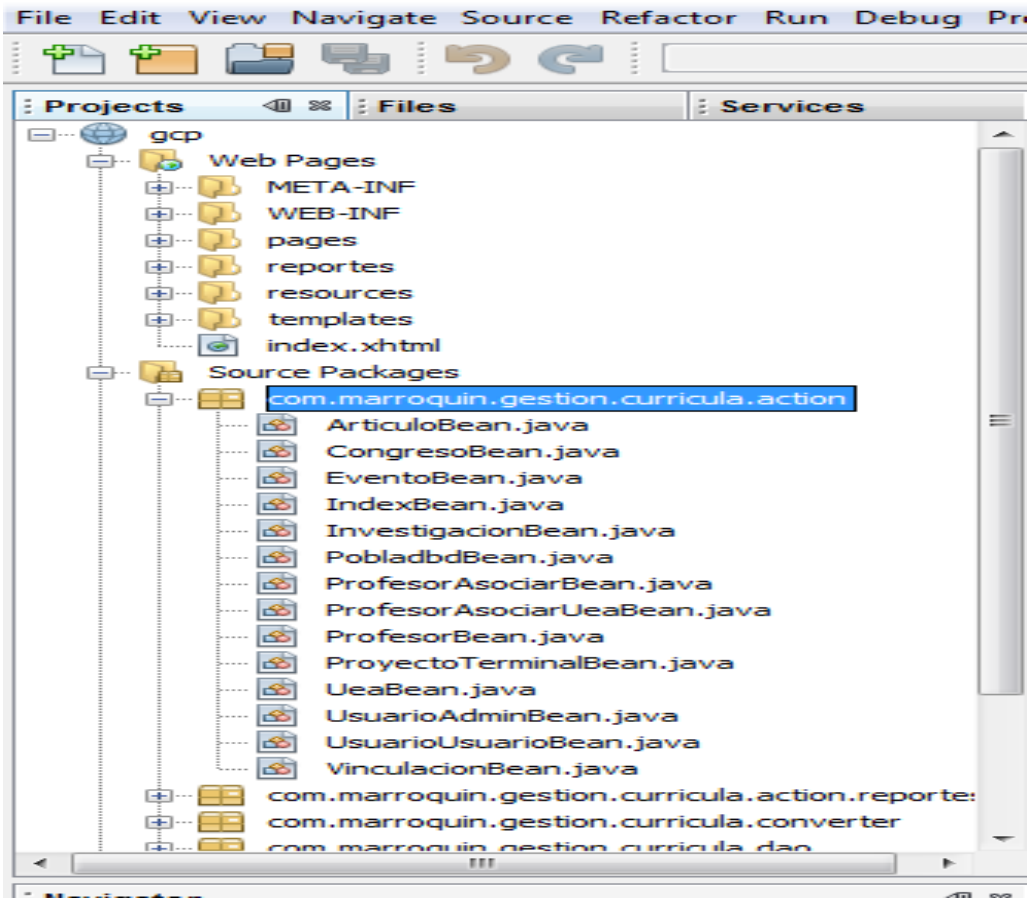


Figura 34

```
public class ArticuloBean implements Serializable{
   private Articulo articulo=new Articulo();
   private Articulo selectedArticulo;
   @ManagedProperty("#{articuloService}")
   private ArticuloService dao;
   private LazyDataModel<Articulo> allArticulos;
   @ManagedProperty("#{dataModelArticulo}")
   private DataModelArticulo articulosm;
   private boolean cargarAsociacion;
   private List<Articulo> filterArticulos;
   public ArticuloBean() { }
   public String articuloProfesor(){
```

```
    selectedArticulo = dao.obtener(selectedArticulo.getId(),null);
    return ""; }

  public void addArticulo(ActionEvent e){
    dao.guardar(articulo);
     FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO, "Guardado
Correctamente", null);
     FacesContext.getCurrentInstance().addMessage(null, message);
     articulo=new Articulo();
  }
   public void updateArticulo(ActionEvent e){
    dao.actualizar(selectedArticulo);
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO, "Cambios Guardados",
null);
    FacesContext.getCurrentInstance().addMessage(null, message);
  }

  public void deleteArticulo(ActionEvent e){
    dao.eliminar(selectedArticulo);
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO, "Eliminado
Correctamente", null);
    FacesContext.getCurrentInstance().addMessage(null, message);
    selectedArticulo=null;
  }

  public LazyDataModel<Articulo> getAllArticulos() {
     if(allArticulos==null){
        allArticulos=articulosm;
     }
     if(cargarAsociacion){
        Map<String, String> ca =new HashMap<String, String>(1);
        ca.put("profesores","profesores");
        articulosm.setLoadAsociacion(ca);
     }
     return allArticulos;
  }

  //Getter y setter

  public Articulo getArticulo() {
     return articulo;
  }
```

```
    public void setArticulo(Articulo articulo) {
        this.articulo = articulo;
    }

    public Articulo getSelectedArticulo() {
        return selectedArticulo;
    }

    public void setSelectedArticulo(Articulo selectedArticulo) {
        this.selectedArticulo = selectedArticulo;
    }

    public void setDao(ArticuloService dao) {
        this.dao = dao;
    }

    public List<Articulo> getFilterArticulos() {
        return filterArticulos;
    }

    public void setFilterArticulos(List<Articulo> filterArticulos) {
        this.filterArticulos = filterArticulos;
    }

    public boolean isCargarAsociacion() {
        return cargarAsociacion;
    }

    public void setCargarAsociacion(boolean cargarAsociacion) {
        this.cargarAsociacion = cargarAsociacion;
    }

    public void setArticulosm(DataModelArticulo articulosm) {
        this.articulosm = articulosm;
    }
}
```

En la figura35 se muestra el paquete com.marroquin.gestion.curricula.action.reportes que contiene las clases managedBean que se utilizan para generar reportes en los JSF

