Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Sistema de información para la toma de decisiones de proyectos de investigación y de proyectos terminales del observatorio Tecno - Educativo

Cruz Pérez Deyvid Lucrecio 207200035

Gutiérrez Flores Jaime Ulises 207200132

Trimestre 13-P

Fecha de entrega: 21 de Agosto del 2013

Asesor: Silva López Rafaela Blanca, Profesora Titular B. Departamento de Sistemas

Contenido

Objetivo general

Diseñar e implementar un sistema de información que permita realizar extracción de datos a partir de textos, para la generación de reportes a detalle y de alto nivel de los proyectos de investigación del Departamento de Sistemas, así como la gestión de proyectos terminales en el observatorio Tecno – Educativo.

Objetivos específicos

1. Diseñar e implementar el módulo de carga masiva de datos de los proyectos de investigación del Departamento de Sistemas en el observatorio Tecno – Educativo.
2. Diseñar e implementar el módulo de gestión de reportes programados de proyectos de investigación.
3. Diseñar e implementar el módulo para la gestión de reportes dinámicos de proyectos de investigación.
4. Diseñar e implementar el módulo para la gestión de reportes específicos de proyectos de investigación.
5. Diseñar e implementar el módulo para la gestión de propuestas de proyectos terminales.
6. Diseñar e implementar el módulo para la gestión de proyectos terminales concluidos.

Introducción

El Departamento de sistemas de la División de Ciencias Básicas e Ingeniería de la UAM-A. cuenta con un espacio virtual denominado: Espacio virtual del observatorio Tecno – Educativo, este espacio virtual está enfocado a la gestión únicamente de proyectos de investigación registrados y publicados por los profesores investigadores del departamento, dicho espacio virtual se encuentra colocado en el portal semántico del departamento de sistemas montado en el siguiente link: www.sistemas.azc.uam.mx. Sin embargo, el observatorio Tecno - Educativo aún no cuenta con funciones que permitan un mejor uso y aprovechamiento del mismo, es por ello que este proyecto consiste en diseñar e implementar un sistema de información para dicho observatorio, que permita crear, consultar y eliminar reportes de los proyectos de investigación, estos reportes serán de tres tipos: programados, dinámicos y específicos. Así mismo este proyecto estará enfocado en integrar la segunda fase de dicho espacio virtual, que consiste en implementar el módulo de gestión de propuestas de proyectos terminales y el módulo de gestión de proyectos terminales. El sistema permitirá el registro de propuestas de proyectos terminales por parte de los profesores y la publicación de proyectos terminales concluidos por parte de los alumnos respectivamente.

Justificación

Actualmente, existe un espacio virtual en el portal del Departamento de Sistemas de CBI de la UAM-A, denominado espacio virtual del observatorio Tecno – Educativo, que permite a los profesores investigadores de este plantel llevar un adecuado control de los proyectos de investigación, en los que se encuentra trabajando cada área de investigación de los

departamentos de la universidad. Sin embargo, este espacio virtual no cuenta con un sistema para generar reportes de los proyectos de investigación. Surge así la necesidad de crear un sistema que permita generar los reportes adecuados según sea la necesidad del usuario, es decir, de acuerdo a la información que el usuario requiera para su reporte. El reporteador propuesto permitirá al usuario según sea el tipo: primario (administrador, jefe de departamento), secundario (profesor investigador, jefe de área) gestionar eficientemente los reportes de acuerdo a sus privilegios. Este sistema reporteador permitirá al usuario primario crear, eliminar y consultar reportes de tipo: programables (información trimestral y anual), dinámicos (información seleccionada por el usuario y requerida en cualquier momento por el mismo), y de tipo específico (reportes que contengan información específica sobre un campo de información), mientras que el usuario secundario solo podrá realizar consultas de dichos reportes.

Por otra parte el Observatorio Tecno – Educativo: no cuenta con un espacio que permita a los profesores realizar la gestión de propuestas de proyectos terminales y a los alumnos realizar la publicación de sus proyectos terminales concluidos. Por esta razón, la segunda fase de este proyecto consiste: en diseñar e implementar los módulos respectivos a estas operaciones, con la finalidad de complementar el espacio virtual del Departamento de Sistemas de la División de Ciencias Básicas e Ingeniería de la UAM-A. Estos módulos del sistema propuesto permitirán realizar las operaciones de altas, bajas, consultas y reportes, tanto de propuestas de proyectos terminales sugeridas por los profesores, como de los proyectos terminales publicados por los alumnos.

La razón por la cual es apropiado que un ingeniero en computación realice este proyecto es: porque éste cuenta con las características y conocimientos adecuados para la correcta realización de dicho proyecto. Conocimientos de Programación, Bases de datos, Metodologías de análisis y diseño de sistemas de información, Ingeniería de software, Interoperabilidad, Estructura de datos con orientación a objetos, Diseño de algoritmos, entre otros, por lo cual una persona que no cumple con este perfil no podría realizar este proyecto.

Antecedentes

Referencias Internas

El proyecto “Sistema de Información para la toma de decisiones de Proyectos de Investigación y proyectos terminales del observatorio Tecno - Educativo” tiene como objetivo: el apoyar a los profesores, personal administrativo y alumnos de la UAM-A, en la gestión de reportes correspondientes a un proyecto de investigación, así como en la gestión de proyectos terminales colocados en el observatorio Tecno - Educativo según sea el caso.

En la UAM-A se han realizado 6 proyectos terminales similares a la propuesta aquí mencionada.

1. *Implementación de un gestor de documentos* [1]. Este proyecto terminal está enfocado al diseño de una aplicación para la gestión de documentos.

2. *Sistema generador de reportes del puerto Ethernet* [2]. Este proyecto terminal está enfocado en desarrollar un sistema generador de reportes de recepción y transmisión de datos del puerto Ethernet de un servidor.

3. *Gestor de contenidos de sitios web* [3]. Este proyecto terminal está enfocado en diseñar e implementar un gestor de contenidos de sitios web.

4. *Sistema de Gestión de Congresos* [4]. Este proyecto terminal está enfocado en el control de los congresos de una organización.

5. *Prototipo Web para Gestionar Proyectos de Software* [5]. Este proyecto está enfocado a gestionar proyectos de software.

6. *Espacios virtuales para el trabajo colaborativo del Observatorio Tecno – Educativo: Proyectos de investigación* [6]. Este proyecto terminal está enfocado en la creación de un espacio virtual para realizar la gestión de proyecto de investigación de los profesores de la UAM-A.

Sin embargo, no existe dentro de la UAM-A un proyecto que realice extracción de datos y la gestión de reportes de los proyectos de investigación que realizan los profesores de la UAM-A, así como la gestión de propuestas de proyectos terminales y la publicación de proyectos terminales concluidos.

Referencias Externas

Actualmente, existe software que cumple con las funciones de un reporteador. Estas aplicaciones las podemos encontrar en línea. Tal es el caso de las siguientes aplicaciones:

Herramienta de consultas RBT [7] es un software libre el cual es una aplicación de consultas y reportes basada en navegador, respaldada por la tecnología J2EE escalable[1]. RBT es una herramienta de diseño de reportes y consultas, basada en web, amigable y gratuita, con una aplicación cliente DHTML[2] y un soporte Java Enterprise. El tipo de reportes que genera esta aplicación puede ser utilizado en empresas de distintos rubros teniendo el usuario la facilidad de agregar o quitar columnas de acuerdo a las características del reporte deseado.

Stimulsoft Reports Designer.Web [8] es un software comercial de la empresa Stimulsoft que tiene un costo de $499.95 US y está hecho en PHP[3]. Esta aplicación se considera el primer diseñador de reportes que permite editar completamente reportes en la web a través de una plataforma Windows. Todo lo que se necesita es un explorador web con Flash Player 10[4].

[1] Java Plataform, Enterprise Edition. Plataforma de programación para desarrollar y programar software.
[2] Forma de combinar HTML, JavaScript, DOM, CSS.
[3] Lenguaje de programación interpretado, diseñado para la creación de páginas Web Dinámicas.
[4] Una aplicación en forma de reproductor multimedia que permite reproducir archivos SWF.

MyDBR [9] es un software comercial de MyDBR disponible en una versión libre y en una versión Premium con un costo de 139 EUR que proporciona una solución innovadora para la creación de reportes que está basado en web y que está hecho para los manejadores de bases de datos MySQL[5] y Microsoft SQL Server[6].Esta aplicación se encuentra hecha en PHP a través de una plataforma Windows y proporciona soporte a los teléfonos inteligentes más recientes (Android[7], iPhone[8] y Symbian[9]).

