COMPONENTE SOFTWARE PARA LA MIGRACIÓN DE LA BASE DE DATOS DE REGISTRO ACADÉMICO A COLECCIONES MONGODB

SERGIO ANDRES ORTIZ MACHUCA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

COMPONENTE SOFTWARE PARA LA MIGRACIÓN DE LA BASE DE DATOS DE REGISTRO ACADÉMICO A COLECCIONES MONGODB

SERGIO ANDRES ORTIZ MACHUCA

TRABAJO DE GRADO PARA OPTAR EL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR
Ph.D. SONIA CRISTINA GAMBOA SARMIENTO

CODIRECTOR
M.Sc. CARLOS HUMBERTO CARREÑO DÍAZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

DEDICATORIA

Agradecimientos especiales

A Dios y a mis padres Victer Rosa Machuca y Luis Carlos Ortiz por el apoyo prestado durante toda mi carrera así mismo a toda mi familia:

A MSc Carlos Humberto Díaz por su valiosa orientación intelectual y fuerza motivacional en este proyecto.

A la Doctora Sonia Gamboa por la confianza prestada desde el inicio de este proyecto.

A las personas que de una u otra forma estuvieron apoyándonos en este camino, agradecemos con todo el cariño posible su colaboración.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

TITULO: COMPONENTE SOFTWARE PARA LA MIGRACIÓN DE LA BASE DE DATOS DE REGISTRO ACADÉMICO A COLECCIONES MONGODB[*]

AUTORES: SERGIO ANDRES ORTIZ[**]

PALABRAS CLAVES: Migración, MongoDB, Servicio web, componente

DESCRIPCIÓN:

Los programas académicos de postgrado especialmente maestrías y doctorados deben proporcionar a sus estudiantes una información actualizada a cerca de todos sus procesos académicos, para lo cual es necesario visualizar todos los registros en la web de tal manera que haya una retroalimentación constante entre los actores y dichos procesos.

Para ello es importante disponer de ambientes virtuales con los cuales los actores de estos programas puedan interactuar entre ellos mismos, y fortalecer los escenarios académicos

El componente que se presenta en este proyecto tiene como objetivo migrar la información que se encuentra almacenada en el sistema de Registro Académico a la base de datos MongoDB, ya que esta base de datos servirá posteriormente para la realización del sistema de autoevaluación de la maestría de ingeniería de sistemas de la Universidad Industrial de Santander UIS.

Este componente permite extraer la información tanto de los estudiantes de la maestría de ingeniería de sistemas como la del programa académico, información que está almacenada en la base de datos de admisiones. El proyecto está basado en JEE6 (Java Enterprise Edition 6). Con la implementación de este componente se pretende mejorar y agilizar todos los procesos asociados al manejo de la información del posgrado, lo que le permitirá a la universidad hacer más eficaces los procesos relacionados con la Autoevaluación de la maestría de ingeniería de sistemas.

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisico Mecánicas. Escuela de Ingeniería de Sistemas. Director: Sonia Cristina Gamboa Sarmiengo, PhD. Codirector: Carlos Humberto Carreño Díaz, PhD.

ABSTRACT

TITLE: SOFTWARE COMPONENT FOR THE MIGRATION OF THE ACADEMIC REGISTRY DATABASE TO THE MONGODB COLLECTIONS[*]

AUTHOR: SERGIO ANDRES ORTIZ MACHUCA[**]

KEY WORDS: web service, Migration, MongoDB, component

DESCRIPTION:

Academic programs for graduate students, such as major´s degree, master's degree and PHDs, require access of accurate and updated information. Keeping record of information concerning author and academic process.

To adequately achive this goal its vital to create virtual environments in which the interaction between authors its allowed and the academic process could be constant.
The component of this project has as a main goal to migrate information archived in the academic registry to the MongoDB database, due to this will later be useful for the self-assesment system of the master's degree in system engeneering of the Industrial University of Santander (UIS).

This component allows to extract the information of both the students in the master's degree in system engeneering and the academic program, information which is stored in the admissions database. The project is based in JEE6 (Java Enterprise Edition 6). The implementation of this component pursues to improve and accelerate all the processes associated with the postgraduate degree imformation management, allowing the university to have more efficient processes related with the self-assesment of the master's degree in system engineering.

.

[*] Undergraduate thesis

[**] Faculty of Physicomechanical Engineering. School of System Engineering. Director: Sonia Cristina Gamboa Sarmiengo, PhD. Co: Carlos Humberto Carreño Díaz, PhD.

INTRODUCCIÓN

Durante los últimos años se ha podido notar que los programas académicos de postgrado requieren tener visibilidad en la web para facilitar la interacción con información actualizada acerca de sus propios programas de maestría y doctorado. Esta información es un valioso recurso en los programas académicos, a partir de la cual es posible encontrar patrones, tendencias u otros conocimientos que favorezcan los procesos de toma de decisiones, planificación y autoevaluación.

Este proyecto viabiliza la realización de futuros proyectos como el sistema de autoevaluación de la maestría de ingeniería de sistemas, de manera que el sistema pueda contar con información consistente y actualizada. En primera instancia se trabajará en la Escuela de Ingeniería de Sistemas e informática de la Universidad Industrial de Santander, específicamente en la Maestría en Ingeniería de sistemas e informática. Sin embargo, se busca que el presente proyecto pueda ser llevado a distintos programas académicos en busca de consolidar los diferentes programas académicos de postgrado de la UIS.

PRESENTACIÓN DEL PROYECTO

ESPECIFICACIONES DEL PROYECTO

Titulo
COMPONENTE SOFTWARE PARA LA MIGRACIÓN DE LA BASE DE DATOS DE REGISTRO ACADÉMICO A COLECCIONES MONGODB

Director del proyecto:
Nombre: Ph.D Sonia Cristina Gamboa Sarmiento
Institución: Universidad Industrial de Santander

Codirector del proyecto:
Nombre: M.Sc. Carlos Humberto Carreño Díaz
Institución: Universidad Industrial de Santander

Estudiantes:
Nombre: Sergio Andres Ortiz Machuca
Código: 2050431
Carrera: ingeniería de sistemas

Entidades interesadas en el proyecto:
Escuela de Ingeniería de Sistemas
Universidad Industrial de Santander

1. OBJETIVOS

1.1. OBJETIVO GENERAL

Diseñar e implementar un componente software que permita interacción y migración de la base de datos relacional del sistema de registro académico con la base de datos no relacional MongoDB del Sistema de autoevaluación de programas de postgrado.

1.2. OBJETIVOS ESPECÍFICOS

- Especificar los requerimientos principales para la interacción entre los dos sistemas incluyendo los actores, escenarios y servicios que prestarán.

- Analizar las estructuras de datos relacionales del sistema de registro académico para la migración sus registros a la base de datos no relacional MongoDB.

- Desarrollar una interfaz para administrar la información que se ha migrado, de tal manera que se pueda consultar y modificar la información de usuarios del sistema

- Realizar la documentación de los componentes software diseñados, el código fuente del sistema, así como de las pruebas de funcionalidad y usabilidad de los servicios implantados en el sistema de autoevaluación.

1.3. PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

Este proyecto pretende diseñar e implementar un componente software mediante el cual sea posible extraer la información registrada en el sistema de Admisiones y Registro Académico de la UIS, según requerimientos específicos del sistema de Autoevaluación propuesto para un programa de postgrado.

En la actualidad la Universidad Industrial de Santander – UIS – tiene vinculada la información relacionada con los programas académicos en un sistema de información interno principalmente orientado al registro y control financiero y académico. Este proyecto busca la integración de los servicios ofrecidos por un

sistema de información para Autoevaluación de programas de postgrado con el sistema de admisiones y registro académico de la Universidad Industrial de Santander, de manera que se pueda contar con información consistente y actualizada.

2. MARCO TEÓRICO

2.1. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS.

APLICACIÓN: Programa preparado para una utilización específica, como el pago de nóminas, formación de un banco de términos léxicos, etc.

EJB**:** Es un modelo de componentes, del lado del servidor. Los EJB (Enterprise java beans por sus siglas en inglés) están basados en componentes, lo cual permite su reutilización, se basa en el trabajo del contenedor (el contenedor es un programa java que corre en el servidor

INTERFAZ: Parte de un programa que permite el flujo de información entre un usuario y la aplicación, o entre la aplicación y otros programas o periféricos. Esa parte de un programa está constituida por un conjunto de comandos y métodos que permiten dichas intercomunicaciones.

JAVA: Es un lenguaje de programación desarrollado por James Gosling de Sun Microsystems y fue publicado en 1995. Es un lenguaje compilado e interpretado, y esto es lo que permite que pueda ser ejecutado en varios entornos y realizar cualquier tipo de programa.

JAVASCRIPT: Es un lenguaje de programación orientado a objetos y prototipos, se utiliza principalmente en la interface del cliente, implementado como parte de un navegador web

JSF y EJB funcionan mejor juntos, los creadores de ambos modelos proporcionan puntos de extensión estándar para permitir la integración con otros marcos.

Maestría en Ingeniería de Sistemas e Informática :La Maestría de Ingeniería de Sistemas e Informática es una maestría de investigación que está estructurada de forma tal que los estudiantes puedan desarrollarse en cualquiera de las áreas que se abordan desde los grupos de investigación: modelos y simulación, ingeniería biomédica, gestión y optimización de sistemas, tecnologías de la

información, ingeniería de software, ingeniería telemática y sistemas inteligentes o inteligencia artificial y sistemas de conocimiento experto [1]

RF: abreviatura de Requerimientos Funcionales

RNF: abreviatura de Requerimientos no Funcionales

2.2. DESARROLLO BASADO EN COMPONENTES

Para empezar hablando de este tipo de desarrollo, primero se debe tener presente la definición de componente: Se define como “La una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio”[Szyperski y Pfister,1998][2]

El desarrollo de software basado en componentes es un concepto de desarrollo debido a que utiliza técnicas de software, que describe y construye sistemas abiertos y distribuidos mediante el empalme de partes de software

2.2.1. **Servicios proporcionados por el contenedor EJB**

Los componentes son almacenados y ejecutados en el EJB (Enterprise java beans) donde son empaquetados y a su vez heredan los servicios. Entre los servicios más importantes se encuentran:

- Abre y cierra las transacciones asociadas a las llamadas de los métodos bean
- Corrobora los accesos y los permisos para poder acceder a los métodos bean.
- llama simultánea a un mismo bean desde múltiples clientes.
- proporciona comunicación entre el cliente y el bean en máquinas distintas.
- Gestiona automáticamente múltiples recursos como base de datos o fuente de datos de aplicaciones
- sincronización entre los datos del bean y tablas de una base de datos.
- manejo de Java Message Service (JMS).

---

[1] • http://cormoran.uis.edu.co/eisi/eisi.jsp?IdServicio=S78

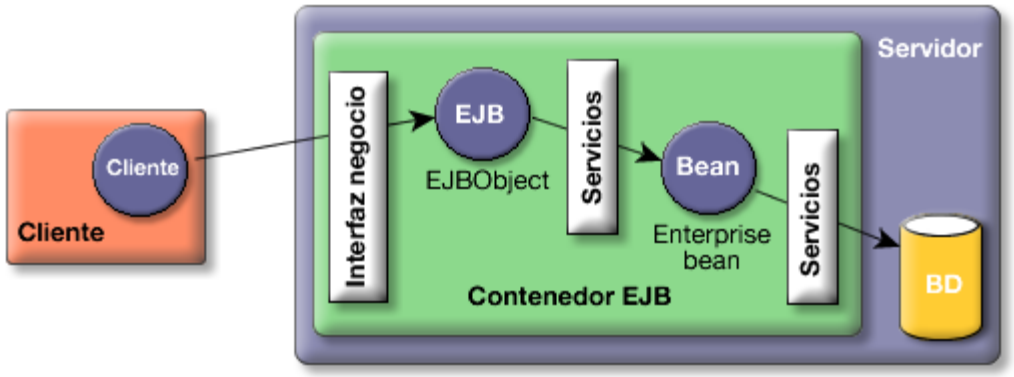
[2] [Szyperski, 1998] Szyperski, C. (1998). Component Software. Beyond Object-Oriented Programming. Addison-Wesley.

2.2.2. **Funcionamiento de los componentes EJB**

Se basa fundamentalmente en el trabajo del contenedor EJB. Este contenedor es un programa Java que corre en el servidor y que posee clases y objetos necesarios para el funcionamiento de los Enterprise bean.

En la figura 1 se puede ver una representación del funcionamiento básico de los Enterprise Beans. El cliente realiza la petición al bean, y el servidor donde se encuentra el bean se halla ejecutándose en máquinas virtuales en el mismo periodo de tiempo, algo importante para resaltar es que el cliente nuca se comunica directamente con el Enterprise bean, sino que el contenedor EJB proporciona un *EJBObject* que hace de interfaz.

Figura 1 Componentes Enterprise JavaBean



Fuente: EJB sesión 01[3]

2.3. MONGODB

Es un sistema de base de datos NoSQL, el cual es orientado a documentos bajo el concepto de código abierto. Es un sistema que permite guardar los datos en documentos tipo BSON con un esquema dinámico, realizando la integración de los datos en aplicaciones fáciles y rápidas.

MogndoDB es la base de datos que empresas como Cisco, IBM, eBay, Forbes están utilizando por su desempeño, agilidad y escalabilidad. Es una base de datos ágil que permite cambiar rápidamente esquemas cuando las aplicaciones van evolucionando eventualmente con el pasar del tiempo y desarrollo,

[3] http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01-apuntes.htm

proporcionando funcionalidad que los desarrolladores esperan de las bases de datos tradicionales.

2.4. CARACTERÍSTICAS DE MONGODB

Como se mencionó anteriormente MongoDB es una base de datos basada en documentos, el cual permite tener como clave/valor un documento,

Algunas de sus características son:

- Distribuido: son sistemas a menudo divididos en varias máquinas para cooperar en grupos y ofrecer a los clientes una mayor agilidad en el procesamiento de datos[4]
- Escalabilidad Horizontal: se añaden nodos dinámicamente y no existen límites definidos para el número de máquinas que se pueden añadir
- Tolerancia a fallos y Redundancia.
- No genera cuellos de botella.
- No utiliza esquema: los clientes pueden almacenar datos como deseen, sin tener que cumplir con algunas normas predeterminadas.
- Modelos de datos no relacionales: los modelos de datos varían dependiendo los requerimientos del negocio, estos no son relacionales por lo tanto son más flexibles en el momento de diseñar[5]

2.5. MORPHIA

Morphia es una biblioteca ligera de código abierto diseñada para reducir la brecha entre los controladores de MongoDB y los objetos POJO de Java. Puede ser una alternativa a SpringData si usted no está usando Spring Framework para interactuar con MongoDB. Teniendo en cuenta que MongoDB no almacena datos en tablas como lo hacen en las base de datos relacionales sino en colecciones[6], no posee un mecanismo que corresponda con la forma en como las aplicaciones que utilizan JPA (Java Persistence API) almacenan los datos.

---

[4] CHANG, F. et al. \Bigtable: A Distributed Storage System for Structured Data". In: ACM Transactions on Computer Systems (2008), p. 1-26

[5] NÄSHOLM. Petter, Extracting Data from NoSQL Databases. A Step towards Interactive Visual Analysis of NoSQL Data Febrero 2012. [En linea] <http://publications.lib.chalmers.se/records/fulltext/155048.pdf >[Citado en 20 de abril de 2013]

[6] En mongodb la estructura de datos básica son denominadas colecciones, lo que corresponde a las "tablas" de una base de datos relacional y los registros son denominados documentos

De esta manera Morphia actúa como ORM (Object Relational Mapping) para MongoDB, permitiendo buscar objetos Java para facilitar su almacenamiento. Para ello Morphia utiliza anotaciones de manera tal que no haya archivos XML que gestionar o actualizar. Mediante su uso es posible buscar objetos POJO de Java para almacenarlos en dicha base datos, permitiendo indicar que una clase se almacena en determinada colección e incluso soporta colecciones de datos polimórficas.