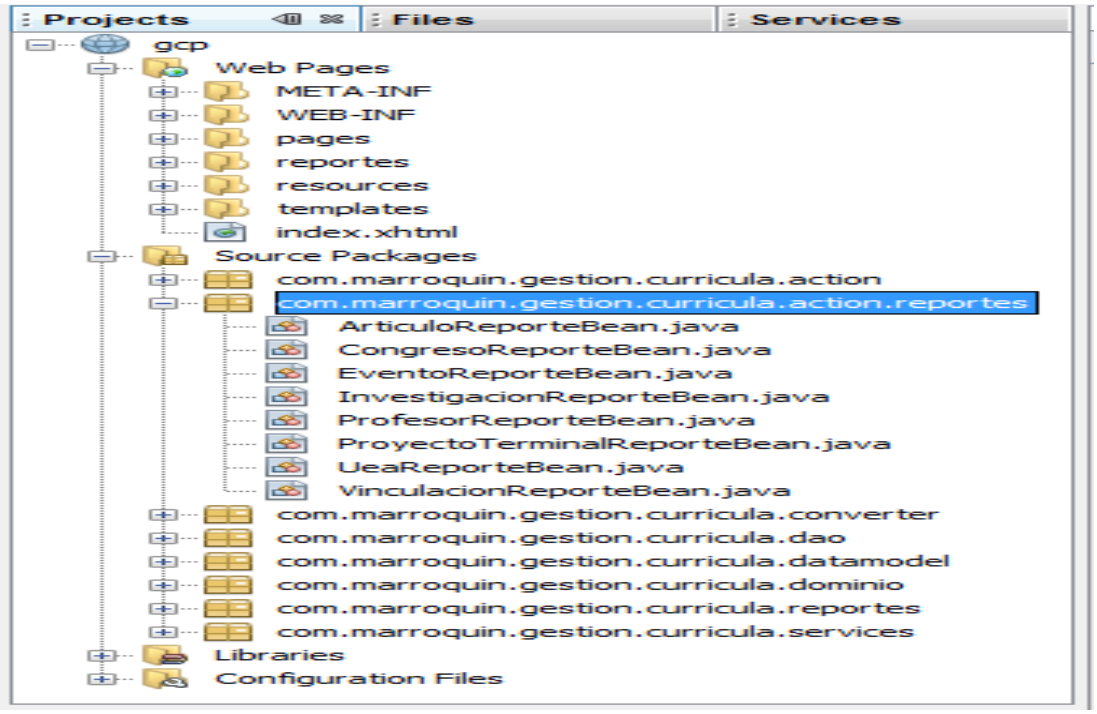


Figura 35

```
@ManagedBean(name = "reporteArticulo")
@ViewScoped
public class ArticuloReporteBean implements Serializable{
   private List<Articulo> articulos=new ArrayList<Articulo>(0);
   private Date fechaInicio;
   private Date fechaFin;
   private List<String> cargaAsociacion;
   @ManagedProperty("#{articuloService}")
   private ArticuloService dao;
   private Articulo selectedArticulo;
   private IdCampo selectedid;
   /**
    * Creates a new instance of ArticuloReporteBean
    */
   public ArticuloReporteBean() {
   }

   public void articulosFecha(ActionEvent e){
      articulos=dao.findByFecha(fechaInicio, fechaFin,cargaAsociacion);
```

```
    }

    public void loadSelectedArticulo(ActionEvent e){
        selectedArticulo=dao.obtener(selectedid.getId(), cargaAsociacion);
    }

    public void generarReporte(ActionEvent e) throws JRException, IOException{
        if(selectedArticulo!=null){
            InputStream reportTemplate =
((ServletContext)FacesContext.getCurrentInstance().getExternalContext().getContext())
                                        .getResourceAsStream("/reportes/reporte-articulo.jasper");
            Map<String,Object> parametros=new HashMap<String, Object>(1);
            articulos=new ArrayList<Articulo>(1);
            articulos.add(selectedArticulo);
            if(!cargaAsociacion.isEmpty()){
                parametros.put("label_profesor","Profesores:");
            }else{
                parametros.put("label_profesor","");
                lazyCollection();
            }
            JasperReport reporte = (JasperReport)JRLoader.loadObject(reportTemplate);
            JasperPrint  jasperPrint=JasperFillManager.fillReport(reporte, parametros,new
JRBeanCollectionDataSource(articulos));

            HttpServletResponse
httpServletResponse=(HttpServletResponse)FacesContext.getCurrentInstance().getExternalContext().get
Response();
            httpServletResponse.addHeader("Content-disposition", "attachment; filename=reporte-
articulo.pdf");
            ServletOutputStream servletOutputStream=httpServletResponse.getOutputStream();
            JasperExportManager.exportReportToPdfStream(jasperPrint, servletOutputStream);
            FacesContext.getCurrentInstance().responseComplete();
        }
        else{
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_WARN, "No existen datos
para generar el reporte", null);
            FacesContext.getCurrentInstance().addMessage(null, message);
        }
    }

    private void lazyCollection(){
        for (int i = 0; i < articulos.size(); i++) {
```

```
      Articulo articulo = articulos.get(i);
      articulo.setProfesoreses(new HashSet<Profesor>(0));
   }
}

public List<Articulo> cargaArticuloAutocomplete(String query) {
   if(query==null || query.length()==0){
      return null;
   }
   else{
       return dao.datosAutocomplete("nomArticulo", query);
   }
}

public List<IdCampo> loadArticuloAutocomplete(String query) {
   if(query==null || query.length()==0){
      return null;
   }
   else{
      List<IdCampo> suggestions=dao.autocomplete("nomArticulo", query);
      IdCampoConverter.setCamposAutocomplete(suggestions);
       return suggestions;
   }
}

public List<Articulo> getArticulos() {
   return articulos;
}

public void setArticulos(List<Articulo> articulos) {
   this.articulos = articulos;
}

public Date getFechaInicio() {
   return fechaInicio;
}

public void setFechaInicio(Date fechaInicio) {
   this.fechaInicio = fechaInicio;
}

public Date getFechaFin() {
```

```
        return fechaFin;
    }

    public void setFechaFin(Date fechaFin) {
        this.fechaFin = fechaFin;
    }

    public void setDao(ArticuloService dao) {
        this.dao = dao;
    }

    public Articulo getSelectedArticulo() {
        return selectedArticulo;
    }

    public void setSelectedArticulo(Articulo selectedArticulo) {
        this.selectedArticulo = selectedArticulo;
    }

    public IdCampo getSelectedid() {
        return selectedid;
    }

    public void setSelectedid(IdCampo selectedid) {
        this.selectedid = selectedid;
    }

    public List<String> getCargaAsociacion() {
        return cargaAsociacion;
    }

    public void setCargaAsociacion(List<String> cargaAsociacion) {
        this.cargaAsociacion = cargaAsociacion;
    }
}
```

En la Figura 36 se muestra el paquete com.marroquin.gestion.curricula.converter en el cual contiene las clases Converter que sirve para convertir un String a un Objeto

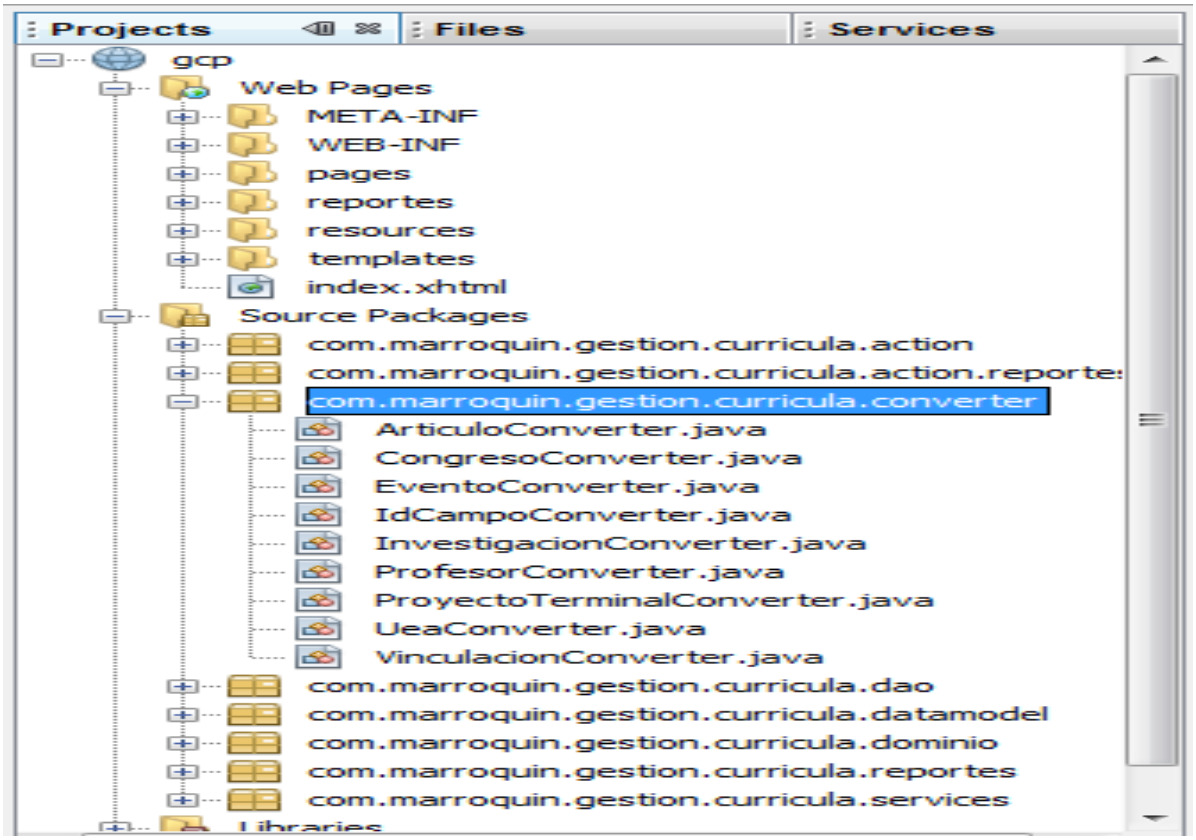


Figura 36

```
@Component("articuloConverter")
public class ArticuloConverter implements Converter{
   private static List<Articulo> dao;
   @Override
   public Object getAsObject(FacesContext fc, UIComponent uic, String value) {
      if (value == null || value.length() == 0) {
         return null;
      }else {
         try {
            Long id=Long.parseLong(value);
            for (Articulo obj : dao) {
               if (obj.getId() == id) {
                  return obj;
               }
            }

         } catch(NumberFormatException exception) {
```

```
            throw new ConverterException(new FacesMessage(FacesMessage.SEVERITY_ERROR, "Error de conversión", "Valor de audio no valido"));
        }
        return null;     } }
 @Override
   public String getAsString(FacesContext fc, UIComponent uic, Object o) {
      return o instanceof Articulo ? ((Articulo) o).getId().toString() : "";
   }
  public static void setDao(List<Articulo> dao) {
      ArticuloConverter.dao = dao;
   }}
```

En la Figura 37 se muestra el paquete com.marroquin.gestion.curricula.datamodel en el cual contiene las clases DOA quecontiene toda la logica para conectarcon la base de datos y obtener todos los datos necesarios correspondientes a las tablas