Una de las mejores características es que se puede utilizar para indexar automáticamente sus colecciones en función de sus anotaciones lo que simplifica enormemente la implementación y el despliegue de cambios.

3. ESPECIFICACIÓN DE REQUERIMIENTOS PARA LA INTERACCIÓN DE LOS SISTEMAS DE AUTOEVALUACIÓN Y REGISTRO ACADÉMICO

3.1. PROPÓSITO

En este documento se pueden observar los requerimientos generales y específicos para la migración de información a la base de datos MongoDB que servirá para la Autoevaluación de postgrados de la escuela de ingeniería de sistemas de la universidad industrial de Santander, teniendo en cuenta las especificaciones del cliente con el fin de satisfacer sus necesidades. El documento va dirigido a los diseñadores y desarrolladores que se encargarán de llevar a cabo el sistema de información.

3.2. ÁMBITO DEL SISTEMA.

Se ha determinado implementar un componente software de migración que sirva de interfaz entre el sistema de información de admisiones y registro académico, y el sistema de información para la Autoevaluación de programas de postgrados de la universidad industrial de Santander. El componente software tendrá las siguientes funciones:

- Manipular de los datos dependiendo del tipo de usuario
- interacción entre los dos sistemas, determinando los actores, escenarios y servicios que prestarán.
- Validar la funcionalidad y usabilidad de los servicios implantados en un portal web.
- Garantizar la seguridad y confidencialidad de los datos

3.3. DESCRIPCIÓN GENERAL

3.3.1. **Perspectiva del Producto**

El proyecto pretende implementar una migración entre el sistema de información de admisiones de la Universidad Industrial de Santander con la base de datos MongoDB, la migración sustrae toda la información que se encuentra en la base de datos del sistema de admisiones.

3.3.2. **Funciones del Producto.**

La interfaz cuenta principalmente con 2 funciones

- Extraer la información de la base de datos del sistemas de admisiones
- Registrar los datos en el sistema de información

Estas dos funciones quedarán registradas en una base de datos para su posterior consulta o posible modificación

3.3.3. **Características de los Usuarios.**

Existirá un súper usuario o administrador el cual tendrá todos los privilegios y características que le permitan administrar completamente la interfaz y estará a cargo del registro de los datos en el sistema.

3.3.4. **Restricciones.**

- Lenguaje y tecnologías en uso JAVA.
- Se desarrollará utilizando NetBeans IDE.
- La base de datos se instalará en MongoDB
- La aplicación requiere de un servidor de base de datos.
- A la información suministrada sólo podrá acceder el usuario registrado.

3.3.5. **Suposiciones y Dependencias.**

Se asume que los requisitos aquí descritos son estables.

Es un desarrollo web, el cual está diseñado para funcionar en cualquier equipo, siempre que se cuente con una conexión a Internet.

3.4. REQUERIMIENTOS FUNCIONALES

Tabla 1 Registro Y Administración De Usuarios En El Sistema

Identificación	**Rf01**
Nombre	REGISTRO Y ADMINISTRACIÓN DE USUARIOS EN EL SISTEMA
Prioridad	ALTA
Descripción	El Súper Usuario del sistema puede administrar la interfaz; crear, modificar, consultar y eliminar tanto usuarios como registros del sistema.

Fuente: Autor

Tabla 2 Autentificación

Identificación	**RF02**
Nombre	AUTENTIFICACIÓN
Prioridad	ALTA
Descripción	El sistema puede ser consultado por el Súper usuario que esté registrado y autenticado

Fuente: Autor

Tabla 3 Consultar Información Del Sistema De Información De Admisiones UIS

Identificación	**RF03**
Nombre	CONSULTAR INFORMACIÓN DEL SISTEMA DE INFORMACIÓN DE ADMISIONES UIS
Prioridad	ALTA
Descripción	EL administrador en cualquier momento puede consultar la base de admisiones uis

Fuente: Autor

3.5. REQUERIMIENTOS NO FUNCIONALES

Tabla 4 Interfaz Del Sistema

Identificación	**RNF01**
Nombre	INTERFAZ DEL SISTEMA
Características	Se presentará una interfaz sencilla para que sea de fácil manejo para el administrador
Prioridad	ALTA
Descripción	Tendrá una interfaz de uso sencillo e intuitivo.

Fuente: Autor

Tabla 5 Documentación Del Sistema

Identificación	**RNF02**
Nombre	DOCUMENTACIÓN DEL SISTEMA
Características	La interfaz debe presentar una Documentación para que al usuario del sistema se le facilite el trabajo.
Prioridad	ALTA
Descripción	La interfaz debe estar complementada de un buen sistema de ayuda.

Fuente: Autor

Tabla 6 Mantenimiento Del Sistema

Identificación	**RNF03**
Nombre	MANTENIMIENTO DEL SISTEMA
Características	La aplicación deberá mantener manual de usuario para facilitar el mantenimiento.
Prioridad	ALTA
Descripción	La aplicación deberá tener documentación de fácil acceso, entendible y actualizada que permita realizar las operaciones de mantenimiento en el menor tiempo y con la mayor eficacia.

Fuente: Autor

Tabla 7 Desempeño Del Sistema

Identificación	**RNF04**
Nombre	DESEMPEÑO DEL SISTEMA
Características	El sistema debe brindar un buen desempeño en cuanto a los datos almacenados, ofreciendo confiabilidad.

Prioridad	ALTA
Descripción	Brindar un buen desempeño a los diferentes usuarios; para que la información almacenada o registros realizados puedan ser consultados y actualizados permanente y simultáneamente sin que afecte el tiempo de respuesta.

Fuente: Autor

Tabla 8 Confiabilidad Del Sistema

Identificación	**RNF05**
Nombre	CONFIABILIDAD DEL SISTEMA
Características	El sistema tendrá que estar siempre en funcionamiento, ya que está diseñado para la carga de datos y consulta de los mismos.
Prioridad	ALTA
Descripción	La disponibilidad tiene que ser continua para el servicio de los usuarios, garantizando el servicio adecuado que ante cualquier falla en alguno de sus componentes contar con un plan de contingencia.

Fuente: Autor

Tabla 9 Seguridad De La Información

Identificación	**RNF06**
Nombre	SEGURIDAD DE LA INFORMACIÓN
Características	Ofrecer seguridad a los usuarios en cuanto a la información.
Prioridad	ALTA
Descripción	Proporcionar confianza al usuario respecto al sistema protegiendo la información y datos que se manejen mediante protocolos de seguridad.

Fuente: Autor

3.6. DESCRIPCIÓN DE ACTORES Y DEFINICIÓN DE CASOS DE USO

3.6.1. **CASOS DE USO**

Tabla 10 Actor Administrador

ACTOR	
Actor	Administrador
Casos de uso	Sincronizar registros, Administrar registros, Administrar usuarios
Tipo	Iniciador primario
Descripción	Este actor es la persona que administra, actualiza y también tiene acceso a la base de dato de admisiones

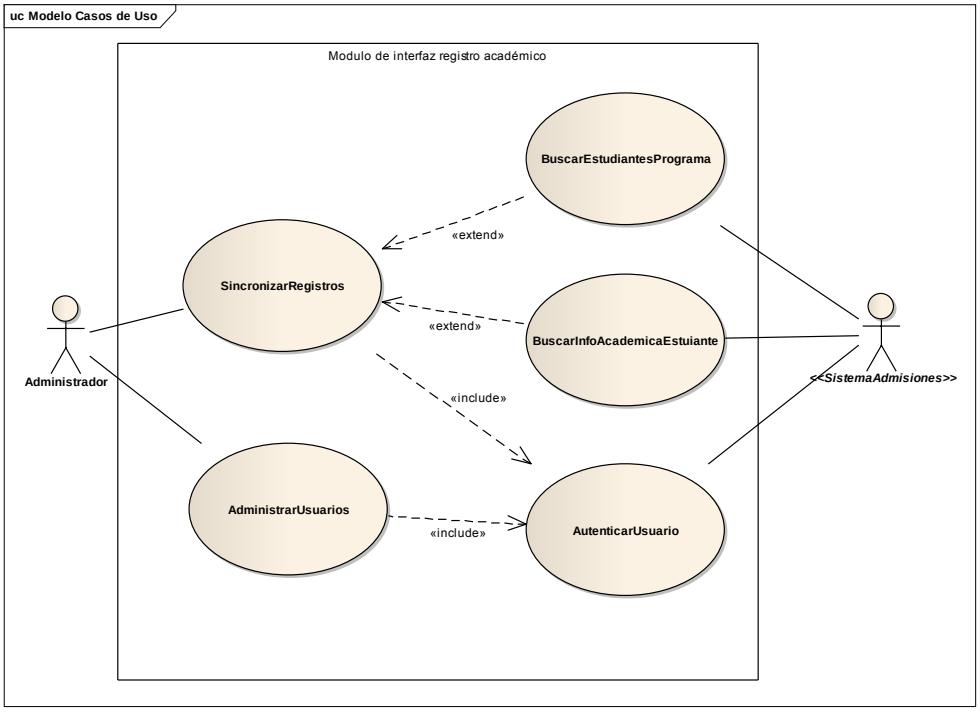
Fuente: Autor

Tabla 11 Actor Sistema Académico

ACTOR	
Actor	Sistema Admisiones
Casos de uso	Consultar BD
Tipo	Participante
Descripción	Este actor dado el caso puede actualizar ó sincronizar la información si previamente se programa alguna de estas dos acciones

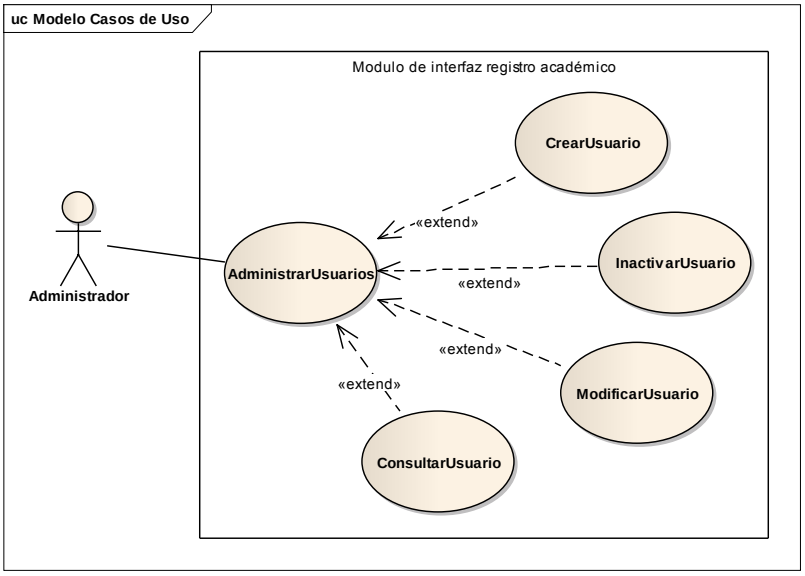
Fuente: Autor

Figura 2 Diagrama de Casos de Uso



Fuente: Autor

Figura 3 Diagrama de Casos de Uso Continuación



Fuente: Autor

Tabla 12 Crear usuario

CASOS DE USO		
Caso de Uso	Crear Usuario	
Actor(es)	Administrador	
Propósito	Registrarse para poder ingresar a las funciones que brinda el sistema	
Descripción	El sistema brinda la opción para que nuevos administradores puedan ingresar y utilizar el sistema, para ello cuenta con un registro donde un nuevo usuario debe ingresar sus datos personales como nombre, apellidos, correo electrónico, teléfono, sexo, fecha de nacimiento y proporcionar una contraseña.	
Precondición(es)	Ninguna	
Flujo Principal	**Acciones de Actor(es)**	**Respuestas del Sistema**
	Un futuro usuario se asigna y desea registrarse en el sistema. Dicho usuario llena el formulario el cual tendrá los datos principales para la información y acceso de cuenta Si la notificación fue satisfactoria el nuevo usuario ya puede ingresar al sistema, del caso contrario deberá rectificar los datos ingresados	El sistema valida y almacena la información suministrada por el nuevo usuario y envía una notificación haciéndole saber al nuevo usuario si el registro fue o no satisfactorio
Sub-flujos	ninguna	
Poscondición(es)	El sistema debe permitirle el ingreso al sistema si el usuario fue creado satisfactoriamente	

Fuente: Autor

Tabla 13 Inactivar usuario

CASOS DE USO		
Caso de Uso	Inactivar usuario	
Actor(es)	Administrador	
Propósito	Inactivar un usuario del sistema	
Descripción	Se desea inactivar a un usuario del sistema, quitándole todos los permisos que éste posee,	
Precondición(es)	El usuario debe estar registrado	
Flujo Principal	**Acciones de Actor(es)**	**Respuestas del Sistema**
	Un administrador desea inhabilitar a un usuario	El sistema muestra los usuarios que están registrados como administradores
	3.Eladministrador selecciona el usuario a desactivar y oprime el botón desactivar usuario	El sistema inhabilita al usuario
Sub-flujos	ninguna	
Poscondición(es)	El usuario queda desactivado	

Fuente: Autor

Tabla 14 Caso de uso Autentificar Usuario

CASOS DE USO		
Caso de Uso	Autentificar Usuario	
Actor(es)	Administrador	
Propósito	Autentificar usuario del sistema	
Descripción	El administrador del sistema desea autenticar un usuario que accede al sistema, para esto ingresa en el módulo autenticar usuario y verifica si el usuario existe y cuenta que si cuenta con los permisos necesario para acceder al sistema	
Precondición(es)	El usuario debe estar registrado	
Flujo Principal	**Acciones de Actor(es)**	**Respuestas del Sistema**
	1 El administrador del sistema desea autenticar un usuario, para esto pasa como parámetros de sistema el ID y el nombre al módulo autenticar usuario	2 El sistema recibe el ID y el nombre del usuario y hace la consulta a la base de datos y verifica si el usuario existe y que tipo de permisos posee
Sub-flujos	ninguna	
Poscondición(es)	El usuario habrá sido validado y se presentará la pantalla para elegir operaciones	

4. DISEÑO DE LOS COMPONENTES A DESARROLLAR

4.1. SELECCIÓN DE LA BASE DE DATOS A UTILIZAR

Dado que se trabajó utilizando una base de datos noSQL, debido a la flexibilidad que ella ofrece, se investigó y se decidió trabajar con la base de datos MongoDb, pues esta base de datos está orientada a documentos. MongoDb permite en dado caso crear nuevos campos, atributos, nuevas colecciones, sin afectar su funcionamiento y usabilidad.

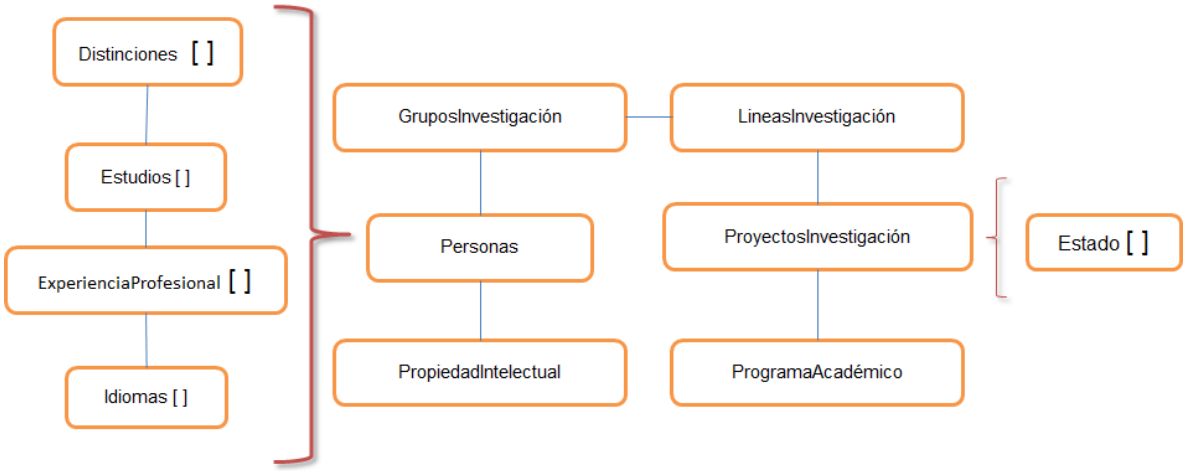
4.2. CARACTERIZACIÓN Y DISEÑO DE ENTIDADES

Para la elaboración de la base de datos se conformó un grupo de estudio conformado por 7 estudiantes de pregrado, un estudiante de postgrado, los directores del proyecto y un administrativo relacionado con la problemática de la Autoevaluación en la escuela de ingeniería de sistemas e informática de la UIS.

A partir de esto se determinaron las siguientes entidades iniciales como: Personas (que podrían ser administrativos, profesores o estudiantes), GruposInvestigacion (las cuales conforman la maestría), LineasInvestigación, ProgramaAcadémico y PropiedadIntelectual. Luego de determinar las entidades principales se hizo un proceso de revisión para determinar las principales características de cada una de ellas, donde se observó que no solamente existían estas entidades sino que existían además unas colecciones embebidas dentro de estas como por ejemplo: distinciones, estudios, experiencia Profesional, idiomas, y estados de los proyectos etc. En el anexo A se presenta un diccionario de datos de las diferentes tablas creadas

Como la representación de las bases de datos noSQL no se basan en el modelo entidad relación utilizados comúnmente, se realizó una revisión de las relaciones que tienen entre si y se planteó el siguiente modelo conceptual.

Figura 4 Modelo Conceptual de la Base de Datos



Fuente: Autor

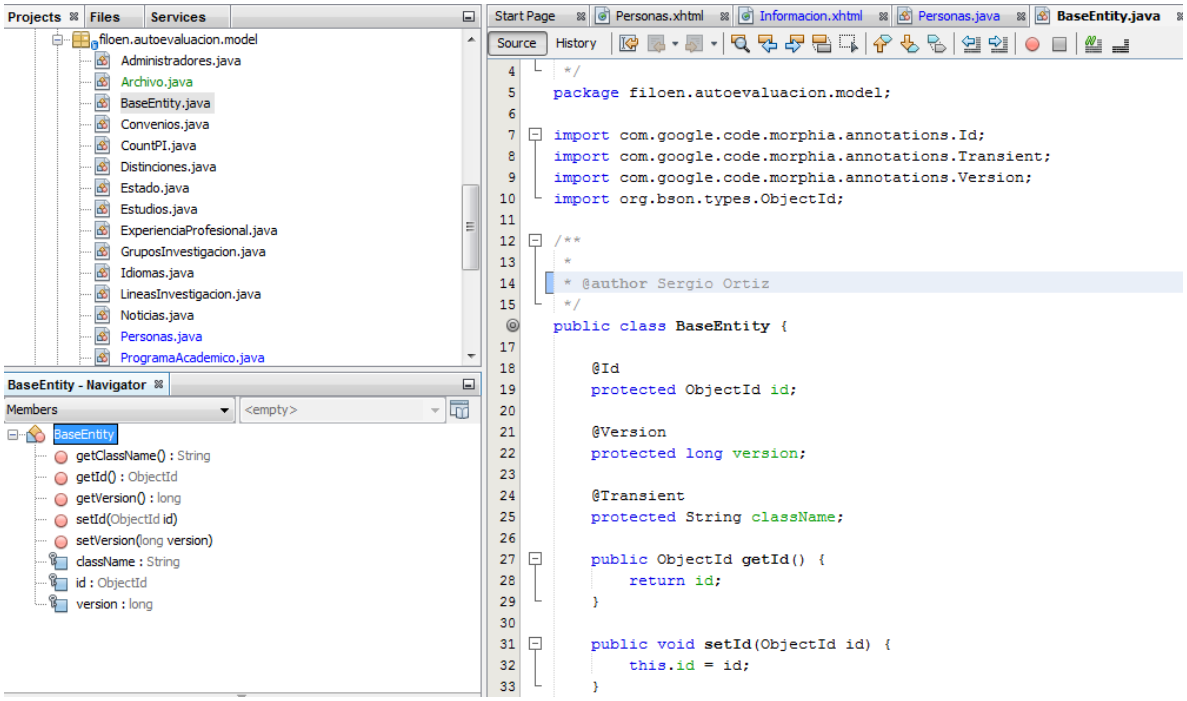
Donde se observan las relaciones que tiene la clase Personas y sus clases embebidas Distinciones, Estudios, ExperienciaProfesional e Idiomas, también se observa la relación que tiene Estado que es embebido de ProyectosInvestigación. Así se logró tener una mejor idea del diagrama de clases de la base de datos que el proyecto requería.

4.3. DISEÑO DE ENTIDADES UTILIZANDO *MORPHIA FRAMEWORK*

Por medio del uso de *Morphia Framework* se pueden definir entidades (@Entity), las cuales representan colecciones de la estructura de datos de MongoDB) como Personas, GruposInvestigacion, LineasInvestigacion, etc) y las transforma en una clase POJO (Plain Old Java Objects) cuyos atributos representen las propiedades de cada colección en la base de datos. Para el control del identificador principal y manejo de versiones de archivo, se creó una entidad abstracta denominada *BaseEntity* que permitiera heredar dichas propiedades, facilitando así la reducción de complejidad y de código, además de permitir el uso de algunos patrones de diseño utilizados en el desarrollo.

En la figura 5 se presenta la clase BaseEntity, en donde se observa el manejo de las anotaciones @Id y @version, de *Morphia Framework*.

Figura 5 Clase BaseEntity



Fuente: Autor

A continuación se puede observar parte del código utilizando donde se observan las propiedades de *@Entity* y *@Embedded*, las cuales permiten identificar si la clase representa una colección o una clase embebida.

Figura 6 Ejemplo Clase Personas

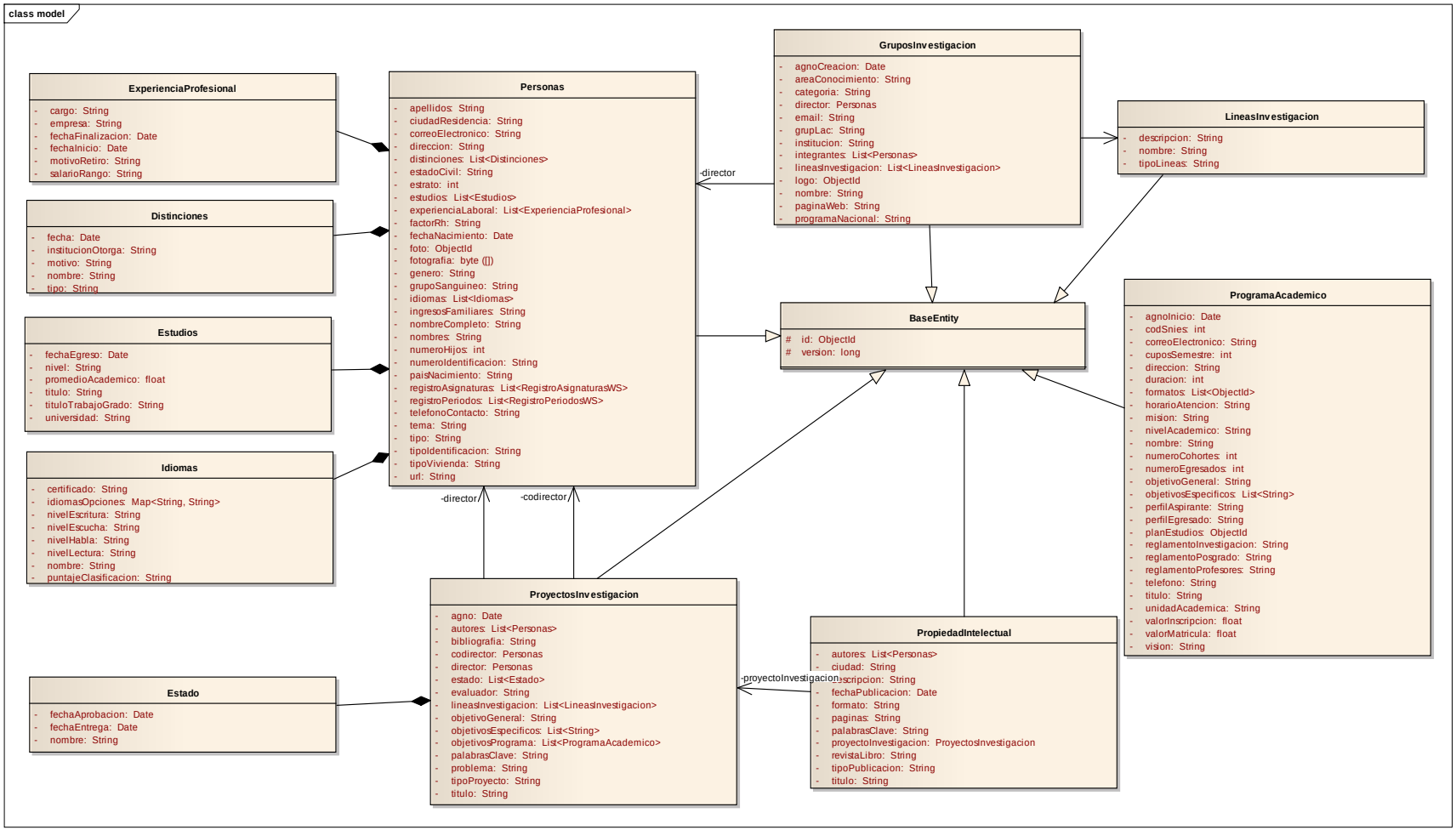
```java
@Entity(value = "personas")
public class Personas extends BaseEntity{

    private String tipo;
    private String nombres;
    private String apellidos;
    private String tipoIdentificacion;
    private String numeroIdentificacion;
    private Date fechaNacimiento;
    private String paisNacimiento;
    private String genero;
    private String estadoCivil;
    private int numeroHijos;
    private String ingresosFamiliares;
    private String tipoVivienda;
    private String direccion;
    private int estrato;
    private String ciudadResidencia;
    private String correoElectronico;
    private String telefonoContacto;
    private ObjectId foto;

    @Embedded
    private List<Estudios> estudios;

    @Embedded
    private List<Idiomas> idiomas;

    @Embedded
    private List<ExperienciaProfesional> experienciaLaboral;
```

Fuente: Autor

4.3.1. **Diagrama de clases**

Después de la elaboración del modelo conceptual, el siguiente paso realizado fue la creación del Diagrama de Clase de la Base de Datos de MongoDB la cual se muestra a continuación en la figura 7. De igual manera se aprecian las relaciones que tienen con la superclase *BaseEntity*, heredando sus atributos y métodos. Una de las clases principales denominada “Personas” está compuesta por clases embebidas como idiomas, estudios, distinciones y experiencia profesional.

Figura 7 Diagrama de Clases MongoDB



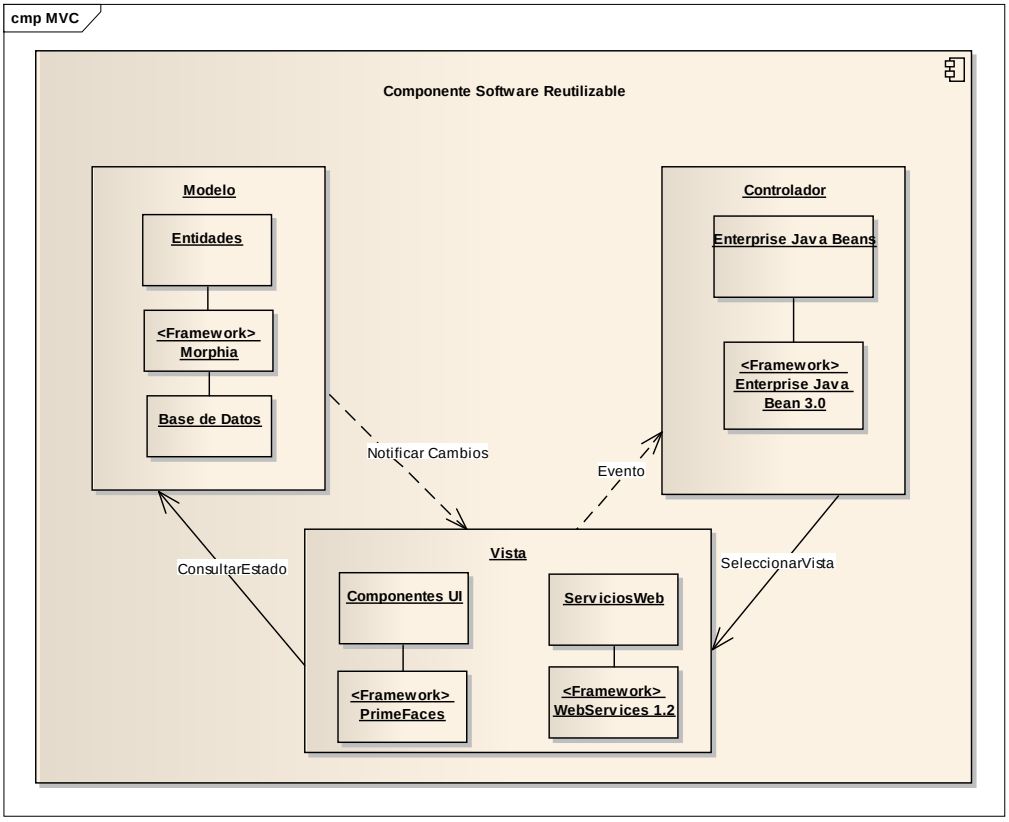
Fuente: Autor

5. DISEÑO E IMPLEMENTACIÓN DE COMPONENTE SOFTWARE

5.1.1. **DISEÑO Y DESCRIPCIÓN DE LA ARQUITECTURA MVC**

Franky[7] describe la arquitectura MVC de la siguiente manera: “El modelo, maneja las reglas del negocio y las estructuras de datos; la vista, maneja la presentación de los datos del sistema al usuario; y el controlador: Transforma pedidos del usuario en operaciones sobre los objetos del modelo y selecciona la vista para mostrar los resultados del usuario”. La adaptación de este estilo arquitectónico para los componentes software reutilizables se puede apreciar en la figura 8.