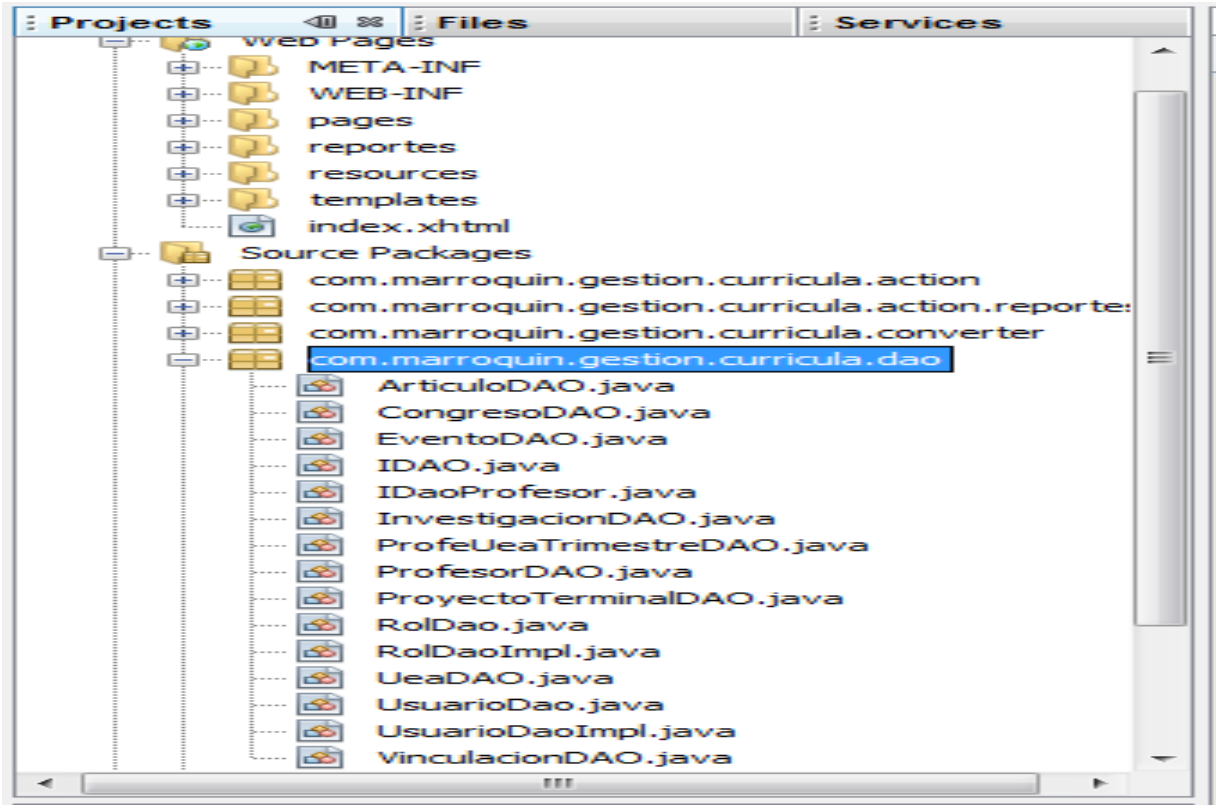


Figura 37

```
@Repository("articuloDao")
public class ArticuloDAO implements IDAO<Articulo>,Serializable {
   private HibernateTemplate ht;
```

```
    @Autowired
    public void setSessionFactory(SessionFactory sessionFactory){
                        ht = new HibernateTemplate(sessionFactory);
    }
    @Override
    public void actualizar(Articulo entity) {
       ht.update(entity);
    }
    @Override
    public void eliminar(Articulo entity) {
       ht.delete(entity);
    }
    @Override
    public long guardar(Articulo entity) {
       long id=(Long)ht.save(entity);
       return id;
    }
    @Override
    public Articulo obtener(long id,List<String> cargaAsociacion) {
       DetachedCriteria crit= DetachedCriteria.forClass(Articulo.class);
       if(cargaAsociacion!=null && cargaAsociacion.contains("profesores")){
          crit.setFetchMode("profesoreses", FetchMode.JOIN);
       }
        crit.add(Restrictions.idEq(id));
       return (Articulo)DataAccessUtils.uniqueResult(ht.findByCriteria(crit));
    }

    @Override
    public List<Articulo> obtenAll(List<String> cargaAsociacion) {
      DetachedCriteria crit= DetachedCriteria.forClass(Articulo.class);
      if(cargaAsociacion!=null && cargaAsociacion.contains("profesores")){
         crit.setFetchMode("profesoreses", FetchMode.JOIN);
      }
      crit.addOrder(Order.asc("nomArticulo"));
      return ht.findByCriteria(crit);
    }

    @Override
    public List<Articulo> findByFecha(Date rangoInferior, Date rangoSuperior,List<String>
cargaAsociacion) {
       DetachedCriteria crit=DetachedCriteria.forClass(Articulo.class);
       crit.add(Restrictions.between("fecha", rangoInferior, rangoSuperior));
```

```
    if(cargaAsociacion!=null && cargaAsociacion.contains("profesores")){
      crit.setFetchMode("profesoreses", FetchMode.JOIN);
    }
    crit.addOrder(Order.asc("fecha"));
    return ht.findByCriteria(crit);
}

 @Override
public List<Articulo> lazyLoanding(int first, int pagesize,String sortField,boolean sortOrder,
       Map<String,String>filtros,Map<String,String> loadAsociacion) {
    List<Articulo> lista=null;
    Session session = ht.getSessionFactory().openSession();
    Transaction tx = session.beginTransaction();
    try{
       Criteria crit=session.createCriteria(Articulo.class);
       if(filtros!=null){//propertyName,valor,modo de comparacion
          Iterator<Map.Entry<String, String>> iterator = filtros.entrySet().iterator();
          while (iterator.hasNext()) {
             Map.Entry<String, String> entry = iterator.next();
             crit.add(Restrictions.like(entry.getKey(), entry.getValue(), MatchMode.START));
           }
       }

       if(sortField==null){
          sortField="nomArticulo";//campo por default por el cual se ordena
       }

       crit.setFirstResult(first).setMaxResults(pagesize);
       if(sortOrder){//orden ascendente
          crit.addOrder(Order.asc(sortField));
       }else{
          crit.addOrder(Order.desc(sortField));
       }

       crit.setProjection(Projections.distinct(
             Projections.projectionList().add(Projections.id())
                   .add(Projections.property("nomArticulo"))
             ));
       List list=crit.list();
       List idlist=new ArrayList<Long>();
       for (Iterator it = list.iterator(); it.hasNext();) {
          Object[] object =(Object[]) it.next();
```

```
            idlist.add((Long)object[0]);
        }

        if(idlist.size()>0){
            crit=session.createCriteria(Articulo.class);
            crit.add(Property.forName("id").in(idlist));
            if(loadAsociacion!=null ){
                Iterator<Map.Entry<String, String>> iterator = loadAsociacion.entrySet().iterator();
                while (iterator.hasNext()) {
                    Map.Entry<String, String> entry = iterator.next();
                    crit.setFetchMode(entry.getValue(), FetchMode.JOIN);
                }
            }
            if(sortOrder){//orden ascendente
                crit.addOrder(Order.asc(sortField));
            }else{
                crit.addOrder(Order.desc(sortField));
            }
        }
        else{
            return new ArrayList();
        }
        lista=crit.setResultTransformer(CriteriaSpecification.DISTINCT_ROOT_ENTITY).list();
        tx.commit();
    }
    finally{
        session.close();
    }
    return lista;
}

@Override
public int entidadesTotal() {
    int intResult = DataAccessUtils.intResult(ht.find("SELECT COUNT (a.id) FROM Articulo a"));
    //System.out.println("Total de registros: "+intResult);
    return intResult;
}

@Override
public List<Articulo> datosAutocomplete(String findByCampo, String matchCampo) {
    //Busqueda por el campo findByCampo=nombre_articulo
    String query="FROM Articulo a where a."+findByCampo+" like :match";
```

```
        return ht.findByNamedParam(query, "match", matchCampo+'%');
    }

    @Override
    public List<Articulo> filtrarAsociacion(String asociacion, Long id) {
        DetachedCriteria crit=DetachedCriteria.forClass(Articulo.class);
        //asociacion=profesores
        crit.createAlias(asociacion,"p");
            crit.add(Restrictions.eq("p.id", id));
        crit.addOrder(Order.asc("nomArticulo"));
        return ht.findByCriteria(crit);
    }

    @Override
    public List<IdCampo> autocomplete(String findbycampo, String match) {
        String query="select NEW
com.marroquin.gestion.curricula.dominio.IdCampo(p.id,p."+findbycampo+") FROM Articulo p where
p."+findbycampo+" like :match";
        return ht.findByNamedParam(query, "match", match+'%');
    }
}
```