Figura 8 Componente Modelo Vista Controlador



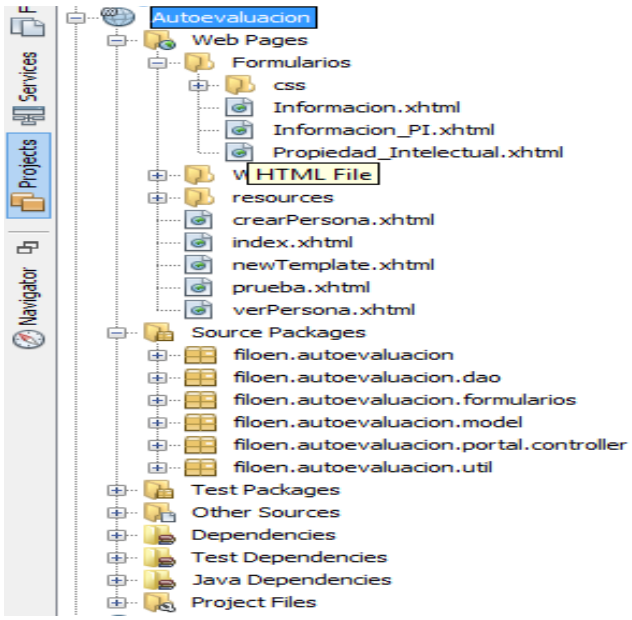
Fuente: Autor

[7]FRANKY, Maria Consulo. Curso: Desarrollo de aplicaciones en Java EE5. CincoSOFT LTDA 2007

En el Modelo estarían las entidades soportadas por el *framework* [8]de Morphia que se encarga de realizar llevar un control de las entidades, las cuales representan colecciones de la base de datos. En la Vista se ubican los componentes o controles para la interfaz de usuario, desarrollado por medio de PrimeFaces, en ella se presentan todos los archivos xhtml. El controlador utiliza los ManagedBeans soportados por Enterprise JavaBeans 3.0. Los controladores son los encargados de manejar la lógica del negocio del repositorio, se encarga de procesar la información suministrada por los clientes y presentar los resultados.

En cuanto a la implementación del patrón de diseño Modelo Vista Controlador en el proyecto general de Autoevaluación de postgrado, éste se dividió en diferentes paquetes que contienen por separado los elementos de los componentes en este caso el modelo la vista y el controlador;

Figura 9 Paquetes MVC

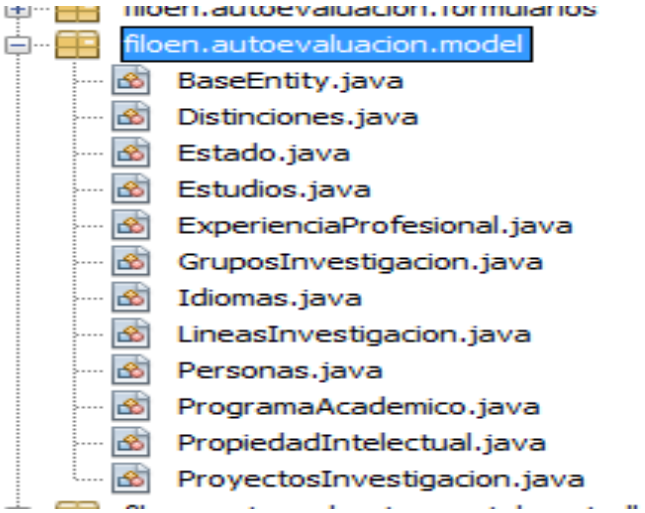


Fuente: Autor

[8] Un *Framework es* utilizado en la ingeniería del software para referirse a una estructura –comúnmente organizada en capas– que indica qué tipo de programas se puede o se debe construir y cómo se interrelacionan; además ofrece una plataforma reutilizable y universal para el desarrollo de aplicaciones, productos o soluciones informáticas.

De esta manera se separan cada una de las responsabilidades en los componentes desarrollados. La estructura de paquetes en donde se pueden localizar las entidades (por ejemplo el paquete filoen.autoevaluacion.model) presenta cómo se encuentran organizadas todas las clases que corresponden al modelo.

Figura 10 Paquete de las entidades que representan el Modelo de MVC

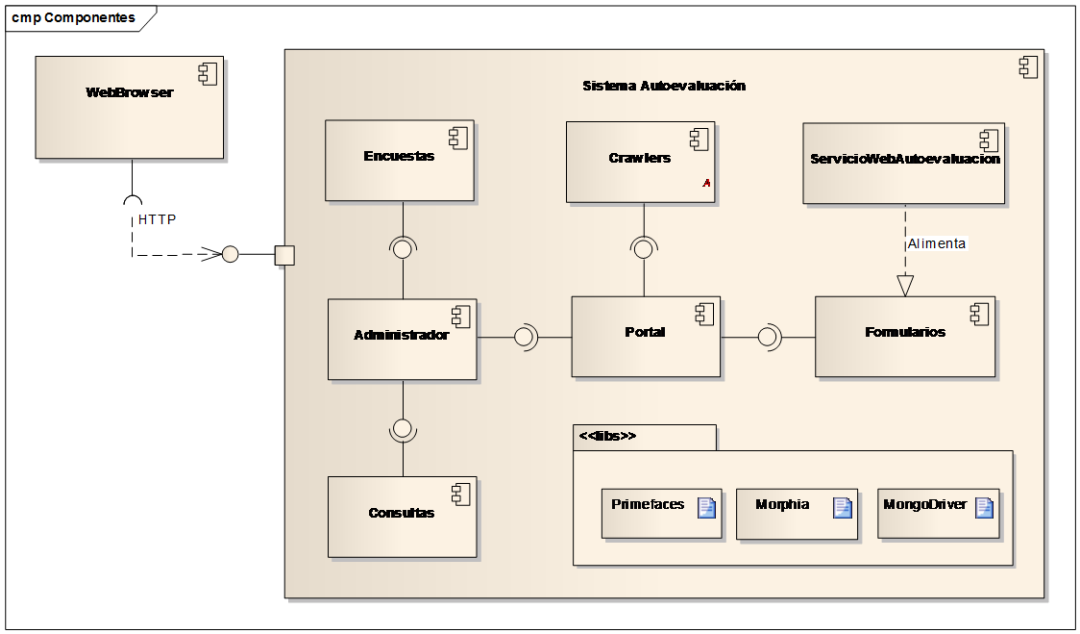


Fuente: Autor

5.1.2. **MODELAMIENTO DE COMPONENTES DEL SISTEMA**

El Diagrama de componentes de la figura 11 muestra cómo se relacionan los subcomponentes que se integraron para desarrollar el proyecto del sistema de Autoevaluación, tales como el portal, los formularios, las encuestas y las diferentes librerías que intervinieron como PrimeFaces, Morphia y MongoDriver. También se puede observar que este sistema está basado en una interfaz orientada a la web.

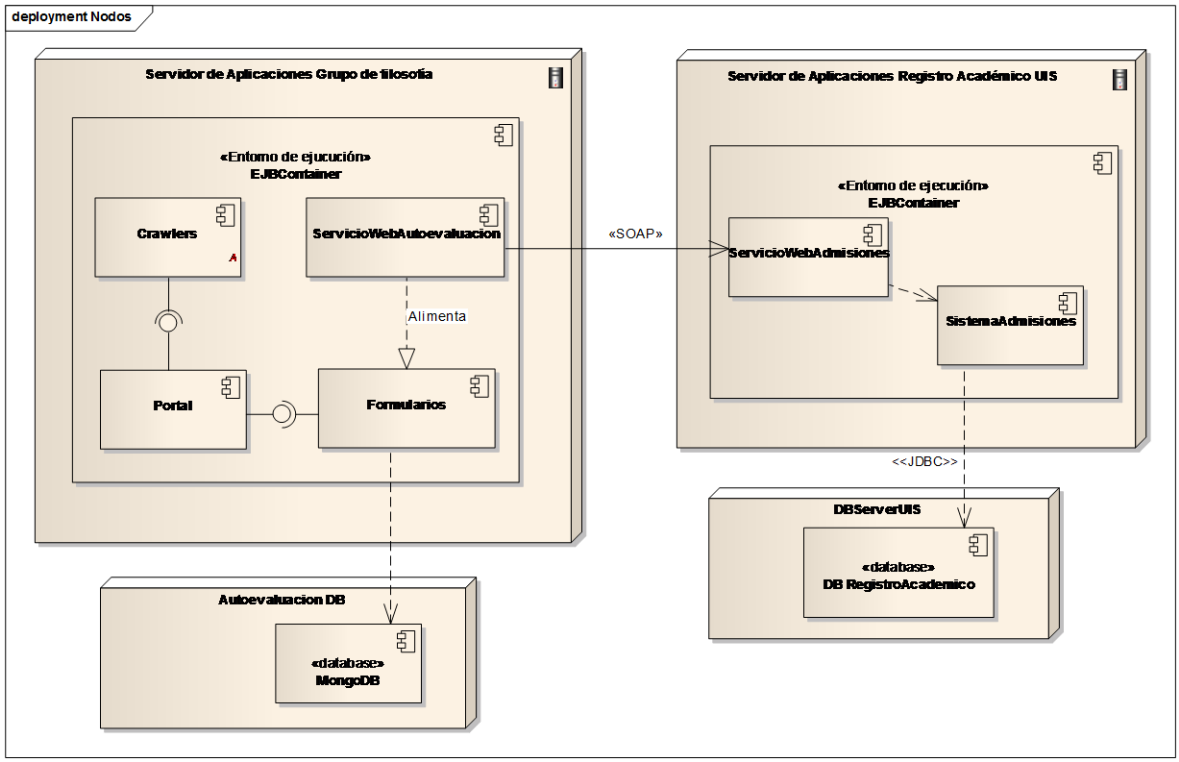
Figura 11 Diagrama de Componentes



Fuente: Autor

Después del diseño del diagrama de componentes, se realizó el diagrama de despliegue, en donde se describe el funcionamiento y las distintas interacciones entre los dos sistemas. En primer lugar el servicio web de Autoevaluación envía una petición al sistema de admisiones mediante el protocolo SOAP que se encuentra en el servidor de registros académicos de la UIS, éste envía la petición al servicio web de admisiones el cual responde al servicio web de Autoevaluación y alimenta los formularios y a su vez los componentes del sistema de auto evaluación (figura 12).

gFigura 12 Diagrama de Despliegue



Fuente: Autor

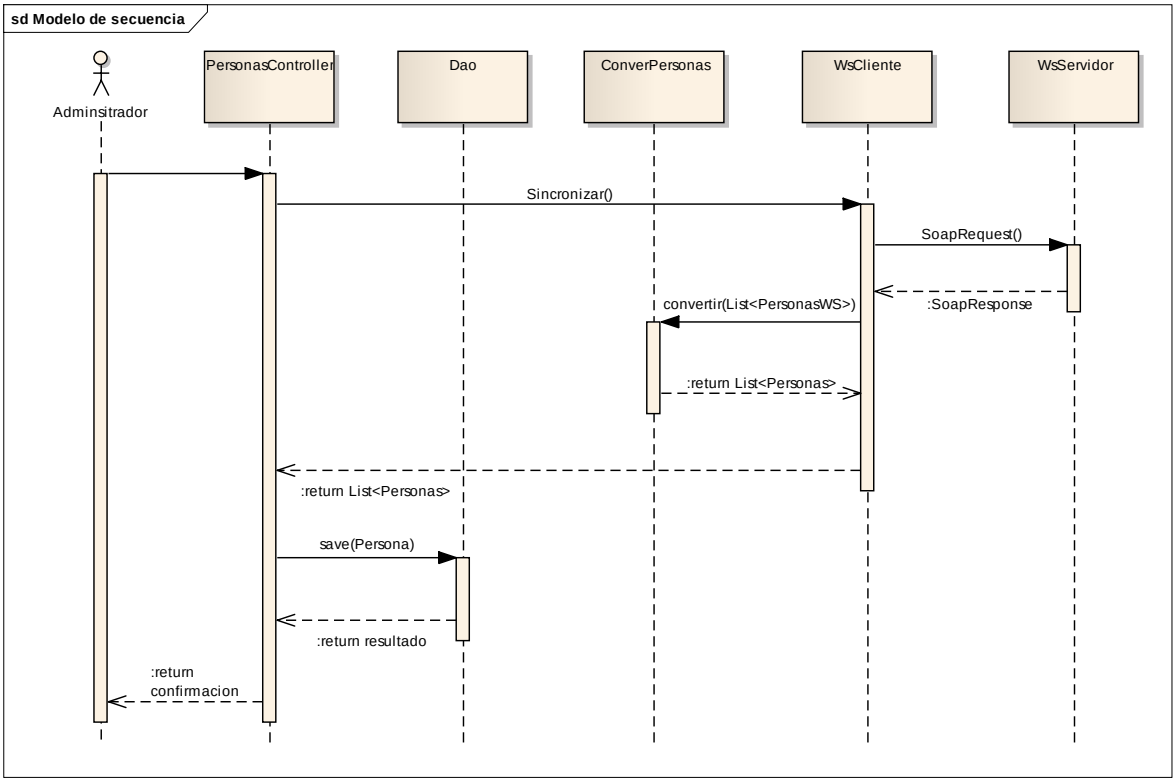
5.2. DESARROLLO DEL COMPONENTE DE COMUNICACIÓN CON EL SISTEMA DE REGISTRO ACADÉMICO

Para el buen y eficaz funcionamiento del componente software se realizó un servicio web para la interacción con la base de datos de registro académico. Para ello se creó un servicio web admisiones y un servicio web para el sistema de autoevaluación, donde el sistema de Autoevaluación es el cliente del servicio. Una vez determinados los actores y sus roles se especificó la información más relevante y los datos que se deben consultar del registro académico de cada estudiante. Se decidió extraer toda la información académica relacionada con los estudiantes activos, sus datos personales, así como toda la información relacionada con las asignaturas, planes de estudio, las materias electivas, notas, créditos, etc.

En la figura 13 se presenta de manera detallada las clases involucradas en el servicio web para la sincronización de los datos, el tipo de clases y métodos

involucrados, de modo similar se indica el tipo de resultado obtenido en el servicio web.

Figura 13 Diagrama de secuencia del ServicioWeb



Fuente: Autor

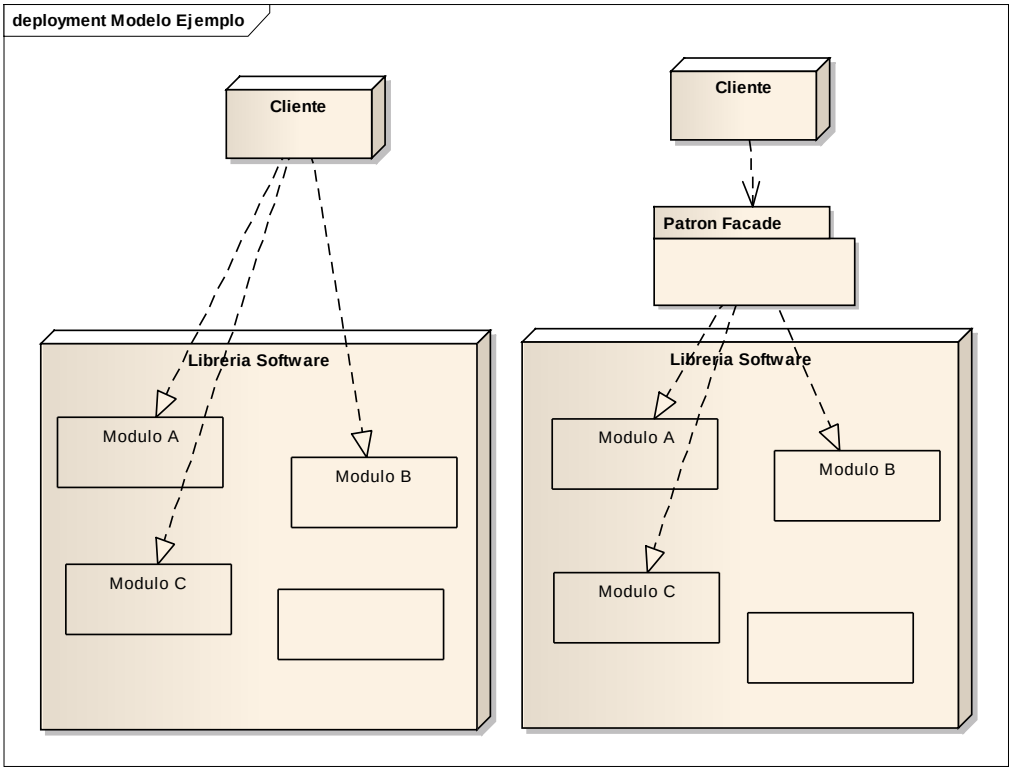
5.2.1. **Patrones de diseño utilizados en el desarrollo del componente**

5.2.1.1. **Patrón Facade**

El patrón de diseño Facade es un tipo de patrón estructural cuya finalidad está en la de reducir la complejidad del código fuente llevando a cabo una división en subsistemas, así logrando minimizar las comunicaciones y dependencias de ciertos componentes. Este patrón se aplica regularmente cuando se necesite proporcionar una interfaz simple para un subsistema complejo, o cuando se quiera estructurar varios subsistemas en capas, ya que las fachadas serían el punto de entrada a cada nivel.

A continuación se representa un diagrama con la estructura del patrón de diseño. En el ejemplo se utiliza el patrón a través de la clase Facade el cual crea un método único que comparten los modulo A, modulo B y Modulo C, éste resume o facilita todo el proceso, ya que el cliente sólo llama el método único.

Figura 14 Patrón Facade

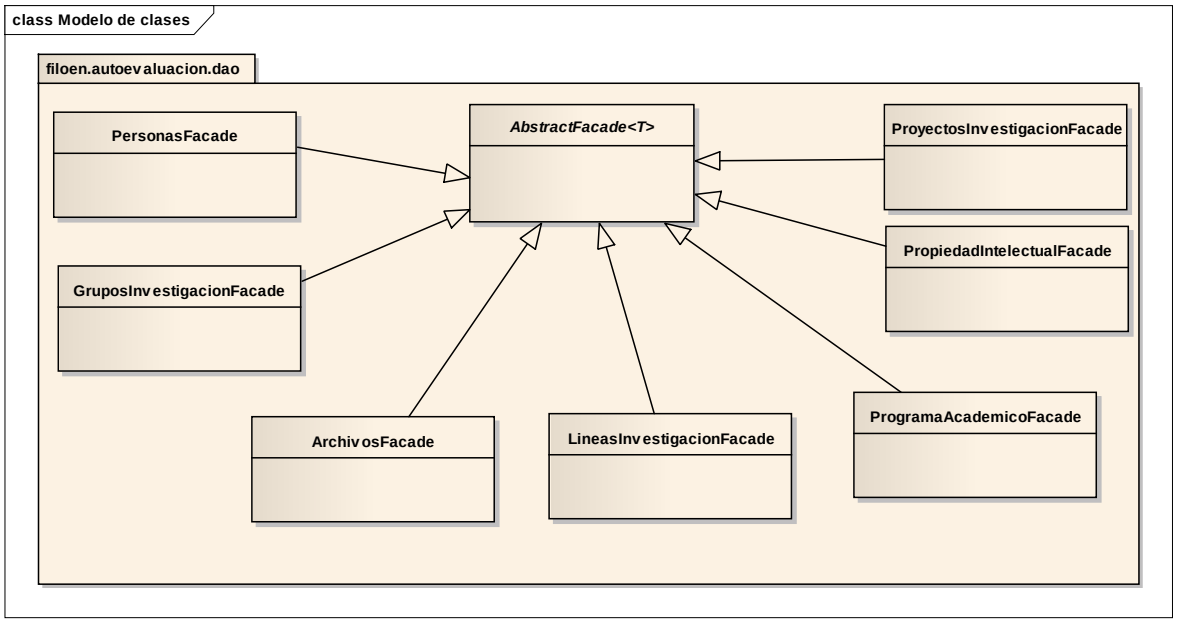


Fuente: Patrones de diseño. Facade pattern[9]

En la figura 15 se puede observar el diagrama de clases del patrón Facade utilizado en el desarrollo del proyecto. Este patrón se encuentra en el paquete filoen.autoevaluacion.dao en donde se aprecia la generalización hacia la clase AbstractFacade<T> por lo cual los subsistemas como PersonasFacade, ArchivosFacade, LineasdeInvestigacionFacade, estos heredan las propiedades de ésta clase, lo que hace que se reduzcan las comunicaciones y dependencias de ciertos componentes.

[9] Tomado de Patrones de diseño. Facade Pattern disponible en: http://www.thecoldsun.com/es/content/11-2008/patrones-de-diseno-facade-pattern

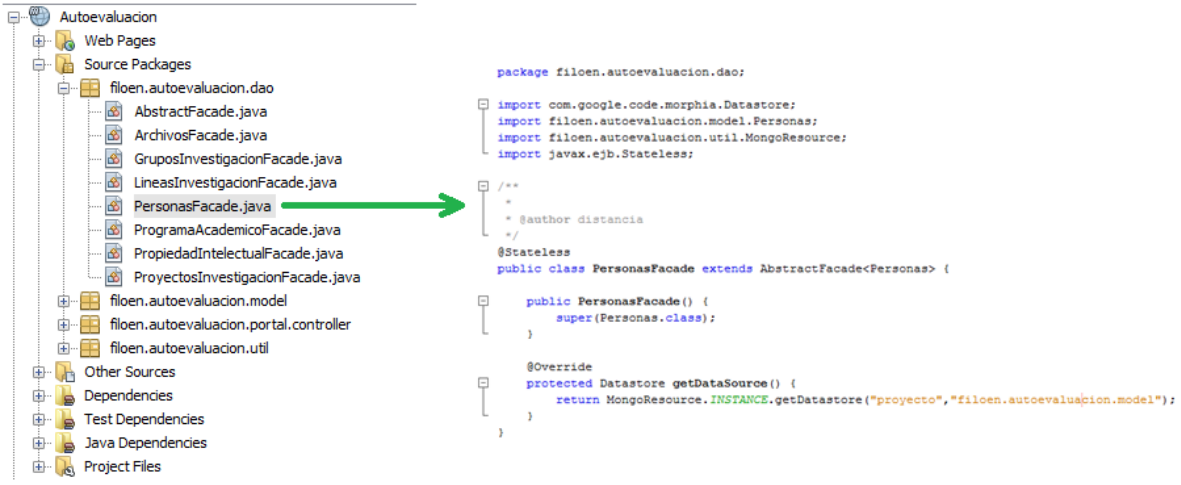
Figura 15 Patrón Facade aplicado al Componente Software



Fuente: Autor

A continuación se muestra la implementación de éste mismo patrón de diseño y su respectivo código aplicado a la Clase PersonasFacade donde se aprecia que PersonasFacade es una clase extendida de a clase AbstractFacade<T>

Figura 16 Patrón Facade aplicado a la Clase Personas



Fuente: Autor

5.2.1.2. **Patrón de diseño Front Controller**

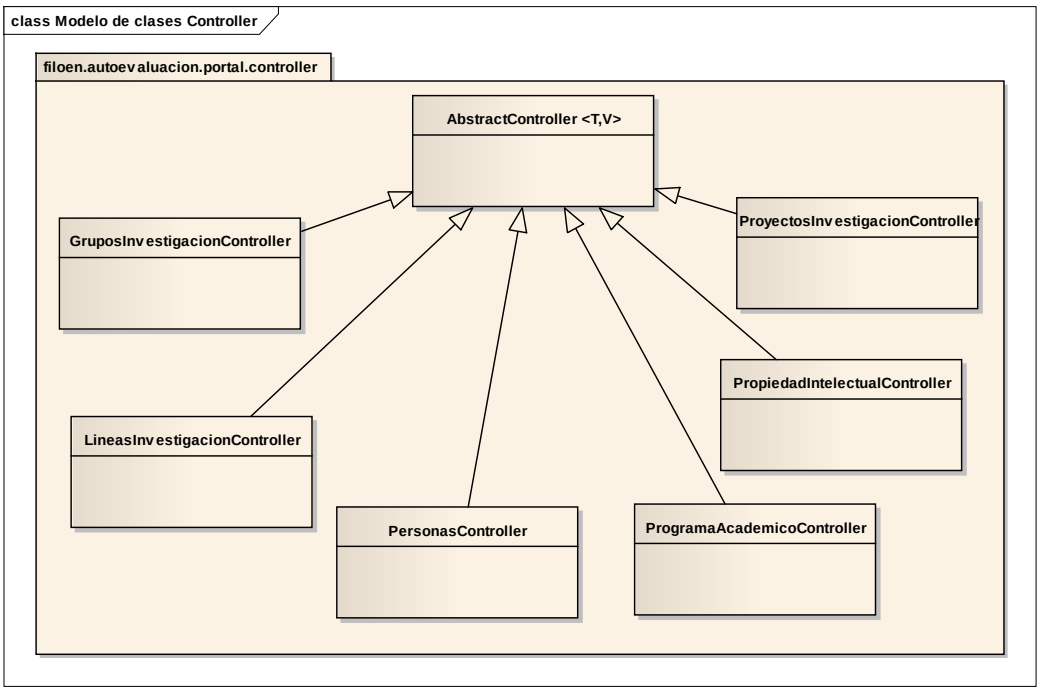
El patrón de diseño del controlador frontal (Front Controller) se utiliza para proporcionar un mecanismo centralizado de manejo de peticiones para que todas las solicitudes sean atendidas por un único controlador. Este controlador puede hacer la autenticación / autorización / registro o seguimiento de la solicitud y luego pasar a las solicitudes de los controladores correspondientes tales como GruposdeInvestigacionController, PersonasController, LineasInvestigacion Controller, ProgramasAcademicosController.

Las ventajas que proporciona este patrón de diseño son:

- ✓ Centralización en un único punto la gestión de las peticiones.
- ✓ Se ayuda la reusabilidad de código.
- ✓ Se mejora la gestión de la seguridad.

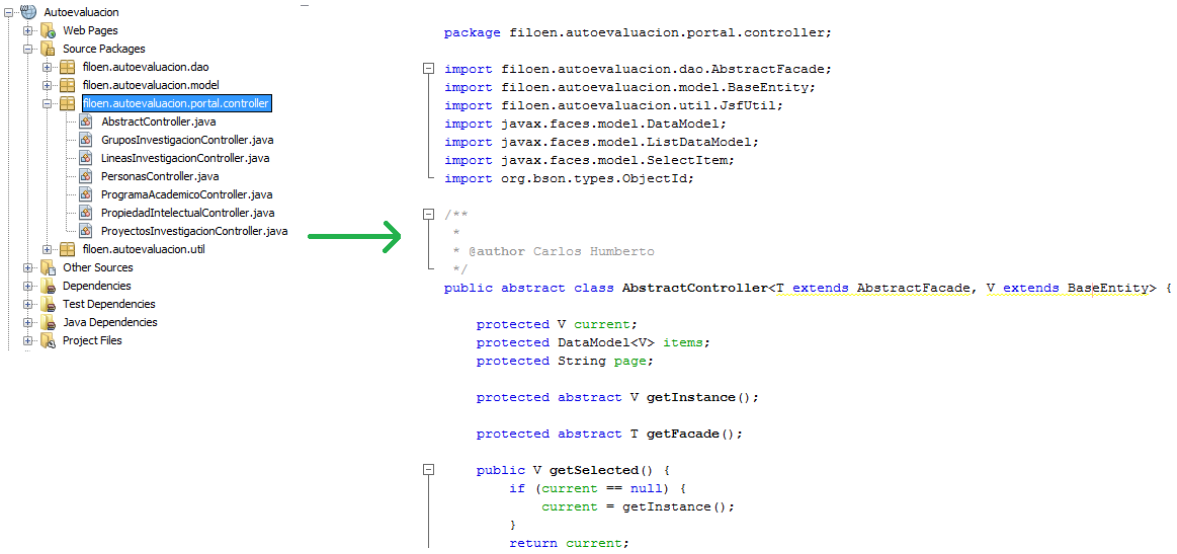
Se observa el diagrama de clases y el código implementado en el proyecto con el patrón Controller con sus respectivas generalizaciones y sus diferentes controladores

Figura 17 Patrón Controller aplicado al Componente Software



Fuente: Autor

Figura 18 Patrón de Diseño Front Controller



Fuente: Autor

5.2.1.3. **Patrón de diseño Singleton**

El Patrón Singleton es uno de los patrones de diseño más simples en Java. Este tipo de patrón de diseño viene bajo un patrón creacional, puesto que éste ofrece una mejor manera de crear un objeto. Este patrón consiste en una sola clase, que es responsable de crear un objeto propio, mientras se asegura de que el objeto único se haya creado. Esta clase proporciona una manera fácil de acceder a su único objeto, el cual se puede acceder directamente sin necesidad de crear una instancia del objeto de la clase.

El patrón Singleton provee una única instancia global gracias a que:

- ✓ La propia clase es responsable de crear la única instancia.
- ✓ Permite el acceso global a dicha instancia mediante un método de clase.
- ✓ Declara el constructor de clase como privado para que no sea instanciable directamente.

El patrón Singleton utilizó en la creación de la clase MongoResource, dado que se necesitaba un solo gestor de conexión y éste patrón permite garantizar una única conexión a la base de datos mongodb, ya que no se deja instanciar en ninguna otra clase debido a que el constructor de ésta clase es creada tipo private, esto

significa que no se puedan establecer conexiones paralelas a la base de datos, aumentando la seguridad y la gestión de recursos del sistema.