En la Figura 38 se muestra el paquete com.marroquin.gestion.curricula.datamodel en el cual contiene las clases que sirven para conectar las tablas de las vistas en despliegues de listas.

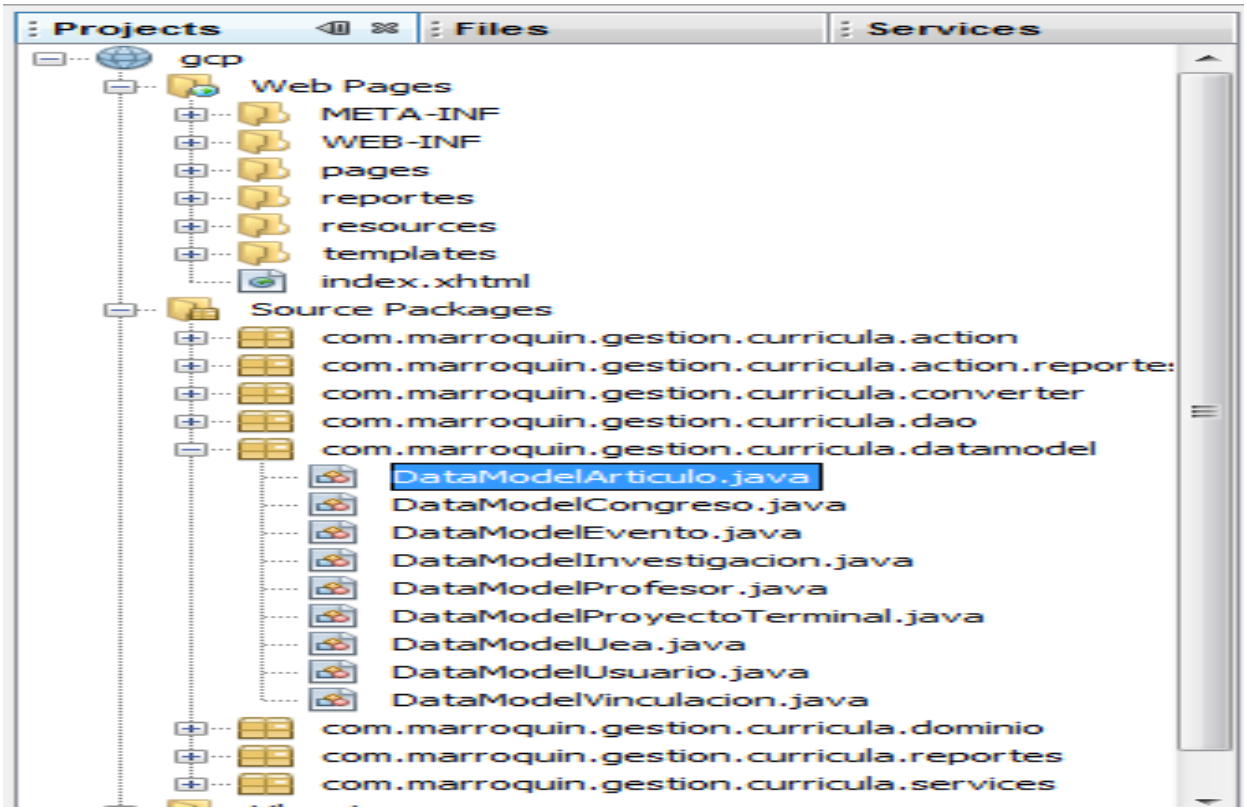


Figura 38

```
@Component("dataModelArticulo")
public class DataModelArticulo extends LazyDataModel<Articulo> {
   @Autowired
   private ArticuloService dao;
   private Map<String,String> loadAsociacion;
   boolean so;

   @Override
   public List<Articulo> load(int first, int pageSize, String sortField, SortOrder sortOrder, Map<String,
String> filters) {
      if(getRowCount() <= 0){
            setRowCount(dao.entidadesTotal());
      }
      setPageSize(pageSize);
      //Ordenar asc,desc por el campo sortField
      if(sortOrder==SortOrder.DESCENDING){
         so=false;//orden descendente
      }else{
```

```
            so=true;//ascendente
        }
        return dao.lazyLoanding(first, pageSize, sortField, so, filters, loadAsociacion);
    }

    public void setLoadAsociacion(Map<String, String> loadAsociacion) {
        this.loadAsociacion = loadAsociacion;
    }
}
```