Figura 19 Aplicación del Patrón Singleton en MongoResource

```
public enum MongoResource {

    INSTANCE;
    private MongoClient mongoClient;
    private Morphia morphia;
    private ResourceBundle properties;

    private MongoResource() {

        try {
            if (properties == null) {
                properties = getBundle();
            }

            if (mongoClient == null) {
                mongoClient = getClient();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

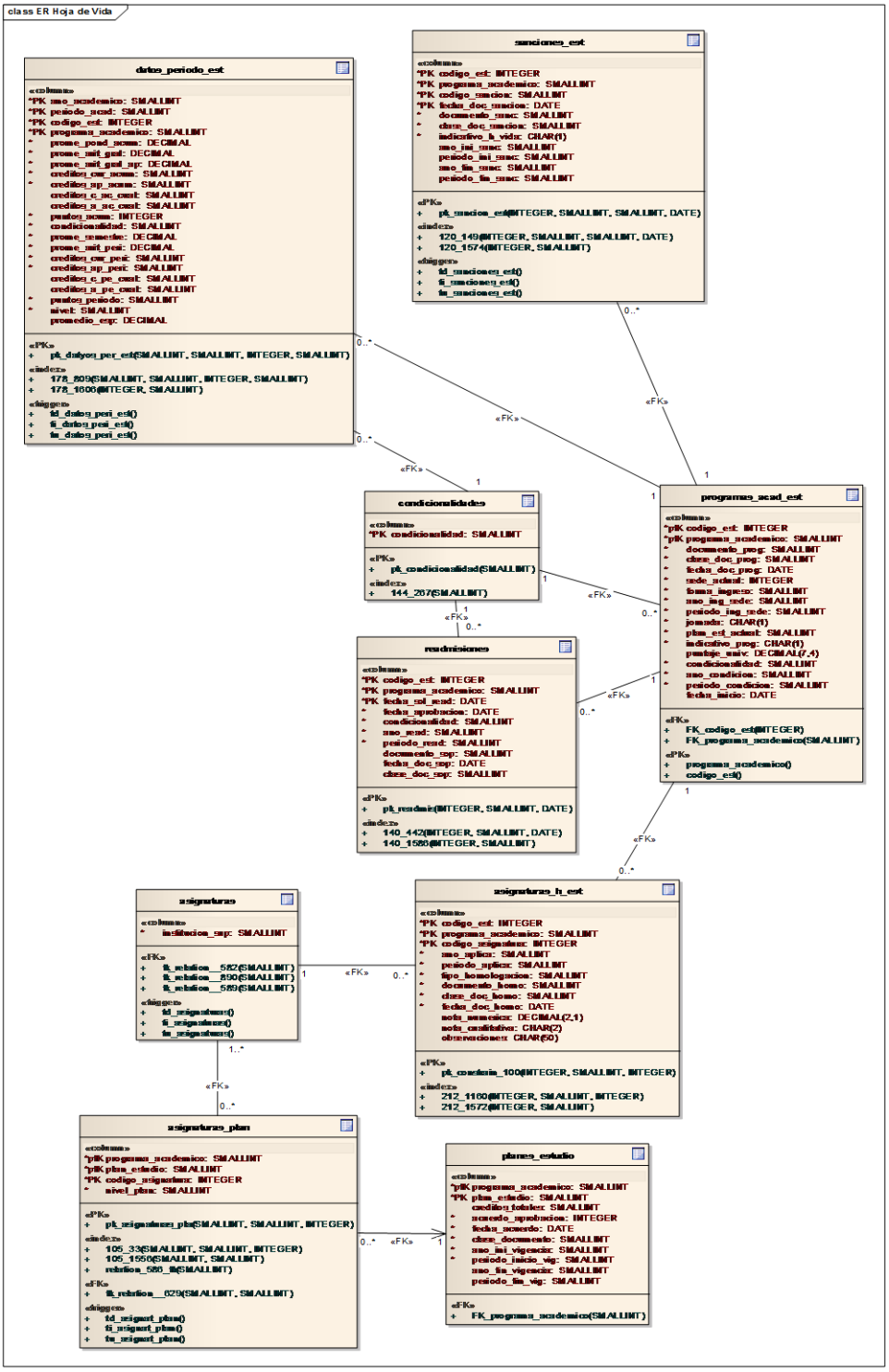
Fuente: Autor

5.2.2. Generación del servicio web en el servidor de registro académico

5.2.2.1. Análisis y diseño de la base de datos de sistema de Registro Académico

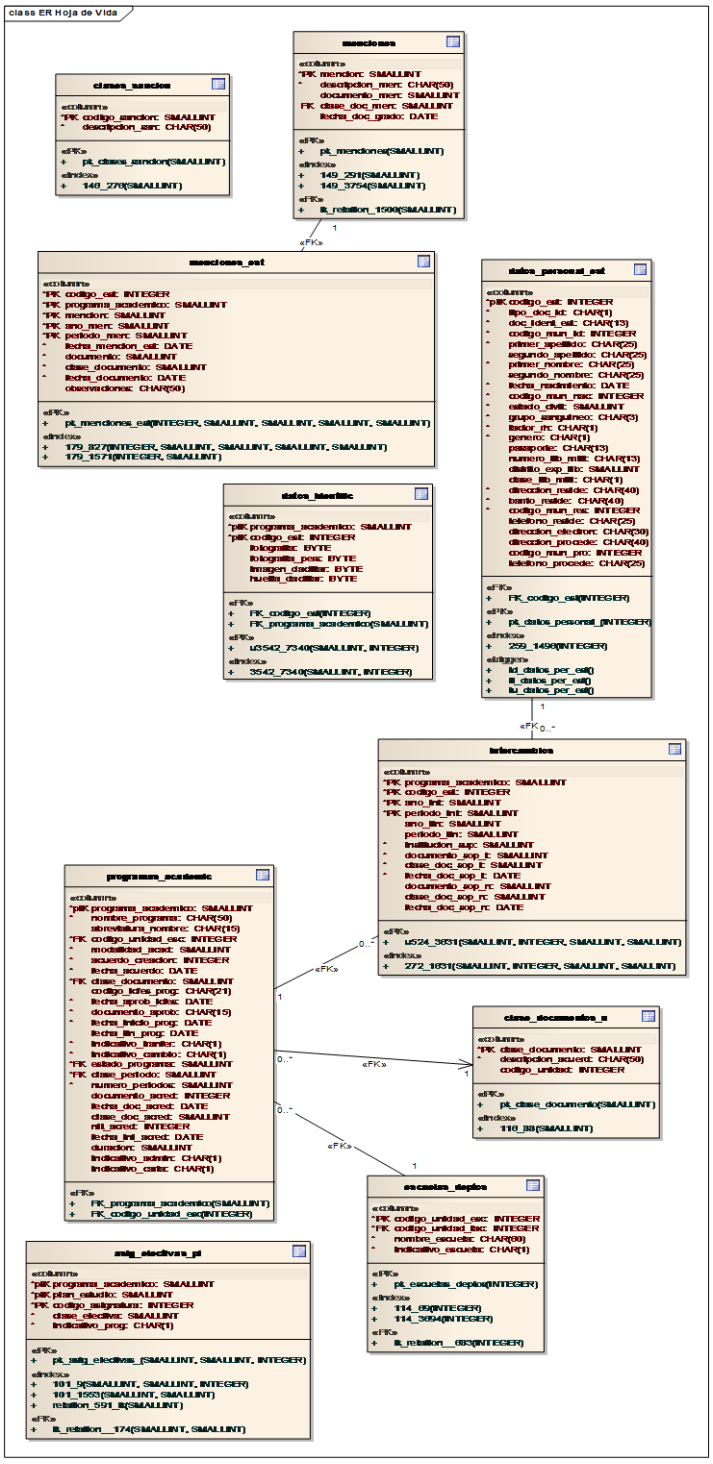
Para la creación del servicio web lo primero que se realizó fue una revisión de las tablas de la base de datos del sistema de Registro Académico de la universidad. A partir de ello se hizo una depuración de las tablas que serían utilizadas por el sistema de Autoevaluación. Asimismo se revisaron las relaciones -primarias y foráneas-, sus multiplicidades e índices para poder generar el modelo de clases a utilizar y así poder vincular adecuadamente la información registrada en la base de datos del sistema de Autoevaluación de postgrados de la escuela de ingeniería de sistemas (figuras 20,21).

Figura 20 Base de datos Registro Académico



Fuente: Autor

Figura 21 Base de datos Registro Académico Continuación

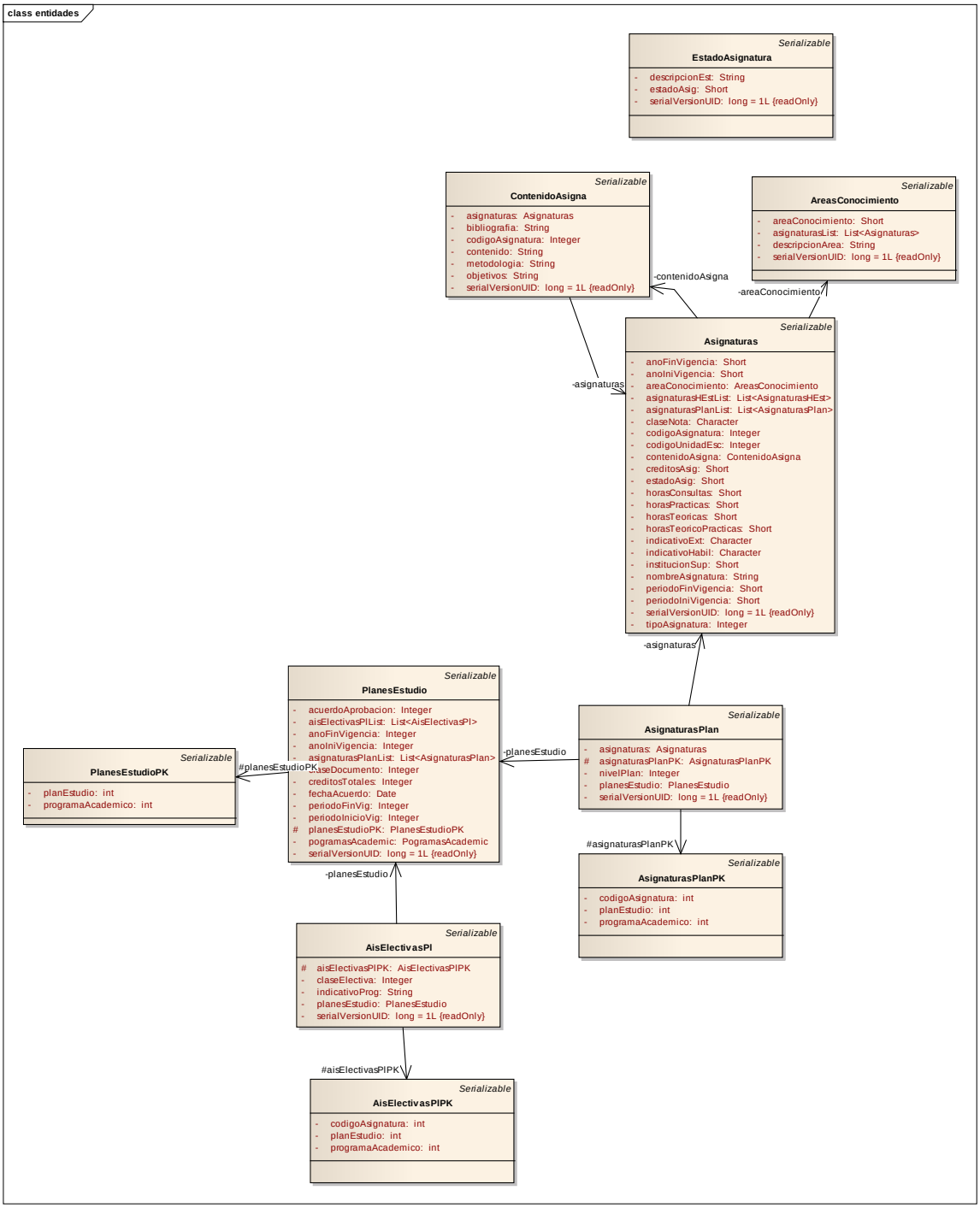


Fuente: Autor

5.2.2.2. **Creación de entidades para la gestión de tablas del Sistema de registro Académico**

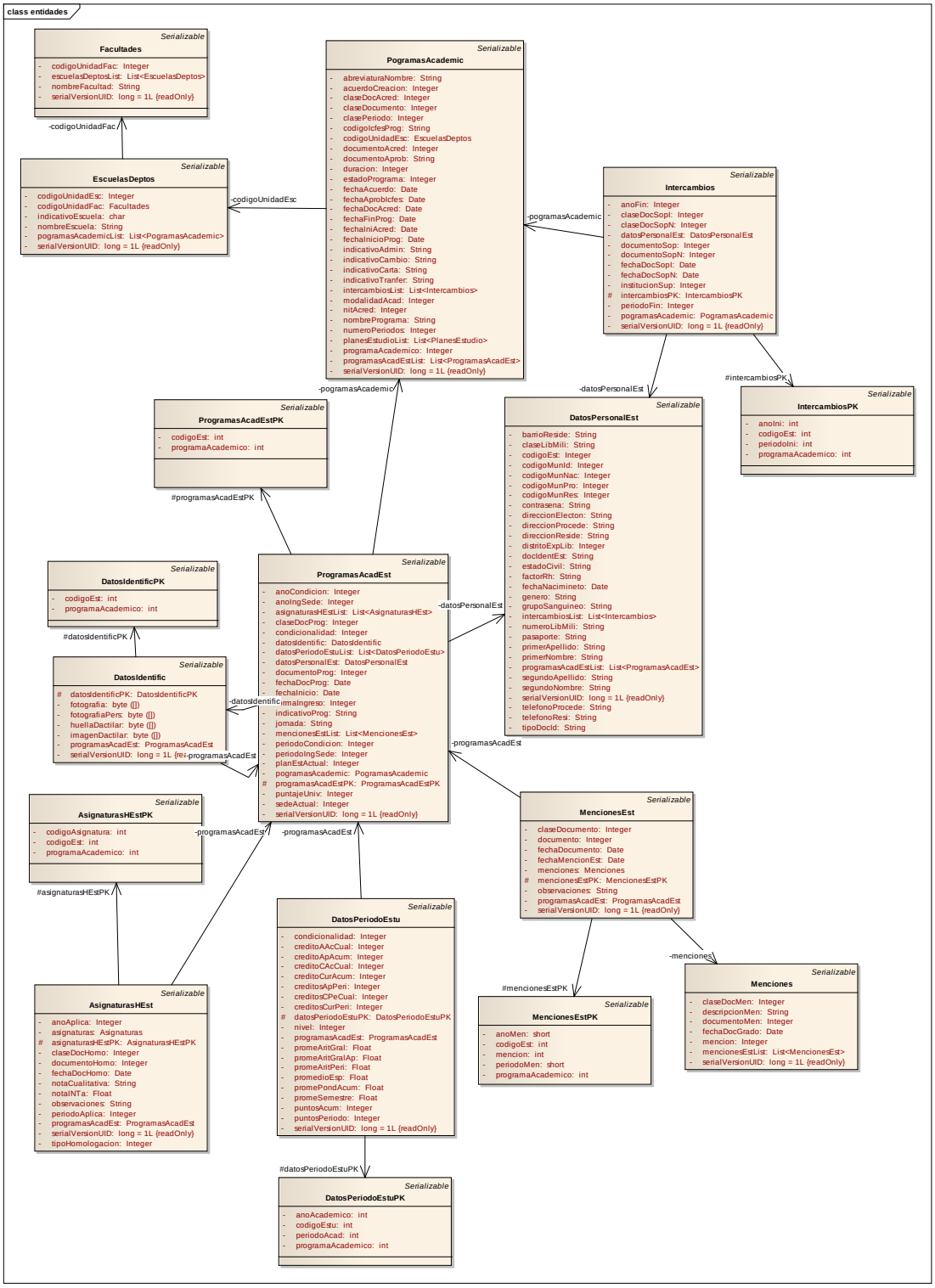
Una vez realizado el proceso de filtrado de tablas el siguiente paso fue modelar el diagrama de clase con sus llaves primarias, foranes y todas las relaciones que ellas tienen como se muestra en las figura 22,23.

Figura 22 Diagrama de Clase Entidades



Fuente: Autor

Figura 23 Diagrama de Clase Entidades Continuación



Fuente: Autor

Por medio de Java Persistence API (JPA) se proporciona un modelo de persistencia basado en POJO's para buscar bases de datos relacionales en Java. Un ejemplo de una entidad creada a partir del diagrama de clases, sería la entidad DatosPersonalEst, esta clase se creó con todos sus atributos y características correspondientes. Las entidades utilizan las anotaciones del API de Persistencia entre las que se encuentran:

@Entity: Una entidad es un objeto de dominio persistencia ligero. Típicamente una entidad representa una tabla de una base de datos relacional, y cada instancia de la entidad corresponde a una fila de esa tabla. El artefacto de programación principal de la entidad es la clase entity, si bien las entidades pueden utilizar las clases de ayuda, el estado persistente de una entidad se representa bien a través de campos persistentes o de propiedades persistentes. Estos campos o propiedades utilizan objetos / anotaciones de mapeo relacional para mapear las entidades y relaciones entre entidades o los datos relacionales en el almacén de datos subyacente.

@table: Este tipo de anotación es básica en la estructura de persistencia de una base de datos relacional. Una tabla contiene una lista de columnas que definen la estructura de la tabla, y una lista de filas que definen los datos de la tabla. Cada columna tiene un tipo específico y tamaño generalizado. El conjunto estándar de tipos de relaciones se limitan a los tipos básicos incluyendo numérico, caractér, fecha y hora, y el binario (aunque la mayoría de las bases de datos modernas tienen otros tipos). Las tablas también pueden tener restricciones que definen las reglas que restringen los datos de fila, como clave primaria, clave externa, y restricciones únicas. Las tablas también tienen otros artefactos, como índices, particiones y los factores desencadenantes.

@Id: Cada objeto de una entidad que se almacena en la base de datos tiene una clave principal. Una vez asignada, la clave principal no puede ser modificada. Se representa el objeto de entidad siempre y cuando existe en la base de datos.

@column: Se utiliza para especificar una columna asignada, es una propiedad de persistencia. Si no se especifica ninguna anotación @column, se aplican los valores predeterminados.

@XmlRootElement: Esta anotación se puede utilizar con los siguientes elementos de programa: una clase de nivel superior o un tipo enum. Cuando una clase de nivel superior o un tipo de enumeración está anotado con esta anotación, entonces su valor se representa como elemento XML en un documento XML.

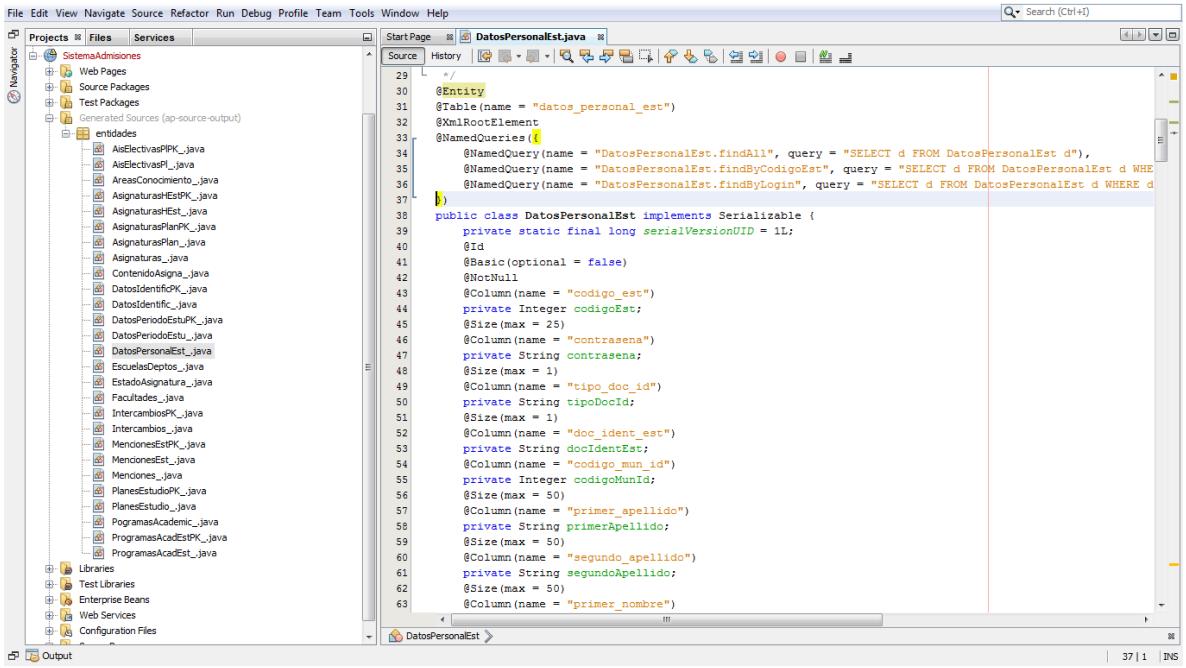
@NamedQueries: Es una consulta definida estáticamente con una cadena de consulta inmutable predefinido. El uso de consultas con nombre en lugar de

consultas dinámicas puede mejorar la organización del código mediante la separación de las cadenas de consulta JPQL desde el código Java. También impone el uso de parámetros de consulta en lugar de incrustar los literales de forma dinámica en la cadena de consulta y los resultados de las consultas son más eficientes.

Cuando una entidad tiene múltiples campos de clave principal, JPA requiere la definición de una clase especial de identificación que se adjunta a la clase de entidad mediante la anotación @ IdClass. El ID de clase refleja los campos de clave principal y sus objetos pueden representar valores de clave principal:

En la siguiente figura se puede observar cómo es la creación de esta entidad con sus respectivos métodos.

Figura 24 Entidad DatosPersonalEst



Fuente: Autor

5.2.3. **Comunicación de los sistemas a integrar mediante servicios web**

Una vez generadas las clases utilizando el API de persistencia se procedió a generar un conjunto de servicios web para la autenticación, búsqueda y sincronización de los registros de usuario con el sistema de Autoevaluación. Para la comunicación se trabajó mediante SOAP, y se crearon clases que permitieran la conversión de las clases generadas que manipulan los datos de la base de datos de Registro académico en una clase compatible con el sistema de Autoevaluación (mediante la clase PersonasConverter). De esta manera la comunicación entre los dos sistemas se realizará de una forma transparente, evitando incompatibilidad entre los modelos de datos relacional y no relacional.

Figura 25 Servicios web en el sistema de Registro Académico

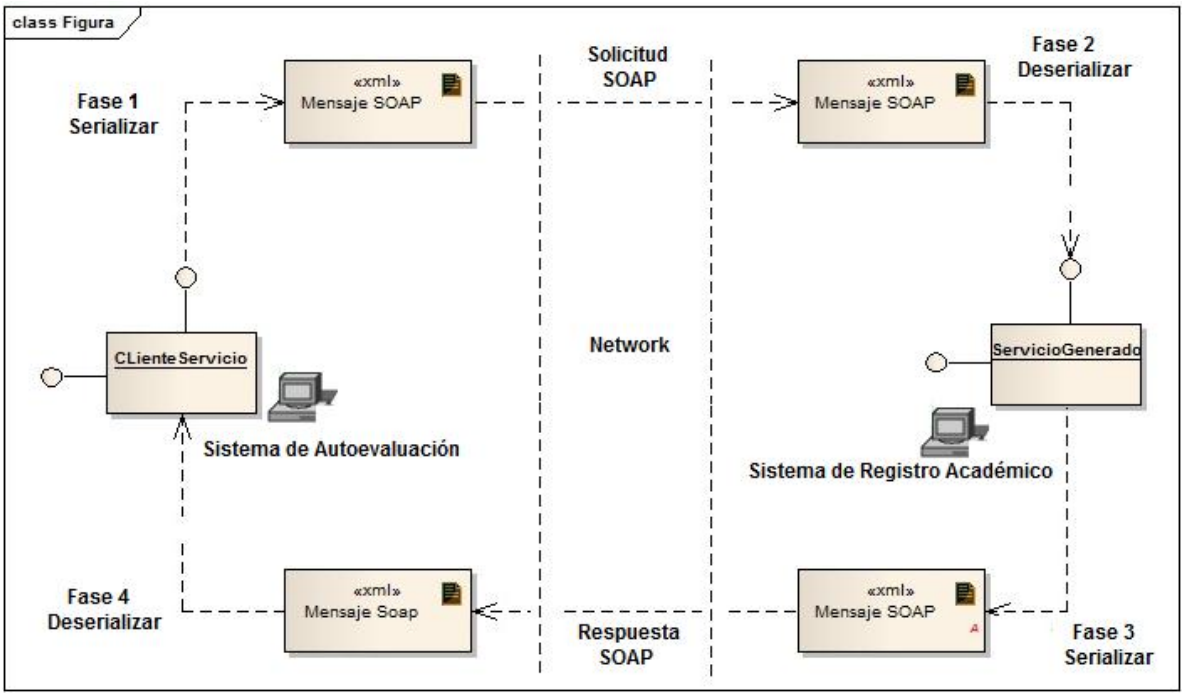
```
@WebMethod(operationName = "loginUser")
public PersonasWS loginUser(@WebParam(name = "cod") int cod, String pass) {
    return PersonasConverter.convertirDatosEstudiante(ejbDatosPer.findByLogin(cod, pass));
}

@WebMethod(operationName = "findByStudentID")
public PersonasWS findByStudentID(@WebParam(name = "cod") int cod) {
    return PersonasConverter.convertirDatosEstudiante(ejbDatosPer.findByStudentId(cod));
}

@WebMethod(operationName = "findByProgramId")
public List<PersonasWS> findStudensByProgramId(@WebParam(name = "cod") int codPrograma) {

    List<PersonasWS> listadoEstudiantes;
    List<ProgramasAcadEst> listadoEstudiantesPrograma;
    PersonasWS temp;
    listadoEstudiantesPrograma = ejbPrograma.findStudensByProgramId(codPrograma);

    if (listadoEstudiantesPrograma != null) {

        listadoEstudiantes = new ArrayList<>();
        for (ProgramasAcadEst progEst : listadoEstudiantesPrograma) {
            temp = PersonasConverter.convertirDatosEstudiante(progEst.getDatosPersonalEst());
            listadoEstudiantes.add(temp);
        }
        return listadoEstudiantes;

    } else {
        return null;
```

Fuente: Autor

En el sistema de Autoevaluación se procedió a generar un controlador encargado del llamado de los métodos del servicio. El ManagedBean llamado ServicioRegistroAcademico se encarga del proceso de consulta, validación, conversión y registro de la información extraída del sistema de Registro

académico. La figura 26 presenta el modelo de comunicación utilizado en la comunicación entre los dos sistemas.

Figura 26 Modelo de comunicación SOAP entre los sistemas de Autoevaluación y Registro Académico



Fuente: Autor

El cliente del servicio se encarga del llamado remoto de la función *findByProgramId*, deserealiza el mensaje XML recibido en el servicio web, lo transforma en un objeto de la clase Personas de *Morphia framework* y posteriormente lo actualiza en la base de datos del Sistema de Autoevaluación con la información académica de los estudiantes para su posterior análisis en los componentes software desarrollados de manera paralela en otros componentes del proyecto.

Figura 27 Llamado del servicio web utilizando el cliente del servicio

```
public void buscarEstudiantesPorPrograma() {

    List<PersonasWS> listaEstudiantesWS = new ArrayList<>();
    List<Personas> listaEstudiantes = new ArrayList<>();

    try {
        AdmisionesWebService_Service servicio = new AdmisionesWebService_Service();
        listaEstudiantesWS = servicio.getAdmisionesWebServicePort().findByProgramId(302);

        for (PersonasWS per : listaEstudiantesWS) {
            listaEstudiantes.add(PersonasConverter.convertirDatosEstudiante(per));
        }