En la Figura 39 se muestra el paquete com.marroquin.gestion.curricula.dominio en el cual contiene las clases de los objetos persistentes con la base de datos

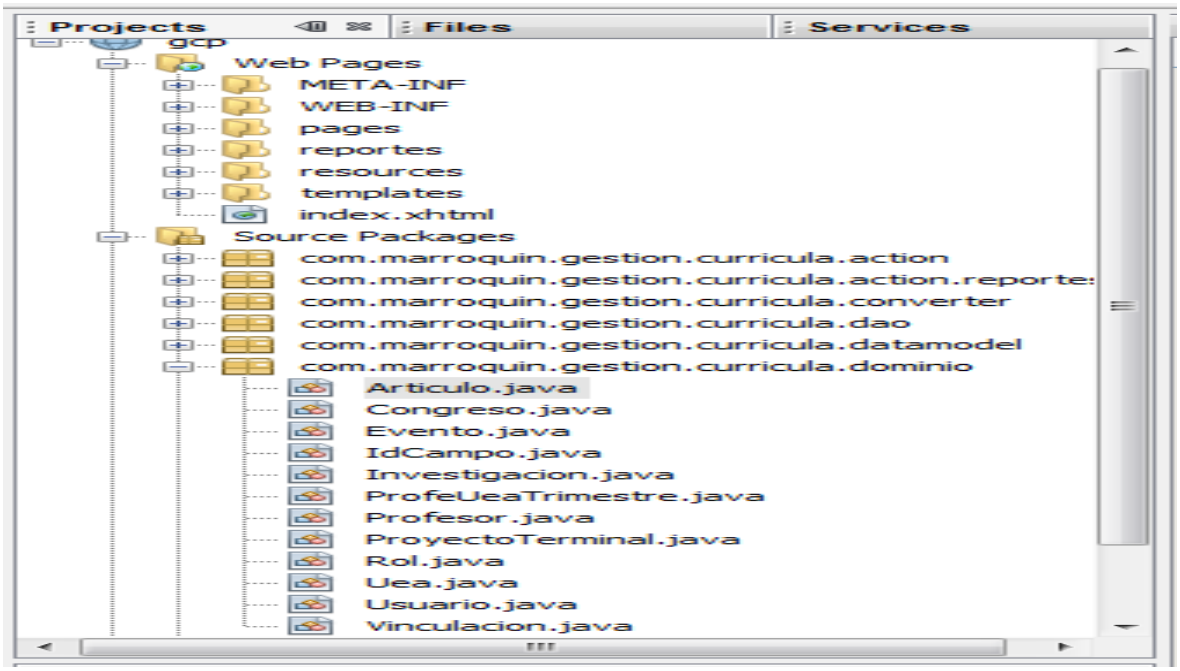


Figura 39

```
@Entity
@org.hibernate.annotations.Entity(dynamicUpdate=true)
@Table(name="articulos"
    ,catalog="gcp_db"
)
public class Articulo  implements java.io.Serializable {
    private Long id;
    private String descripcion;
    private Date fecha;
    private String institucion;
```

```
 private String issn;
 private String nivel;
 private String nomArticulo;
 private String nomRevista;
 private Set<Profesor> profesoreses = new HashSet<Profesor>(0);
 public Articulo() {  }
 public Articulo(String descripcion, Date fecha, String institucion, String issn, String nivel,
       String nomArticulo, String nomRevista, Set<Profesor> profesoreses) {
  this.descripcion = descripcion;
  this.fecha = fecha;
  this.institucion = institucion;
  this.issn = issn;
  this.nivel = nivel;
  this.nomArticulo = nomArticulo;
  this.nomRevista = nomRevista;
  this.profesoreses = profesoreses;
}

public Articulo(String institucion, String nombre_revista, String nombre_articulo,
       String descripcion, Date fecha, String nivel, String issn) {
   this.institucion = institucion;
   this.nomRevista = nombre_revista;
   this.nomArticulo = nombre_articulo;
   this.descripcion = descripcion;
   this.fecha = fecha;
   this.nivel = nivel;
   this.issn = issn;
}

 @Id @GeneratedValue(strategy=IDENTITY)

@Column(name="id", unique=true, nullable=false)
public Long getId() {
   return this.id;
}

public void setId(Long id) {
   this.id = id;
}

@Column(name="descripcion", length=65535)
public String getDescripcion() {
```

```
    return this.descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}
@Temporal(TemporalType.DATE)
@Column(name="fecha", length=10)
public Date getFecha() {
    return this.fecha;
}

public void setFecha(Date fecha) {
    this.fecha = fecha;
}

@Column(name="institucion", length=150)
public String getInstitucion() {
    return this.institucion;
}

public void setInstitucion(String institucion) {
    this.institucion = institucion;
}

@Column(name="issn", length=50)
public String getIssn() {
    return this.issn;
}

public void setIssn(String issn) {
    this.issn = issn;
}

@Column(name="nivel", length=50)
public String getNivel() {
    return this.nivel;
}

public void setNivel(String nivel) {
    this.nivel = nivel;
}
```

```
    @Column(name="nom_articulo", length=100)
    public String getNomArticulo() {
        return this.nomArticulo;
    }

    public void setNomArticulo(String nomArticulo) {
        this.nomArticulo = nomArticulo;
    }

    @Column(name="nom_revista", length=100)
    public String getNomRevista() {
        return this.nomRevista;
    }

    public void setNomRevista(String nomRevista) {
        this.nomRevista = nomRevista;
    }
@ManyToMany(cascade=CascadeType.ALL, fetch=FetchType.LAZY)
    @JoinTable(name="profesores_has_articulos", catalog="gcp_db", joinColumns = {
        @JoinColumn(name="articulo_id", nullable=false, updatable=false) }, inverseJoinColumns = {
        @JoinColumn(name="profesor_id", nullable=false, updatable=false) })
    public Set<Profesor> getProfesoreses() {
        return this.profesoreses;
    }

    public void setProfesoreses(Set<Profesor> profesoreses) {
        this.profesoreses = profesoreses;
    }

    public List<Profesor> profesoresesToList(){
        return new ArrayList<Profesor>(this.profesoreses);
    }

    @Override
    public int hashCode() {
        int hash = 3;
        hash = 59 * hash + (this.id != null ? this.id.hashCode() : 0);
        return hash;
    }

    @Override
```

```
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Articulo other = (Articulo) obj;
    if (this.id != other.id && (this.id == null || !this.id.equals(other.id))) {
        return false;
    }
    return true;
}}
```

En la Figura 40 se muestra el paquete com.marroquin.gestion.curricula.reportes en el cual contiene el codigo fuente de la generacion de reportes

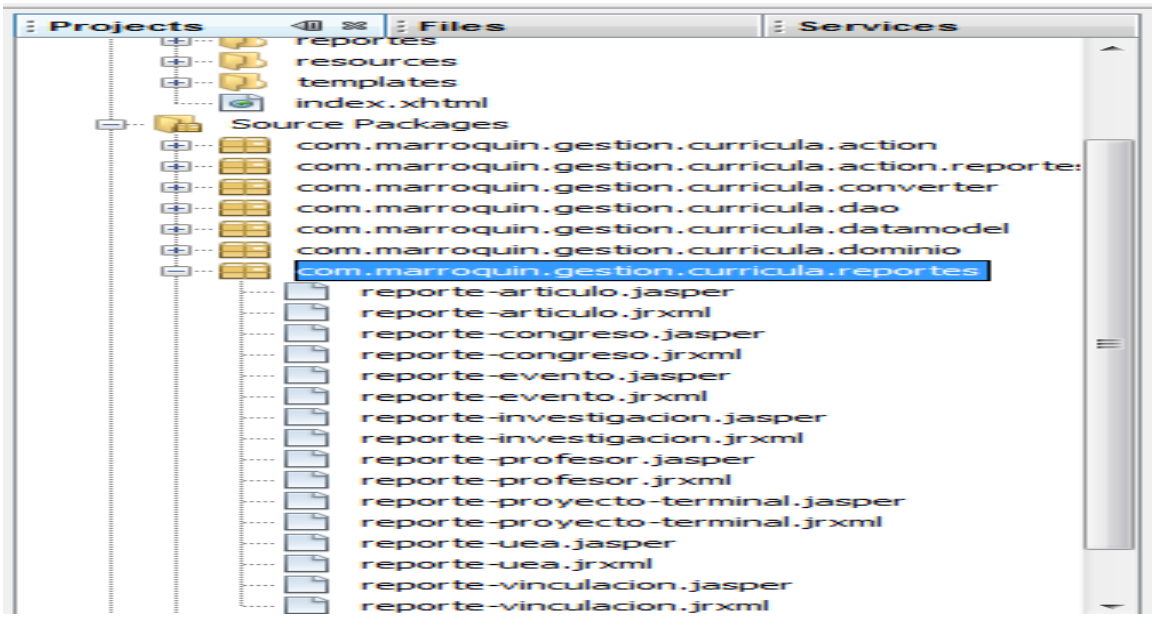


Figura 40

En la Figura 41 se muestra el paquete com.marroquin.gestion.curricula.service contiene las clases que sirven para la abstraccion que se crea entre la capa de presentacion y la capa de negocio, agrupando la funcionalidad de la capa de negocio para ser expuesta en la capa de presentacion.