        for (Personas per : listaEstudiantes) {
            getFacade().create(per);
        }
```

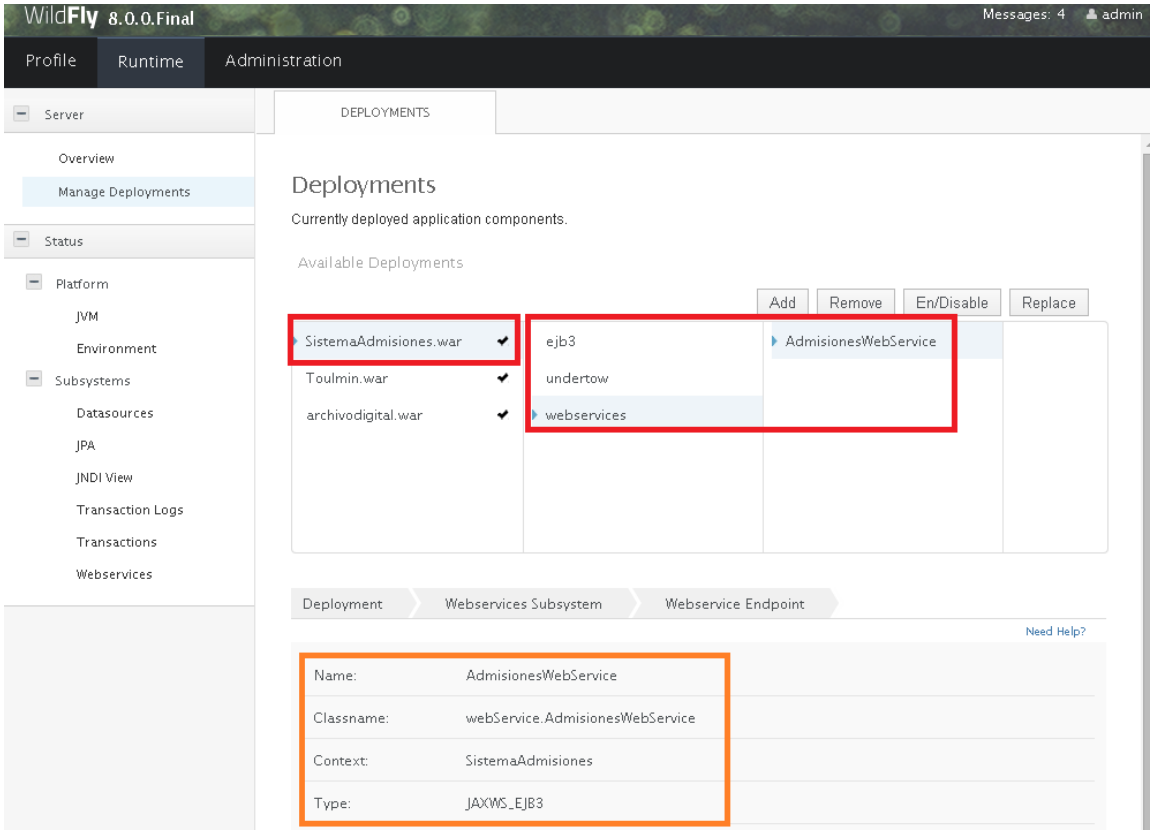
Fuente: Autor

6. INTEGRACIÓN Y PRUEBAS

6.1. IMPLANTACIÓN DE COMPONENTES

Una de las tareas más importantes desarrolladas por el equipo de trabajo fue la integración de los componentes del sistema de Autoevaluación para que se acoplaran adecuadamente, controlando su carga en el servidor web del grupo de investigación (http://filosofiayensenanza.uis.edu.co) el cual se ejecuta utilizando el servidor de aplicaciones WildFly 8.0.0 que se instaló y configuró durante el desarrollo del proyecto.

Figura 28 Servicio web desplegado en el servidor de aplicaciones



Fuente: Autor

Una vez se cargó el servicio y se probó la conectividad con la base de datos del sistema de Registro académico, se procedió a probar que el servicio web pudiera desplegarse y consumirse desde el sistema de Autoevaluación utilizando la ruta del WSDL designada para tal fin, como se observa en la figura 29.

Figura 29 Prueba de despliegue WSDL del Servicio

Fuente: Autor

Después de implementado el servicio web, se realizaron estos dos tipos de pruebas: de funcionalidad y de usabilidad, con el fin de encontrar fallas o errores en el servicio.

6.2. PRUEBAS DE FUNCIONALIDAD

Para las pruebas de funcionalidad se utilizó el método de caja negra, el cual consiste en realizar casos de prueba que demuestren que cada función del sistema es operativa. El método recibe su nombre debido a que se evalúan las respuestas de una entrada en una funcionalidad específica del sistema sin conocer cómo funciona el código internamente.

Para la realización de las pruebas se hicieron las siguientes actividades:

- Revisar las funcionalidades del sistema.
- Identificar las funcionalidades más relevantes del sistema.
- Diseñar los casos de prueba según las funcionalidades identificadas.
- Realizar los casos de prueba empleando datos de entrada.
- Revisar si las salidas que cumplen con la funcionalidad esperada.

De las funcionalidades del servicio se escogieron las más relevantes para el sistema que son:

- Autenticación.
- Sincronizar Registros

Los casos de prueba se muestran en tablas donde se especifica el nombre del caso, la descripción del procedimiento que se realiza, los valores de entrada, las respuesta a estos valores y el resultado correspondiente. Cuando se obtiene un resultado esperado, el caso es marcado con O.K. Cuando se obtiene un resultado no esperado, el caso es marcado con N.E (No esperado).

Tabla 15 Caso prueba Autenticación

Caso Prueba	Descripción	Valores entrada	Respuesta	Resultado
Autenticación Correcta	Se ingresan valores correctos de autenticación de usuarios que se encuentren registrados.	Usuario: admin Pass: admin	Inicio de sesión valido. Bienvenido, Carlos Humberto Carreño Díaz Cerrar Sesión Editar Perfil Administrar	O.K
Autenticación Correcta	Se ingresan valores correctos de autenticación de usuarios que se encuentren registrados.	Usuario: sonia Pass: 1234	Inicio de sesión valido Bienvenido, Sonia Gamboa Cerrar Sesión Editar Perfil Administrar	O.K
	Se ingresan		Inicio de sesión valido Bienvenido, Cristhian	

Autenticación Correcta	valores correctos de autenticación de usuarios que se encuentren registrados.	Usuario: cris Pass: 12345	Ruiz Cerrar Sesión Editar Perfil	O.K
Autenticación Incorrecta	Se ingresan valores con más de cincuenta caracteres.	Usuario:Qqqqq Pass: (%&$”()	size must be between 0 and 30	O.K
Autenticación Incorrecta	Se ingresan valores no válidos.	Usuario: (nulo) Pass: (nulo)	Debe digitar su contraseña. Debe digitar su Usuario	O.K
Autenticación Incorrecta	Se ingresan valores no válidos.	Usuario: Adton Pass: qwertyu	Usuario o Contraseña incorrectos. Verifique su usuario y contraseña	O.K

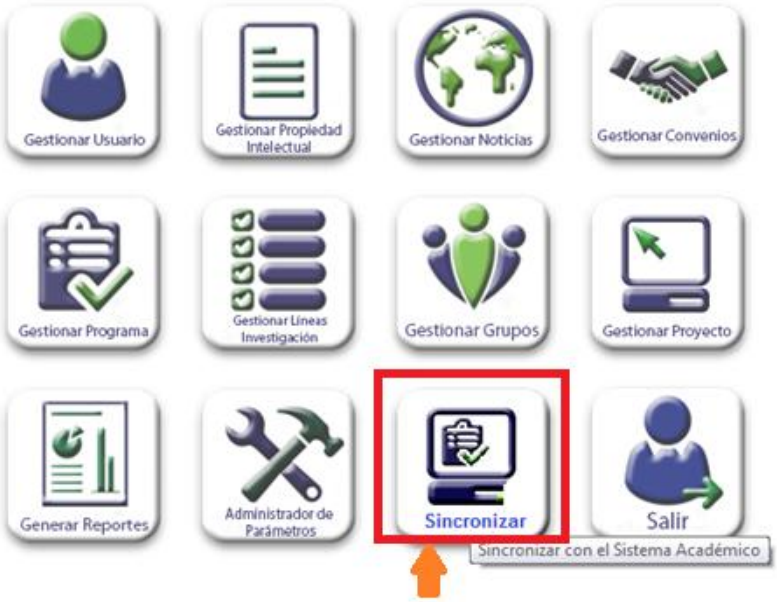
Fuente: Autor

De los casos de prueba anteriores para la autenticación puede concluirse que en su mayoría los datos entrada aplicados a las funcionalidades escogidas cumplen con las salida esperadas. En cuanto a la comunicación de los sistemas, se procedió a realizar prueba con registros de diferentes usuarios, cada uno de ellos con diferentes características, verificando que se cumplieran los siguientes casos:

- Estudiante de primer semestre
- Estudiante de primer semestre sin asignaturas matriculadas ni registro de periodos académicos
- Estudiante de último semestre
- Estudiante egresado, retirado o sin matrícula vigente
- Estudiante con información personal incompleta

En la figura 31 muestra la opción para la interacción con el sistema de registro académico de la Universidad Industrial de Santander. Este proceso deberá ejecutarse al iniciar cada periodo académico, al iniciar un nuevo proceso de autoevaluación o de acuerdo a especificaciones futuras del administrador del sistema.

Figura 30 Menú administrador



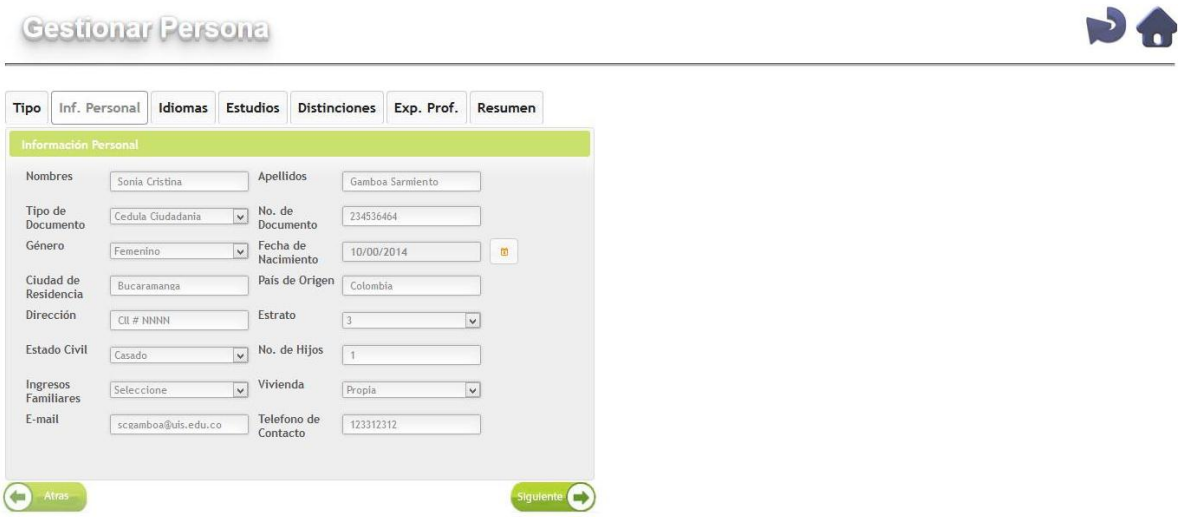
Fuente: Autor

Luego de la ejecución del proceso de consulta, se procedió a verificar en el sistema que los registros fueran registrados correctamente como se muestra en las figuras 32 y 33. Al seleccionar un registro se observa que los datos consultados en el sistema de autoevaluación ha sido actualizados directamente con los datos registrados en el sistema de autoevaluación, así como la información académica.

Figura 31 Resultados de la ejecución de la sincronización de datos en el sistema de autoevaluación

Fuente: Autor

Figura 32 Resultados de la ejecución de la sincronización de datos en el sistema de autoevaluación



Fuente: Autor

7. CONCLUSIONES

- La implementación de la interfaz del sistema de autoevaluación de la maestría de ingeniería de sistemas de la UIS contribuye a la cualificación del programa de postgrado de ingeniería de sistemas como un requisito para la acreditación y por ende enriquece a la universidad.

- El uso de las herramientas y patrones seleccionados agilizaron el proceso de desarrollo del servicio web, mostrando además su flexibilidad para ser usadas en proyectos de gran envergadura y con un numeroso grupo de desarrolladores.

- Con el desarrollo de este servicio web, se obtendrá un cambio notorio en el modo de captación de datos almacenados en registro académico, facilitando y simplificando la sincronización de los dos sistemas.

- La implementación de interfaces intuitivas en el portal, permiten a los usuarios un fácil entendimiento de las funcionalidades del portal.

- La sistematización y descentralización de la captura de los datos optimiza el proceso del manejo de la información.

8. RECOMENDACIONES

- Definir las políticas que permitan controlar los cambios realizados en los componentes y que afecten el funcionamiento del sistema que actualmente los implementa, esta política debería también adaptarse a los cambios en las estructuras de datos y entidades de uso general.

- Seguir fomentando y fortaleciendo el proceso de formación en programación en diversos lenguajes, aprovechando las nuevas tecnologías que van surgiendo cada día.

- Recalcar en las demás escuelas o departamentos, la implementación de sistemas que les permitan facilitar los procesos de autoevaluación que están realizando manualmente.

BIBLIOGRAFIA

CHANG, F. et al. \Bigtable: A Distributed Storage System for Structured Data". In: ACM Transactions on Computer Systems (2008), p. 1-26

IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE std. 830, 1998

Introducción a los Enterprise java vean. Disponible en: http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/ejb/sesion01-apuntes.htm

Lidia Fuentes, José M. Troya y Antonio Vallecillo Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga. ETSI Informática. Campus Teatinos, s/n. 29071 Málaga, Spain. Disponible en: http://www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf

NÄSHOLM. Petter, Extracting Data from NoSQL Databases. A Step towards Interactive Visual Analysis of NoSQL Data Febrero 2012. [En linea] <http://publications.lib.chalmers.se/records/fulltext/155048.pdf >[Citado en 20 de abril de 2013]

Palermo Rolando, 2012. Introducción a los EJB. Science and Engineering blog. Peru. Tomado de http://blog.rolandopalermo.com/2012/03/ejb-30-netbeans-java.html

Patrones de diseño. Facade Pattern disponible en: http://www.thecoldsun.com/es/content/11-2008/patrones-de-diseno-facade-pattern

Pavón Maestras, J. (2013). Java EE – Enterprise Beans (EJB). Madrid: Universidad Complutense Madrid.

Pérez, J. E. (2009). Introducción a JavaScript (p. 140). Disponible en http://books.google.com/books?hl=en&lr=&id=jCXw02NaYvEC&oi=fnd&pg=PP5&dq=JavaScript&ots=U0pc3-KpQW&sig=auHce3I8lNjFvJ4AAqdVXlJpNAo

Prada Carlos David y Ruiz Cristhian Fernando Diseño E Implementación De Un Portal De Conocimiento Orientado A Grupos De Investigación – Webscience. Trabajo de Grado Ingeniería de Sistemas. Bucaramanga. Universidad industrial de Santander Facultad de ingenierías físico mecánicas Escuela de ingeniería de sistemas 2013. 84p

Programa de Maestría EISI disponible en: http://cormoran.uis.edu.co/eisi/eisi.jsp?IdServicio=S78

SQL para MongoDb disponible en: http://docs.mongodb.org/manual/reference/sql-comparison/

Szyperski, C. (1998). Component Software. Beyond Object-Oriented Programming.Addison-Wesley.

Universidad Carlos III de Madrid: Estudio de UWE - Metodología de Desarrollo Web:http://es.scribd.com/doc/44936310/Estudio-de-UWE-Metodologia-de-Desarrollo-Web

ANEXOS

Anexo A diccionario de clase de las diferentes tablas creadas

Líneas de investigación		
Nombre	Tipo	Descripción
nombre	string	Nombre de la línea de investigación
descripción	string	Descripción de la línea de investigación
tipoLineas	string	Tipo de línea de acuerdo a la clasificación de Colciencias

Tabla colección de líneas de investigación

Programa Académico		
Nombre	Tipo	Descripción
nombre	string	Nombre de la maestría
objetivoGeneral	string	Objetivos generales y específicos de la maestría
objetivosEspecificos	array	
mision	string	Misión de la maestría
vision	string	Visión de la maestría
nivelAcademico	string	Nivel académico del programa
unidadAcademica	string	Departamento al que pertenece el programa académico
codSnies	num	Código
añoInicio	date	Año de inicio del programa académico
titulo	string	Título del programa académico
duracion	int	Duración del programa
planEstudios	ref	Plan de estudios del programa
horarioClases	ref	Horario de clase
numCohortes	int	Numero semestres del programa
numEgresados	int	Total estudiantes egresados
cuposSemestre	int	Total numero de estudiantes por chortes
valorInscripcion	num/float	Valor inscripción del programa
valorMatricula	num/float	Valor matricula del programa
perfilAspirante (archivo)	string	Perfil estudiante aspirante
perfilgresado (archivo)	string	Perfil del egresado

dirección	string	Dirección de la planta física
telefono	num/float	Numero telefónico
correoElectronico	string	Correo electrónico
horarioAtencion	string	Horario de atención
reglamentoPostgrado (Archivo)	string	Reglamentos
reglamentoProfesores (Archivo)	string	Reglamentos
reglamentoInvestigacion (Archivo)	string	Reglamentos
formatos	string	Tipo de formato

Tabla colección Programa académico

Propiedad Intelectual		
Nombre	Tipo	Descripción
titulo	string	Nombre de la propieda intelectual
autores [ ]	ref	Nombre de autores
tipoPublicación	string	Titulo de la publicación
fechaPublicación	date	Fecha de la publicación
palabrasClaves	string	Palabas claves
revistaLibro	string	Revista de publicación
ciudad	string	Ciudad de publicación
paginas	int	Numero de paginas de la publicación
proyectoInvestigacion	list	Tipo de proyecto
Trabajo de investigacion	ref	Tipo de ttrabajo de investigación

Tabla Colección Propiedad Intelectual

Proyectos de Investigación (estudiantes)		
Nombre	Tipo	Descripción
autores	ref	Autores del proyecto
director	ref	Director del proyecto
codirector [ ]	ref	Codirector del proyecto
titulo	string	Titulo
Problema	string	Descripción del problema
objetivoGeneral	string	Obetivos generales
objetivosEspecificos	array	Objetivos específicos
palabrasClave STRING	array	Palabas claves
tipoProyecto	string	Tipo de proyecto
año	date	Año publicación
bibliografia	string	Bibliografía
estado [ ] EMBEBIDO	string	nombre
	date	fechaEntrega
	date	fechaAprobación

evaluador	string	Evaluador
lineasInvestigación [ ]	ref	Tipo de línea
objetivoPrograma	ref	Objetivo del programa

Tabla colección Proyectos Investigación

Personal		
Descripción	Nombre	Tipo
Nombre	nombres	string
Apellido	apellidos	string
Tipo de identificación	tipoIdentificacion	string
Numero de identificacion	numeroIdentificaion	string
Fecha de nacimiento	fechaNacimiento	date
Ciudad de nacimiento	ciudadNacimiento	string
País de nacimiento	paisNacimiento	string
Genero	genero	string
Estado civil	estadoCivil	string
Numero de hijos	numeroHijos	int
Ingresos familiares	ingresosFamiliares	string
Tipo de vivienda	tipoVivienda	string
Direccion	direccion	string
Estrato	estrato	int
Ciudad de residencia	ciudadResidencia	string
Correo	correoElectronico	string
Teléfono	telefonoContacto	string
Estudios Realizados (Embebida)		
Nivel de estudis	nivel	string
Títulos	titulo	string
Promedio académico	promedioAcademico	float
Fecha de egreso	fechaEgreso	date
Nombre de universidad	universidad	string
Nombre de trabajo de grado	trabajoGrado	string
Idiomas		
Nombre del idioma	Nombre	string
Nivel de lectural del idioma	nivelLectura	string
Nivel de escucha del idioma	nivelEscucha	string
Nivel de escritura del idioma	nivelEscritura	string
Nivel de habla del idioma	nivelHabla	string
Certificado el idioma	certificado	string
Punta de clasificación del idioma	puntajeClasificacion	string
Experiencia Laboral (Embebida)		
Nombre de la empresa	empresa	string
Fecha en la cual laboro	fecha	date

Tiempo el cual laboró	permanencia	string
Nombre del cargo	cargo	string
Motivo de retiro	motivoRetiro	string
salario	salarioRango	string
Distinciones (Embebida)		
Nombre de la distinción	nombre	string
Motivo de la distinción	motivo	string
Fecha de distinción	fecha	date
Institución que otorgo distincion	institucionOtorga	string

Tabla colección Persona