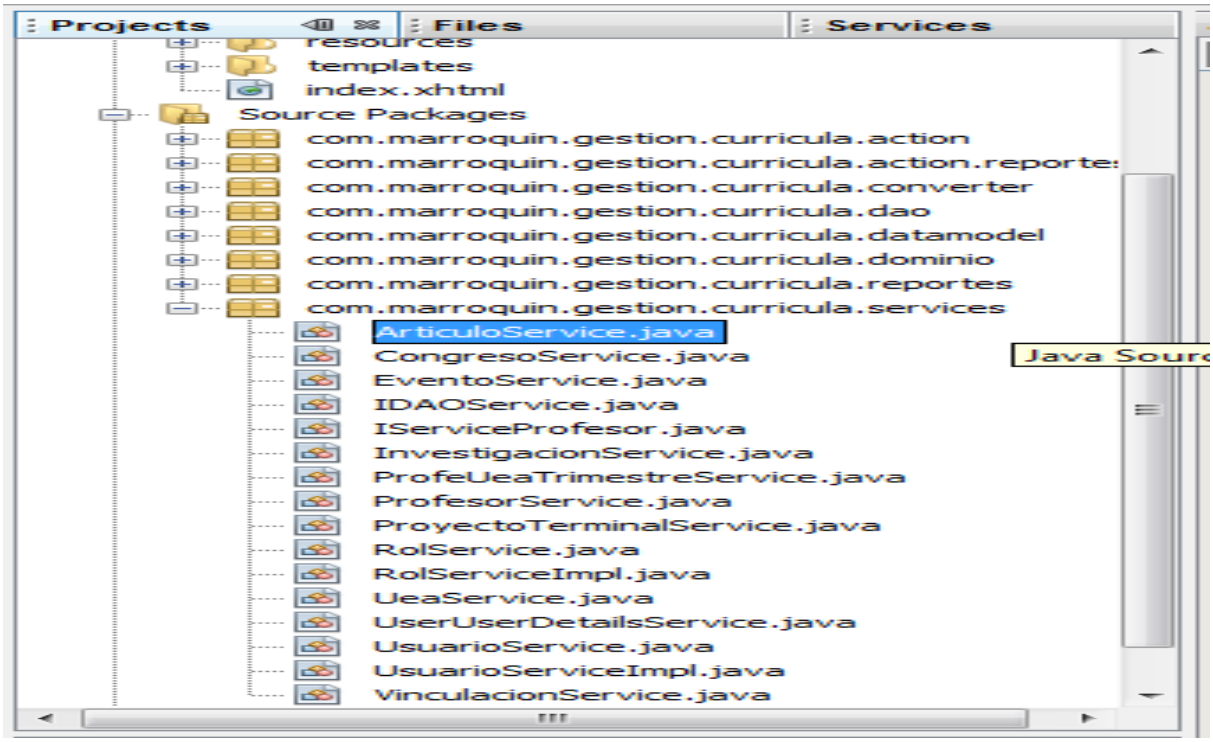


Figura 41

```
@Service("articuloService")
public class ArticuloService implements IDAOService<Articulo>,Serializable {
   @Autowired
   private ArticuloDAO dao;

   @Transactional
   @Override
   public void actualizar(Articulo entity) {
      dao.actualizar(entity);
   }

   @Transactional
   @Override
   public void eliminar(Articulo entity) {
      dao.eliminar(entity);
   }
```

```
    @Transactional
    @Override
    public long guardar(Articulo entity) {
        return dao.guardar(entity);
    }

    @Override
    public List<Articulo> obtenAll(List<String> cargaAsociacion) {
        return dao.obtenAll(cargaAsociacion);
    }

    @Override
    public Articulo obtener(long id, List<String> cargaAsociacion) {
        return dao.obtener(id,cargaAsociacion);
    }

    @Override
    public List<Articulo> findByFecha(Date rangoInferior, Date rangoSuperior,List<String>
cargaAsociacion) {
        return dao.findByFecha(rangoInferior, rangoSuperior,cargaAsociacion);
    }

    @Override
    public List<Articulo> lazyLoanding(int first, int pagesize, String sortField,boolean sortOrder,
            Map<String,String>filtros,Map<String,String> loadAsociacion) {
        return dao.lazyLoanding(first, pagesize, sortField, sortOrder, filtros, loadAsociacion);
    }

    @Override
    public int entidadesTotal() {
      return dao.entidadesTotal();
    }

    @Override
    public List<Articulo> datosAutocomplete(String findByCampo, String matchCampo) {
        return dao.datosAutocomplete(findByCampo, matchCampo);
    }

    @Override
    public List<Articulo> filtrarAsociacion(String asociacion, Long id) {
        return dao.filtrarAsociacion(asociacion, id);
```

```
    }

    @Override
    public List<IdCampo> autocomplete(String findbycampo, String match) {
        return dao.autocomplete(findbycampo, match);
    }
}
```

CONCLUSIONES

Al finalizar este proyecto se logro comprender mejor la arquitectura MVC (modelo vista controlador) utilizado en la implementación de la aplicación de este proyecto terminal.

El proyecto se dividió en módulos que representa cada capa de la funcionalidad de la arquitectura MVC, que hoy en día a nivel desarrollo es la más utilizada, sirviendo como base para el desarrollo del lenguaje JAVA EE.

Para la implementación se utilizaron 3 frameworks, los cuales fueron Spring, Hibernate y JSF bajo la implementación de PrimeFaces.

Cada uno de los framworks mencionados trabajan en las diferentes capas de la arquitectura MVC por ejemplo PrimeFaces trabaja en la capa de la vista permitiendo la creación de las interfaces de usuario de una manera más amigable para el usuario, Spring se utilizo como contenedor de objetos ya que es el encargado de proporcionar a la aplicación los objetos que se utilizan en los controladores evitando crear directamente cada objeto en el código de la aplicación, Hibernet se utilizo para persistir los objetos en la Base de Datos permitiendo trabajar de manera orientada a objetos con la Base de Datos.

También se utilizo MySQL como base de datos del sistema y la herramienta MYSQL WorkBench para el diseño del esquema de la Base de Datos.

En general los Sistemas de Información facilitan el control y manejo de cantidades grandes de información dando una optimización y alto rendimiento en la obtención de recursos dentro del ámbito empresarial.

El Sistema de Gestión de Currícula sirvió para poner en práctica los conocimientos adquiridos durante la licenciatura, adquiriendo bases solidas de implementación y desarrollo en Sistemas de Información al utilizar los frameworks que actualmente son unos de los más utilizados en el ámbito laboral.

Bibliografía

J. López, “Sistema de gestión de conocimiento para el apoyo a la actualización curricular de la ECCI” [En Línea]. Disponible: http://telecsys.unad.edu.co/documentos/revista%20No.6/articulo%201-6.pdf

J. Hernández, Diseño de un sistema para la Gestión de Información Curricular de los profesores en la Universidad de las Ciencias Informáticas” [En Línea]. Disponible: http://www.fec.uh.cu/CUGIO/1%20acciones/ Proyectos-Protocolos/recibido%20dia%2014/Proyecto_Jorge%20Hdez.pdf

C. Cadena y V. Peña, “Sistema informático para manejo de currícula vitarum (CV)”, Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2008.

B. López, “Sistema de gestión de documentos electrónicos en una biblioteca digital”, Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F, México, 2010.

F. Ceballos, *Java2 Interfaces Graficas y Aplicaciones*, 3ra ed. D.F, México: Alfaomega, 2008

Prime Faces facilitando la creacion de aplicaciones wec con JSF 2.0 [En línea]. Disponible: http://www.slideshare.net/gus_farfan/primefaces-14115155

The Apache Software Foundation, Apache Tomcat, [En línea]. Disponible: http://tomcat.apache.org

Oracle Corporation, The JDBC Tutorial: Chapter 3 - Advanced Tutorial, [En línea]. Disponible: http://java.sun.com/developer/Books/JDBCTutorial

C.P. López, *MySQL para Windows y Linux*, 2da ed. D.F. México: Alfaomega, 2007