MySQL Report [10] es un software comercial de la empresa MySQL con un precio de $36 US y hecho en lenguaje PHP generador de reportes que permite crear y administrar un número ilimitado de Reportes MySQL basados en tablas o consultas. Es una amigable interfaz estilo asistente que permite seleccionar sólo los campos que le gustan al usuario con la finalidad de ver en el informe la información que desee y convenga. Muestra un Set de múltiples niveles de agrupación, permite ordenar los datos de manera ascendente o descendente y tiene la capacidad de exportar el informe en formato PDF[10], XML[11] o archivos CSV[12]. Este reporteador se puede encontrar en su modo de prueba o bien se puede adquirir mediante la compra de una licencia.

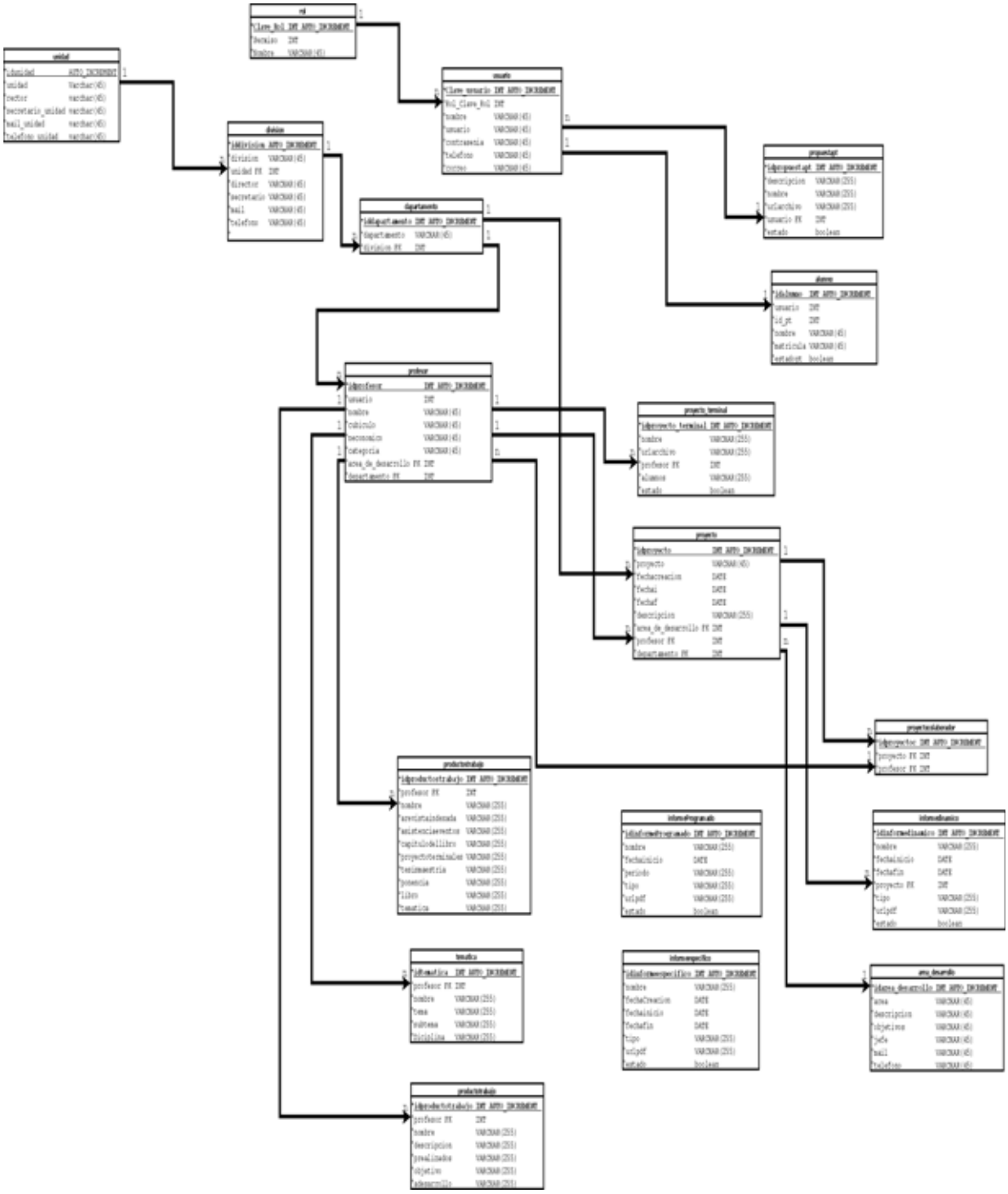
LogAleph [11] es un software libre hecho en lenguaje PHP. Esta aplicación es un generador de reportes que fue implementado para las bibliotecas del SUBA (Unidad de Sistemas y Automatización de Bibliotecas) de la Universidad Autónoma de Chihuahua, entre otras, con la finalidad de tener un control estadístico del uso de Acervos y materiales vencidos de la biblioteca. La aplicación muestra una interfaz amigable para los administradores de servicios bibliotecarios de la universidad con la finalidad de proporcionar un mejor uso de los mismos. Este generador de reportes trabaja en una plataforma Windows, por lo cual los reportes generados utilizan la extensión .xls.

La comparación de las características de los proyectos y software relacionados con esta propuesta se muestra en la Tabla 1.

[5] Sistema de gestión de base de datos relacional, multihilo y multiusuario.
[6] Sistema para la gestión de base de datos producido por Microsoft basado en el modelo racional.
[7] Sistema operativo móvil desarrollado en Linux desarrollado por la Open Handset Aliance.
[8] Teléfono inteligente (Smart phone) con la capacidad de textos táctiles desarrollado por Apple.
[9] Sistema operativo utilizado en PDA´s y Handhelds de PISON.
[10] Formato de almacenamiento de documentos desarrollado por la empresa Adobe Systems.
[11] Metalenguaje extensible desarrollado por el World Wide Web Consortion.
[12] Tipo de documento en formato abierto para representar datos en forma de tabla.

Esquema de la Base de Datos

En la siguiente figura se encuentra el esquema general de la base de datos modelado en el diagrama de clases representando una perspectiva estática del Sistema de información para la toma de decisiones:



Usuarios

El Sistema de información para la toma de decisiones de proyectos de investigación y de proyectos terminales permite gestionar a los diferentes usuarios a través de una validación.

Usuarios sección de Propuestas de Proyectos Terminales

a) El usuario primario (profesor) podrá realizar, altas, bajas, cambios y consultas de sus propuestas de proyectos terminales, tal y como se muestra en los siguientes diagramas de secuencia y actividad:

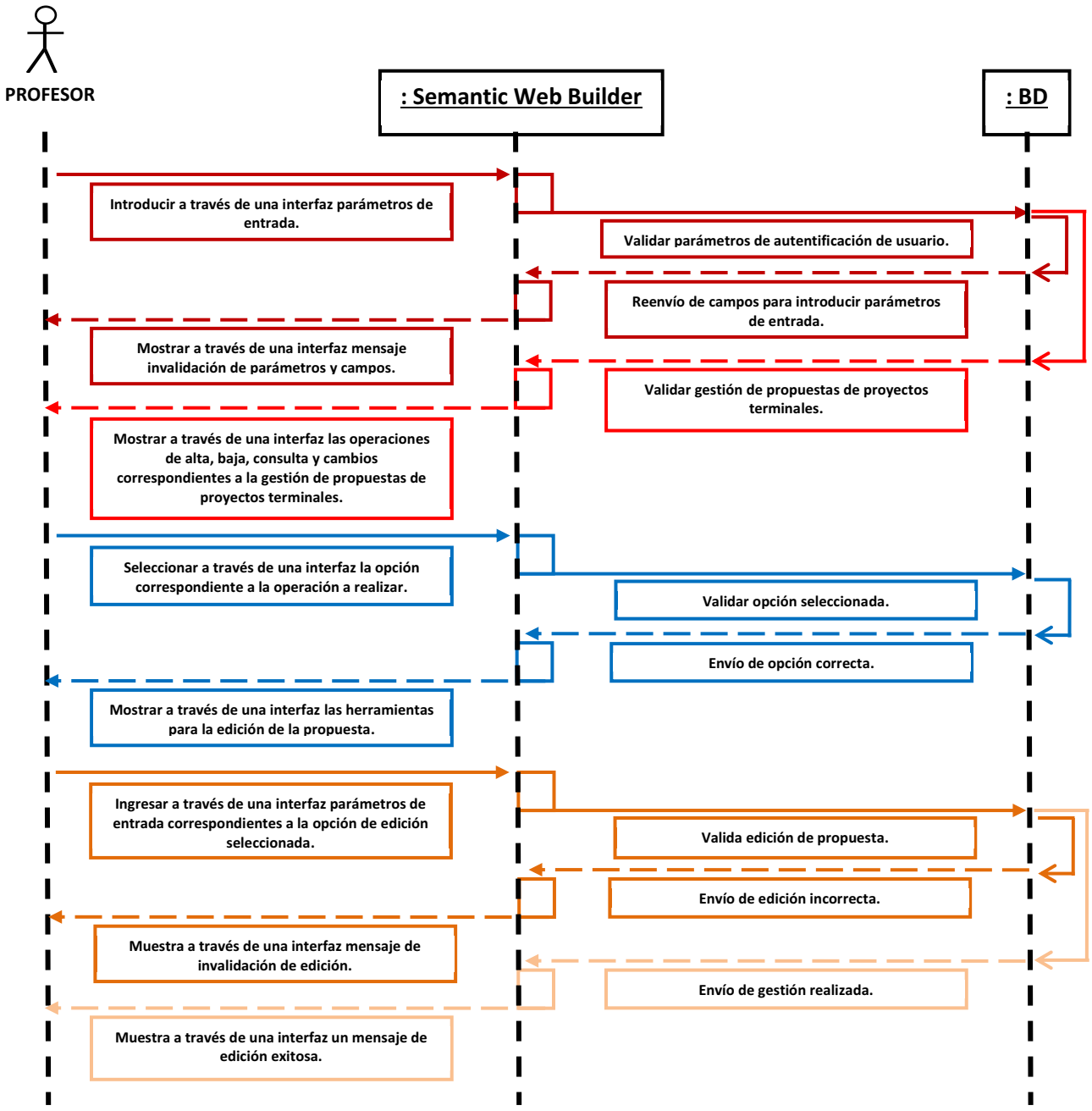
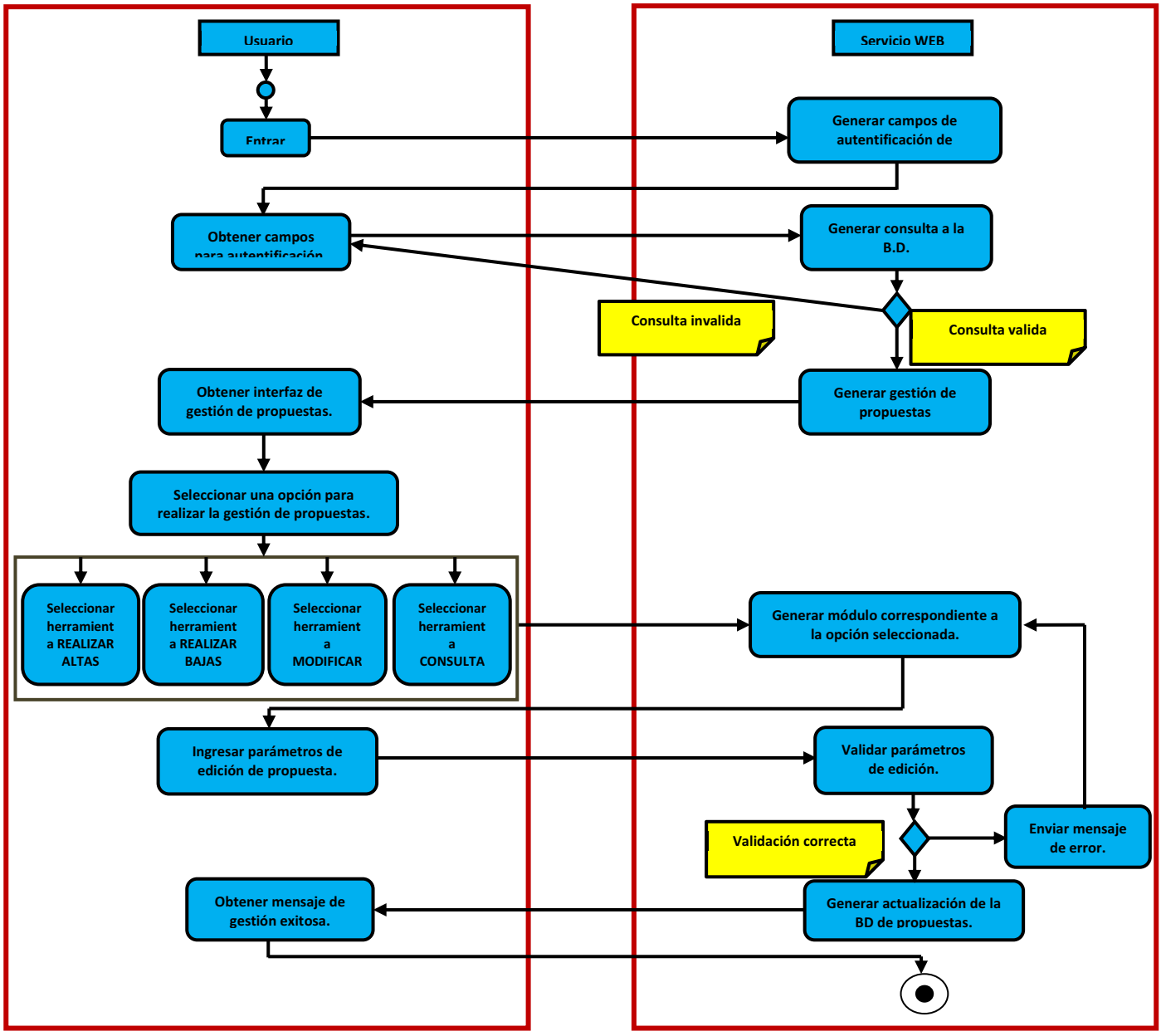


DIAGRAMA DE ACTIVIDAD DE PROFESORES



b) El usuario secundario (alumno) podrá realizar únicamente consultas de las propuestas de proyectos terminales que los profesores publicaron en el servidor, tal y como se muestra en los siguientes diagramas de secuencia y actividad:

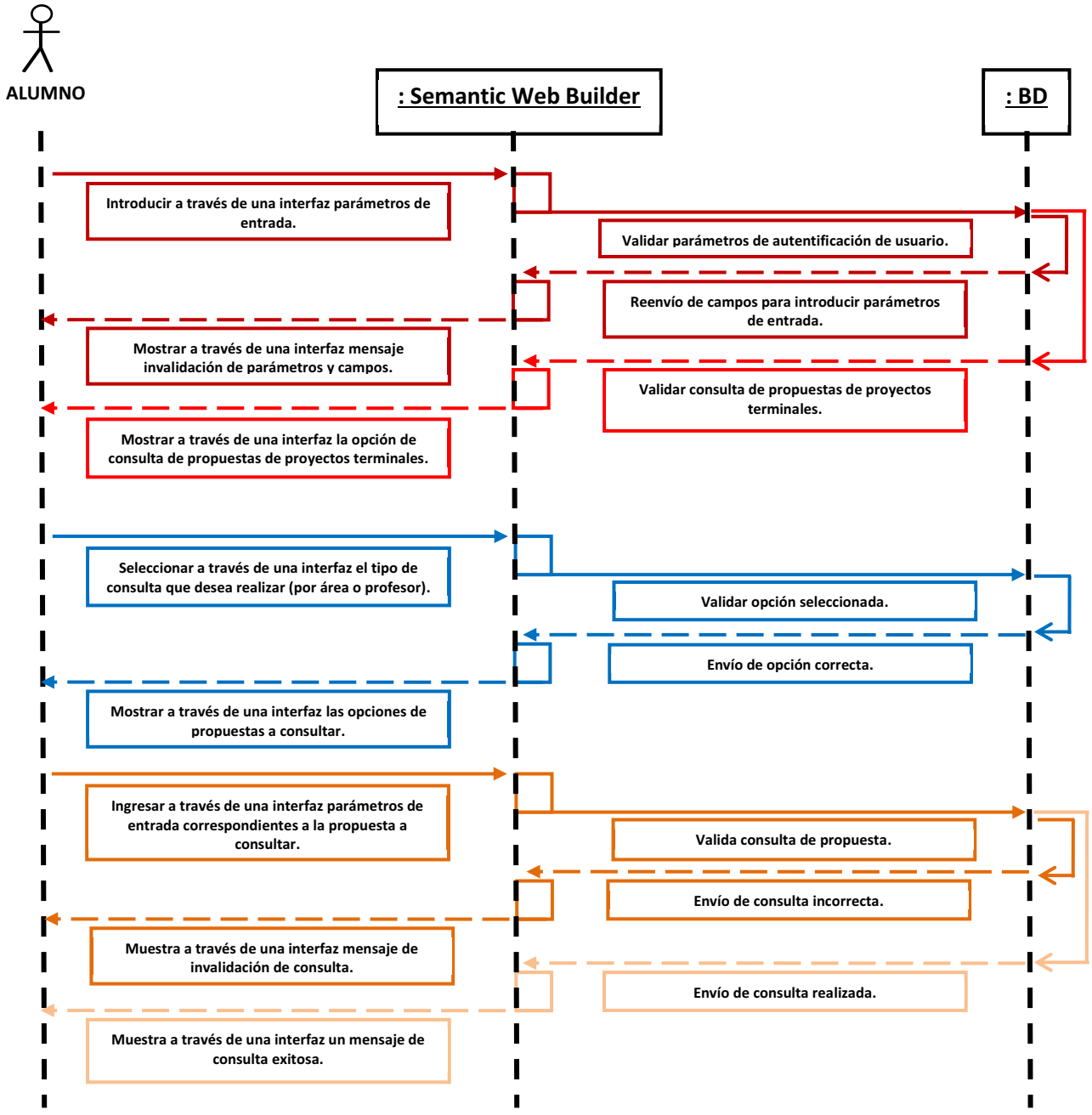
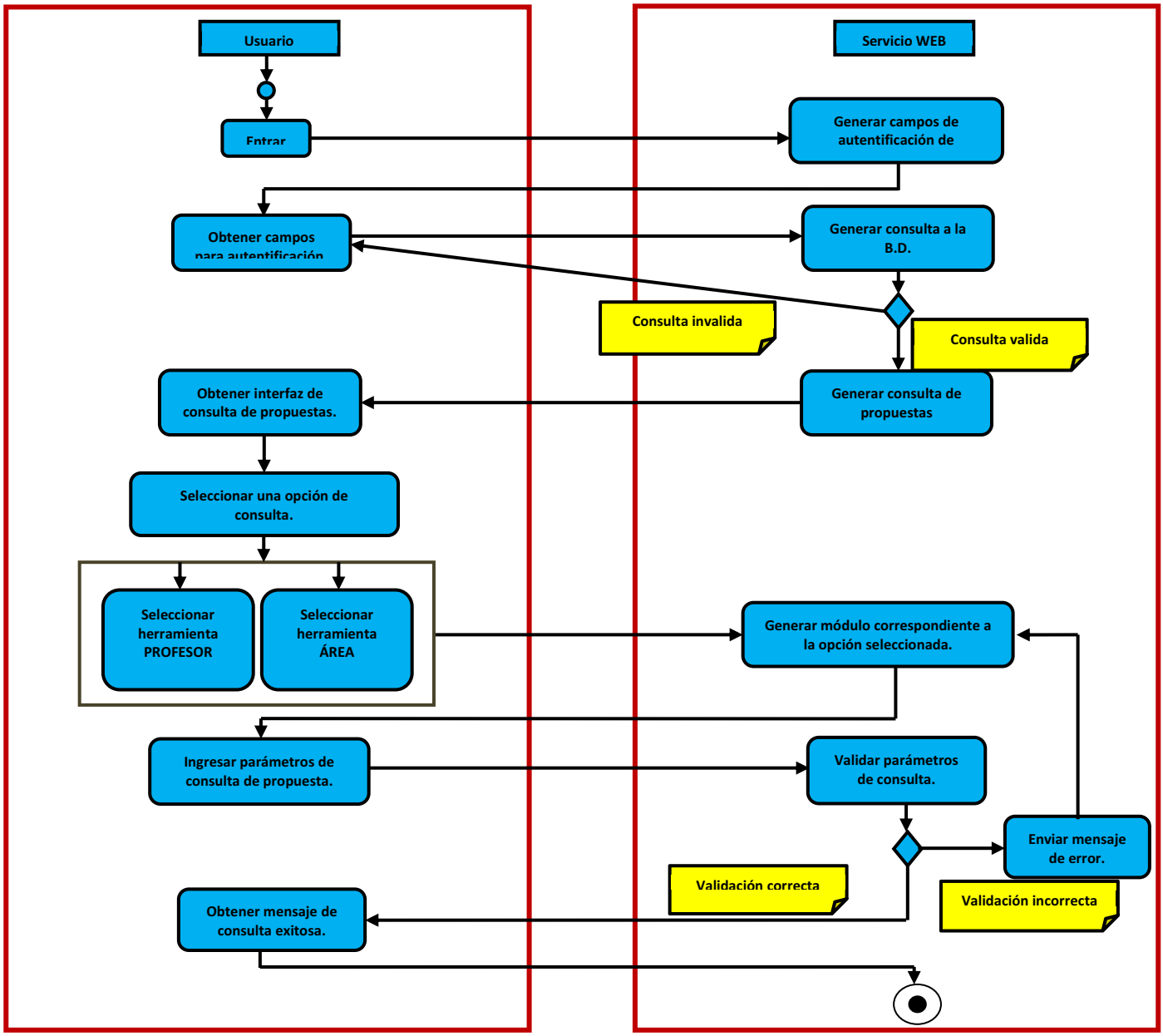


DIAGRAMA DE ACTIVIDAD DE ALUMNOS



Usuarios sección de Proyectos Terminales

a) El usuario primario (alumno(s) desarrollador(es)) podrá realizar únicamente altas, bajas y consultas del proyecto terminal concluido, tal y como se muestra en los siguientes diagramas de secuencia y actividad:

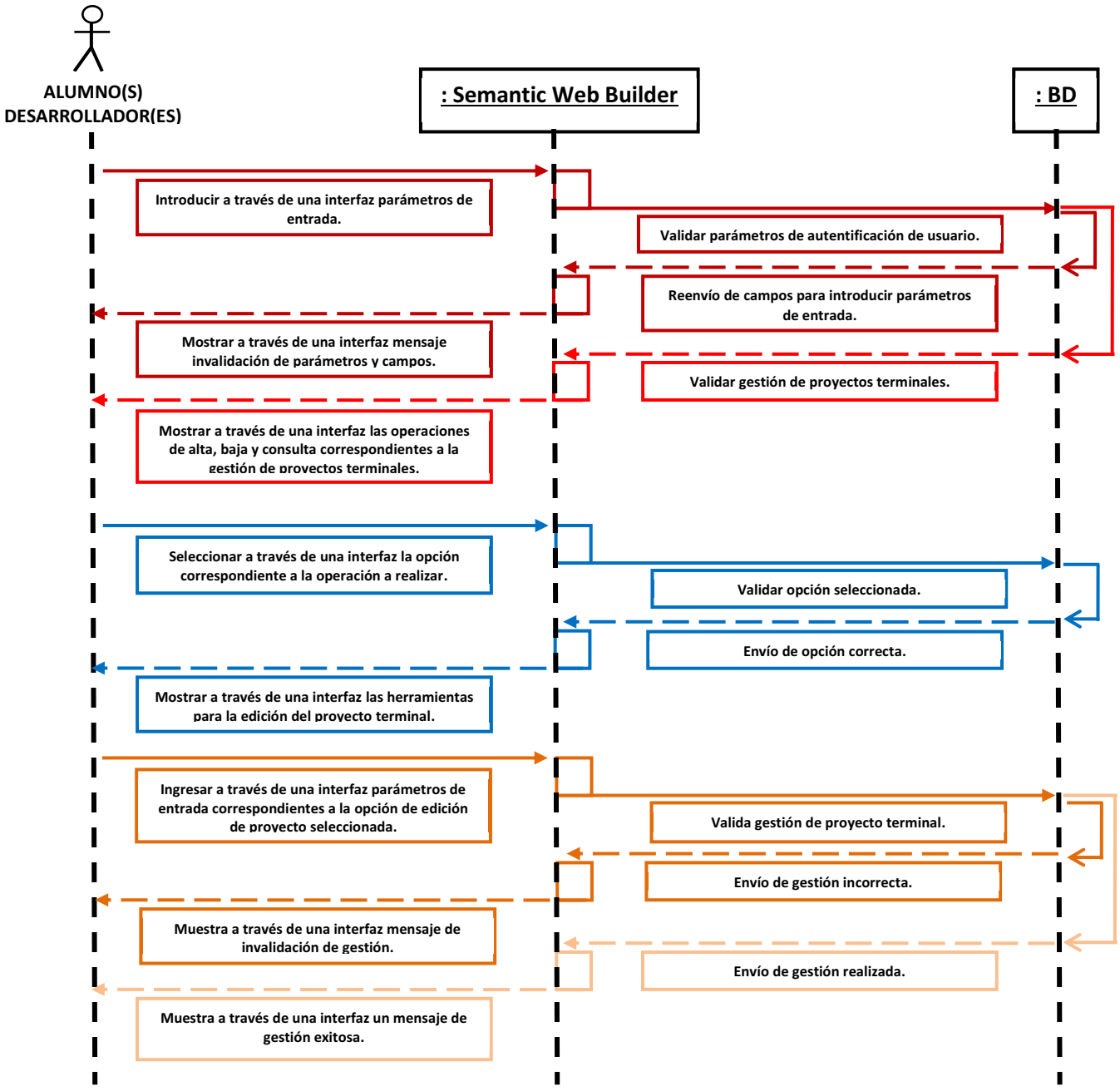
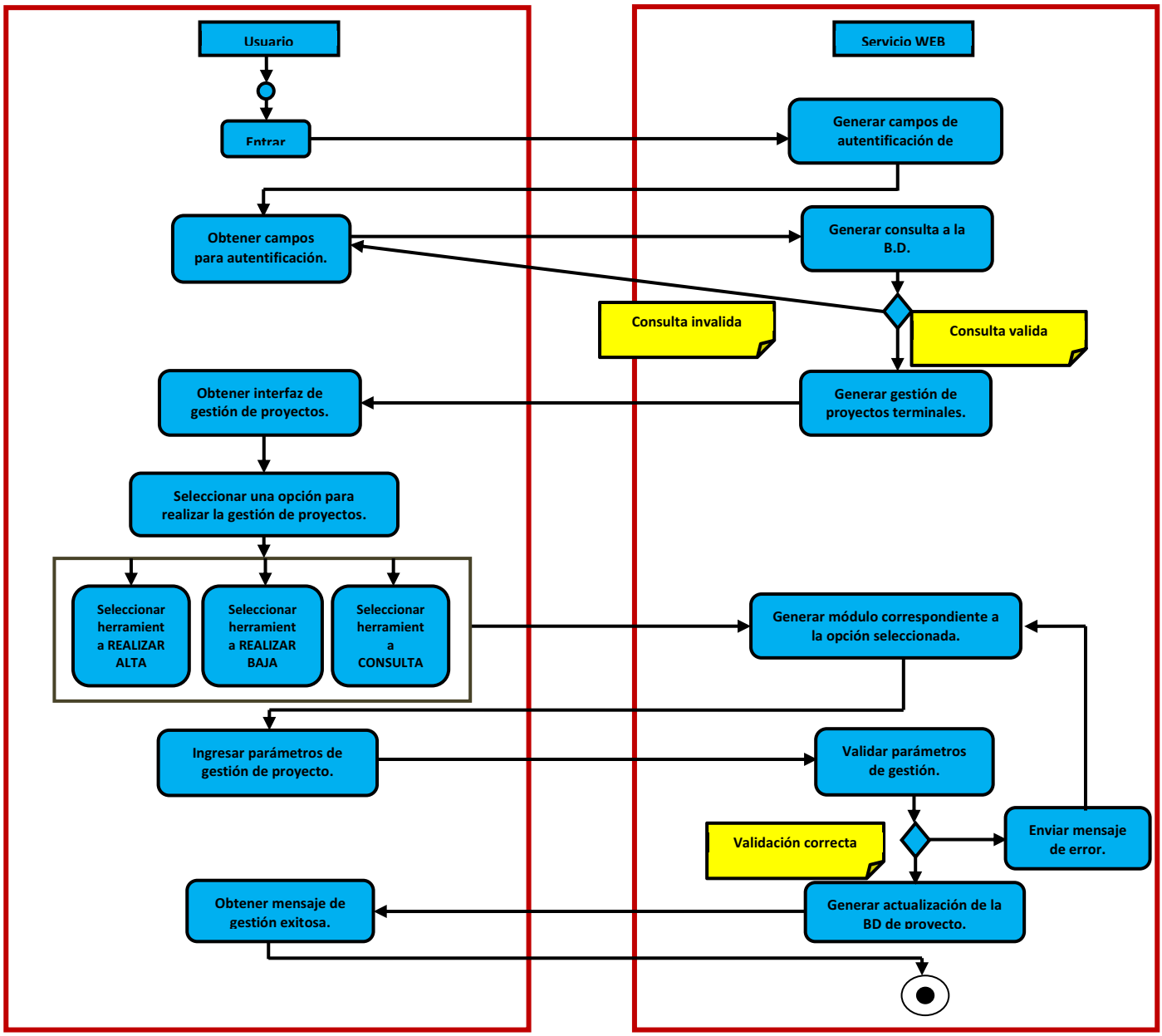


DIAGRAMA DE ACTIVIDAD DE ALUMNOS DESARROLLADORES



b) El usuario secundario podrá realizar únicamente consultas de los proyectos terminales que los alumnos publicarán en el servidor, tal y como se muestra en los siguientes diagramas de secuencia y actividad:

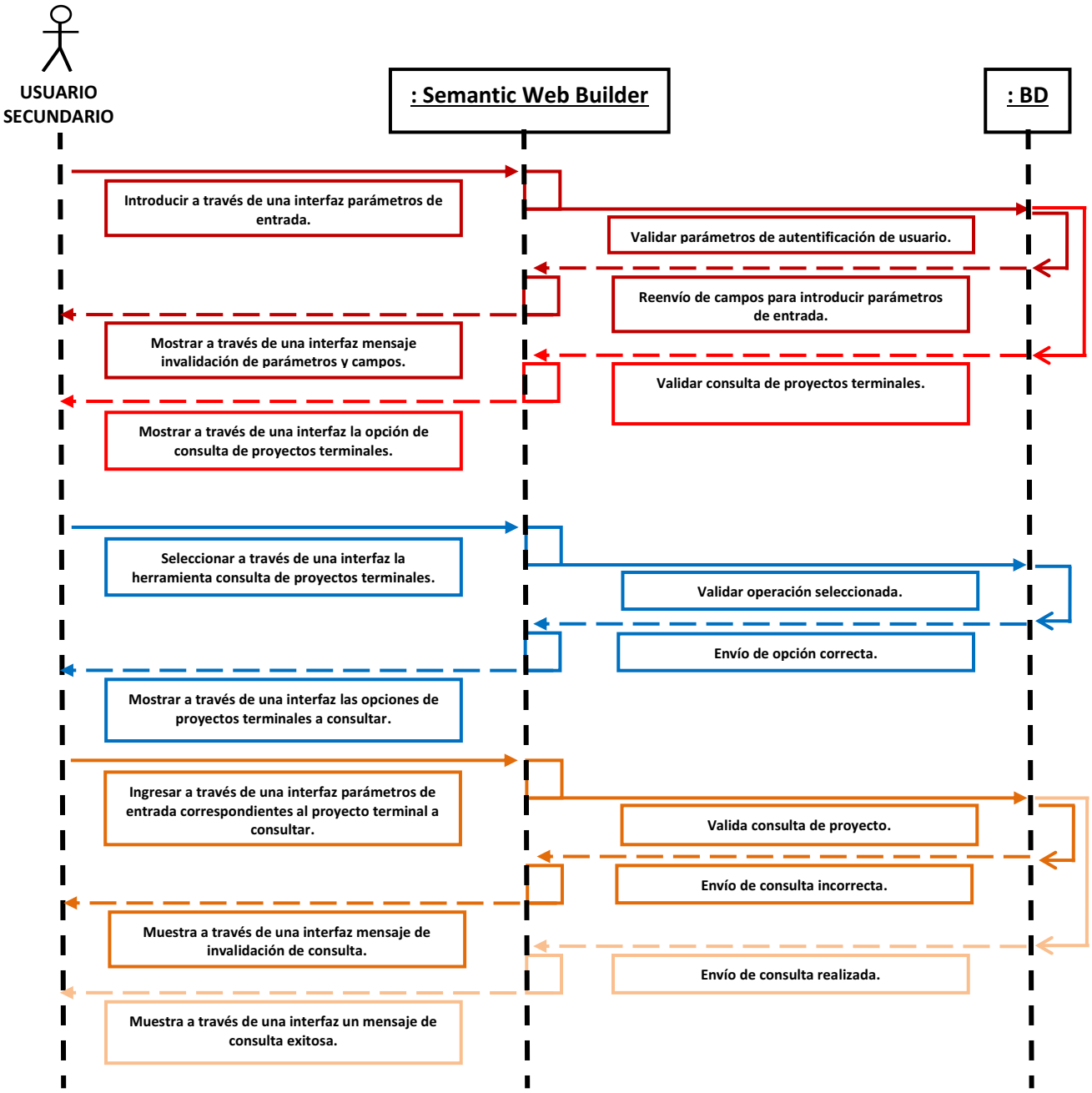
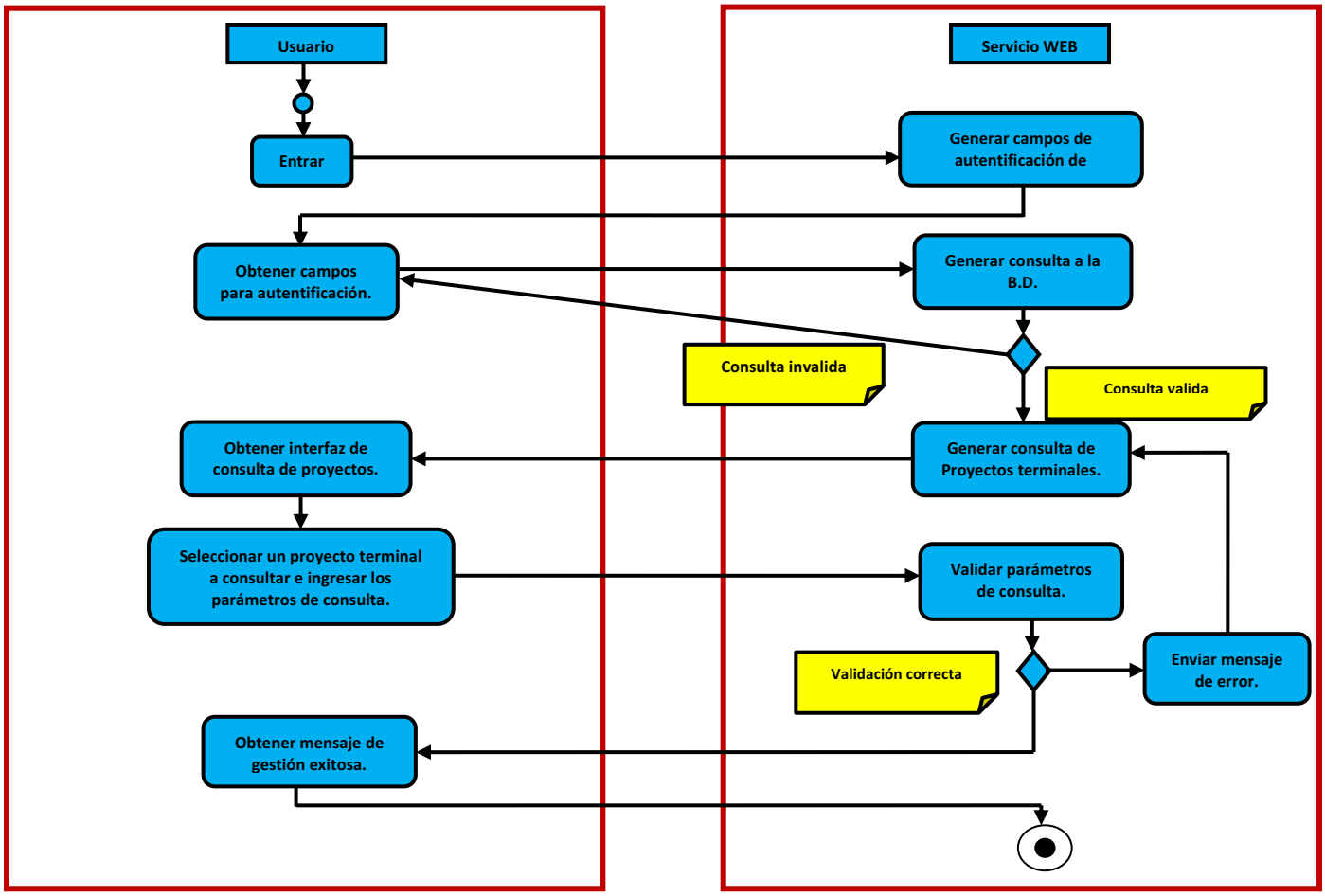


DIAGRAMA DE ACTIVIDAD DE USUARIO SECUNDARIO



Usuarios sección de Reportes

a) El administrador podrá crear, consultar y eliminar cualquier tipo de reporte correspondiente al proyecto de investigación autorizado tal y como se muestra en los diagramas de secuencia y actividad:

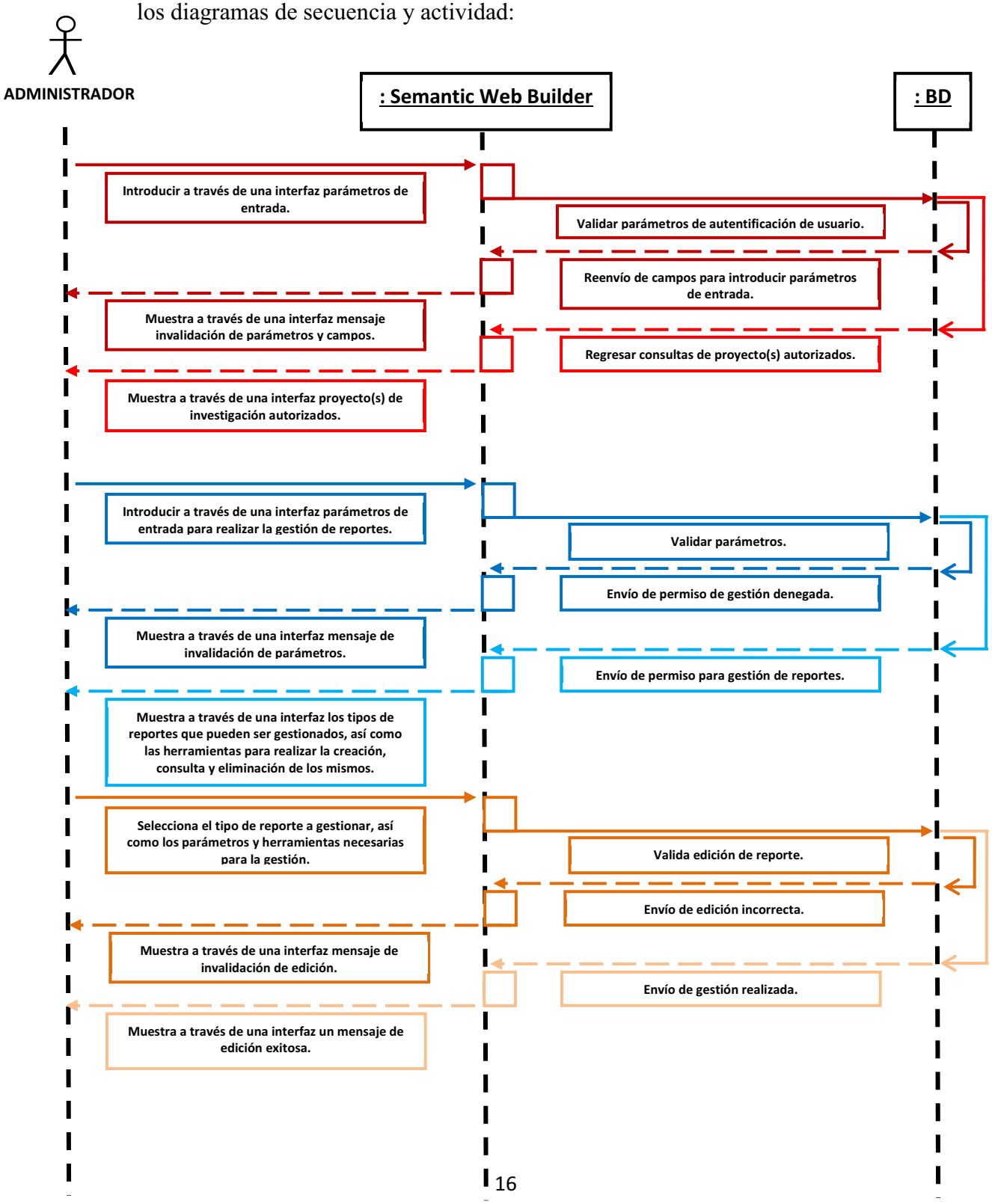
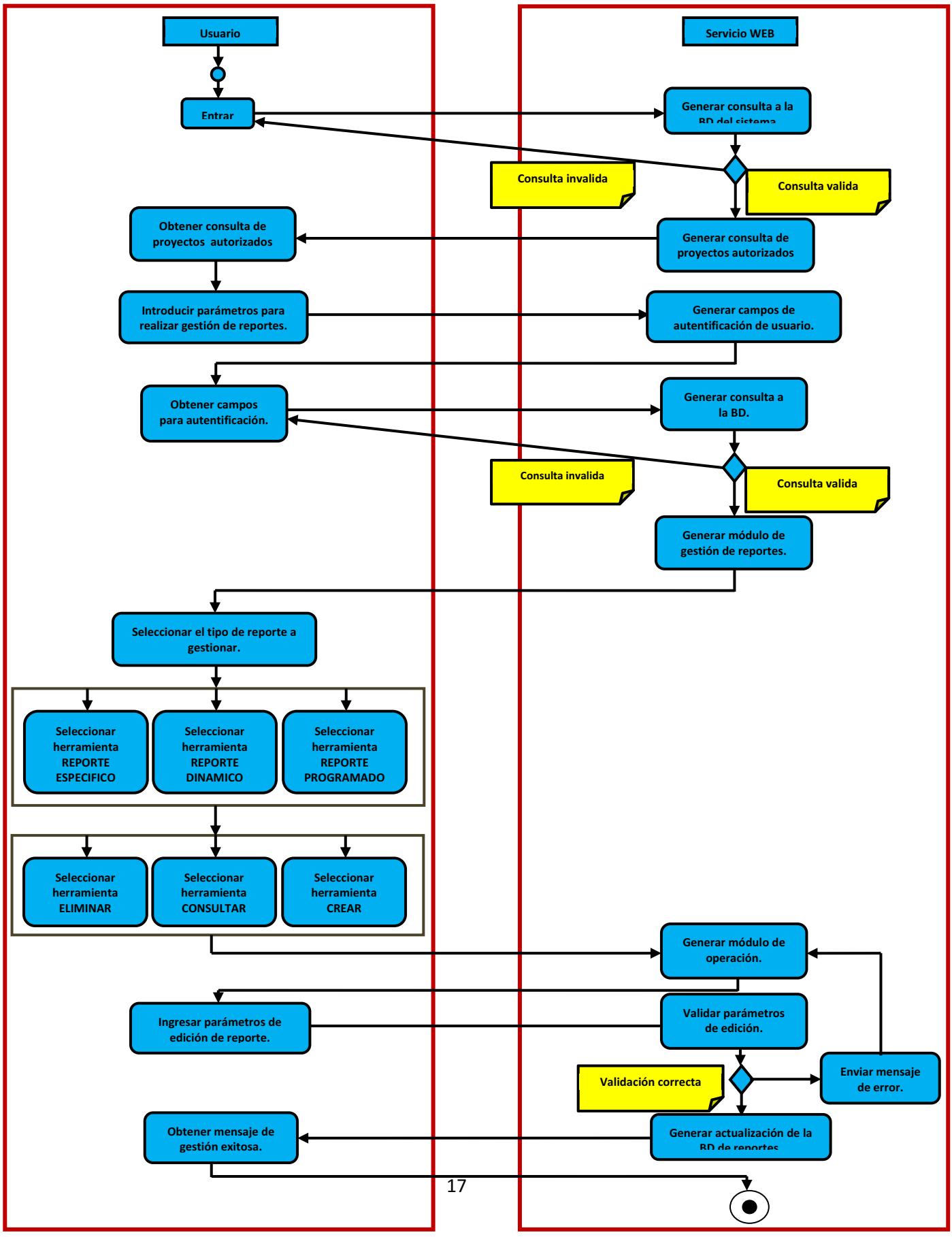


DIAGRAMA DE ACTIVIDAD DE ADMINISTRADOR



b) El Jefe de departamento podrá crear, consultar y eliminar los reportes dinámicos y específicos mientras que solo podrá consultar y eliminar los reportes programados correspondientes al proyecto de investigación autorizado tal y como se muestra en los diagramas de secuencia y actividad:

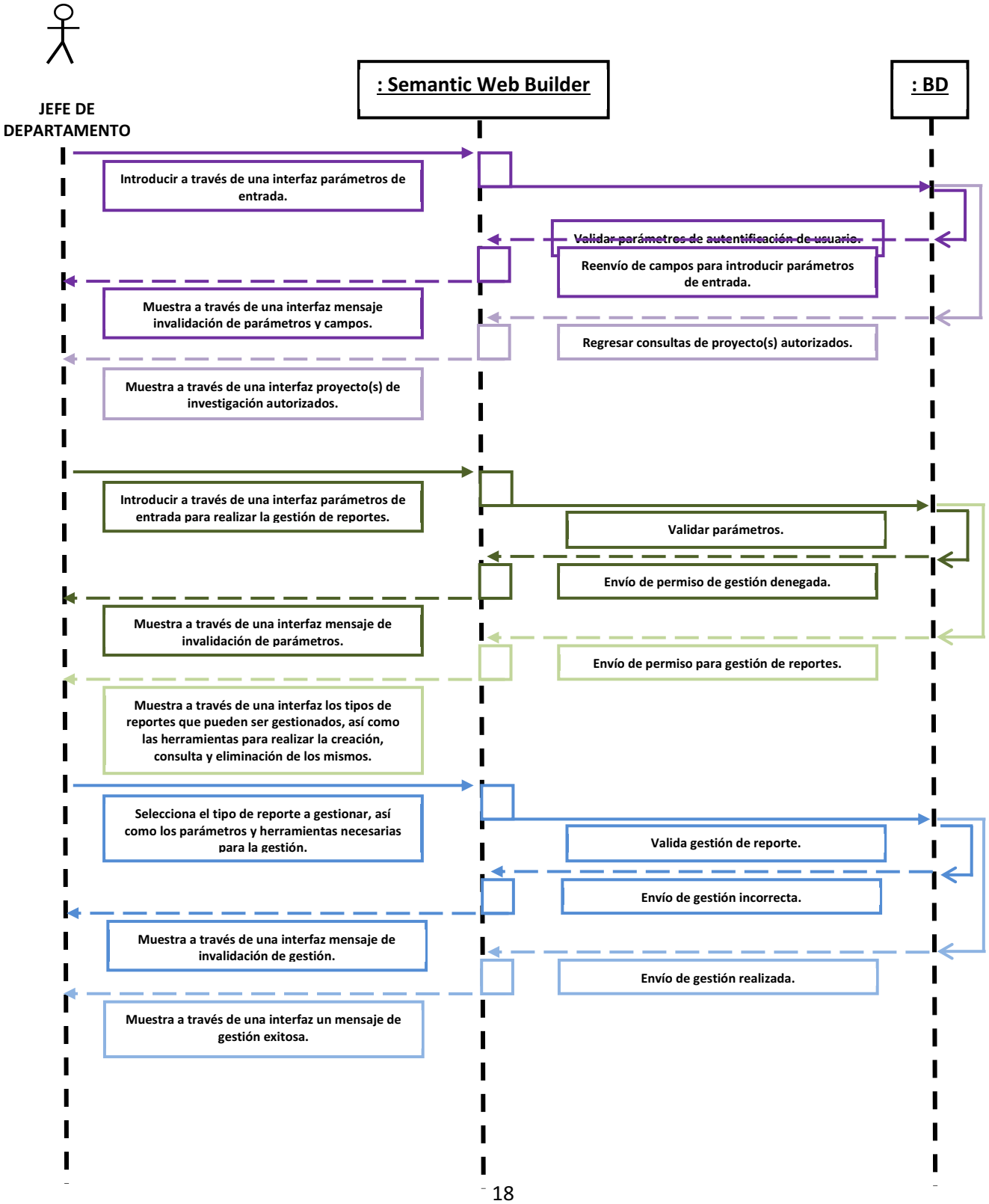
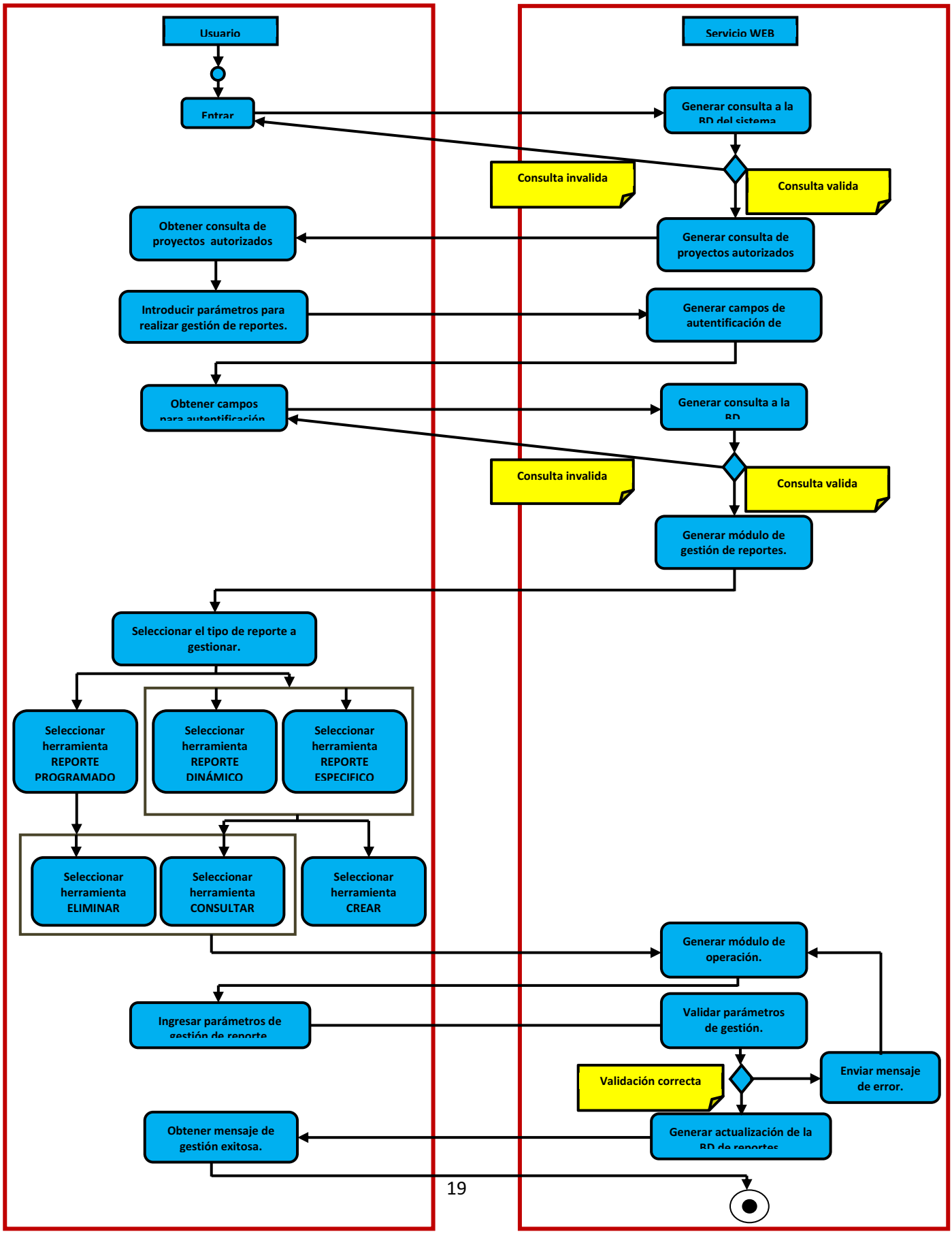


DIAGRAMA DE ACTIVIDAD DE JEFE DE DEPARTAMENTO



c) El usuario secundario (jefe de área y profesor investigador) solo podrá realizar consulta de los reportes (programados, dinámicos y específicos) del proyecto de investigación tal y como se muestra en los diagramas de secuencia y actividad:

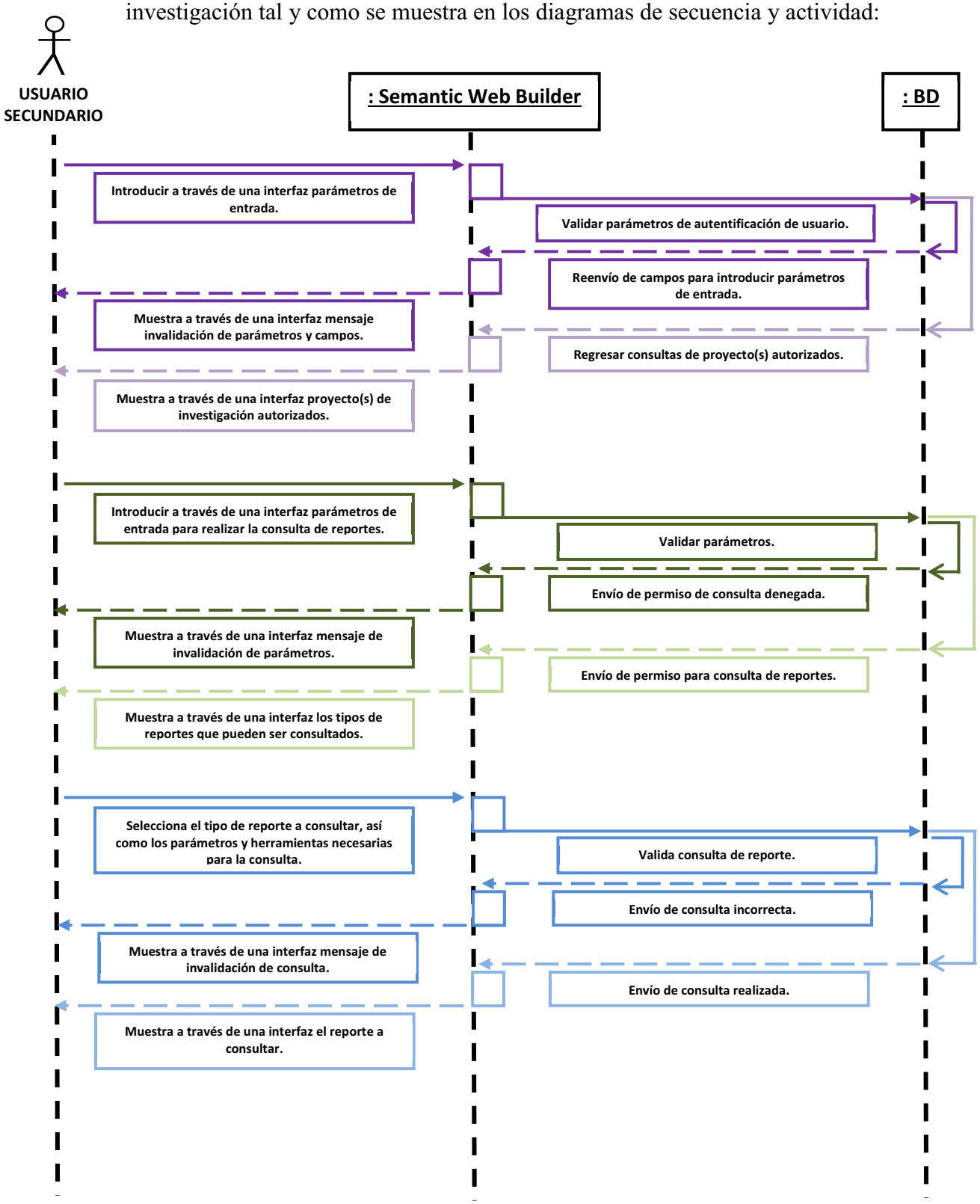
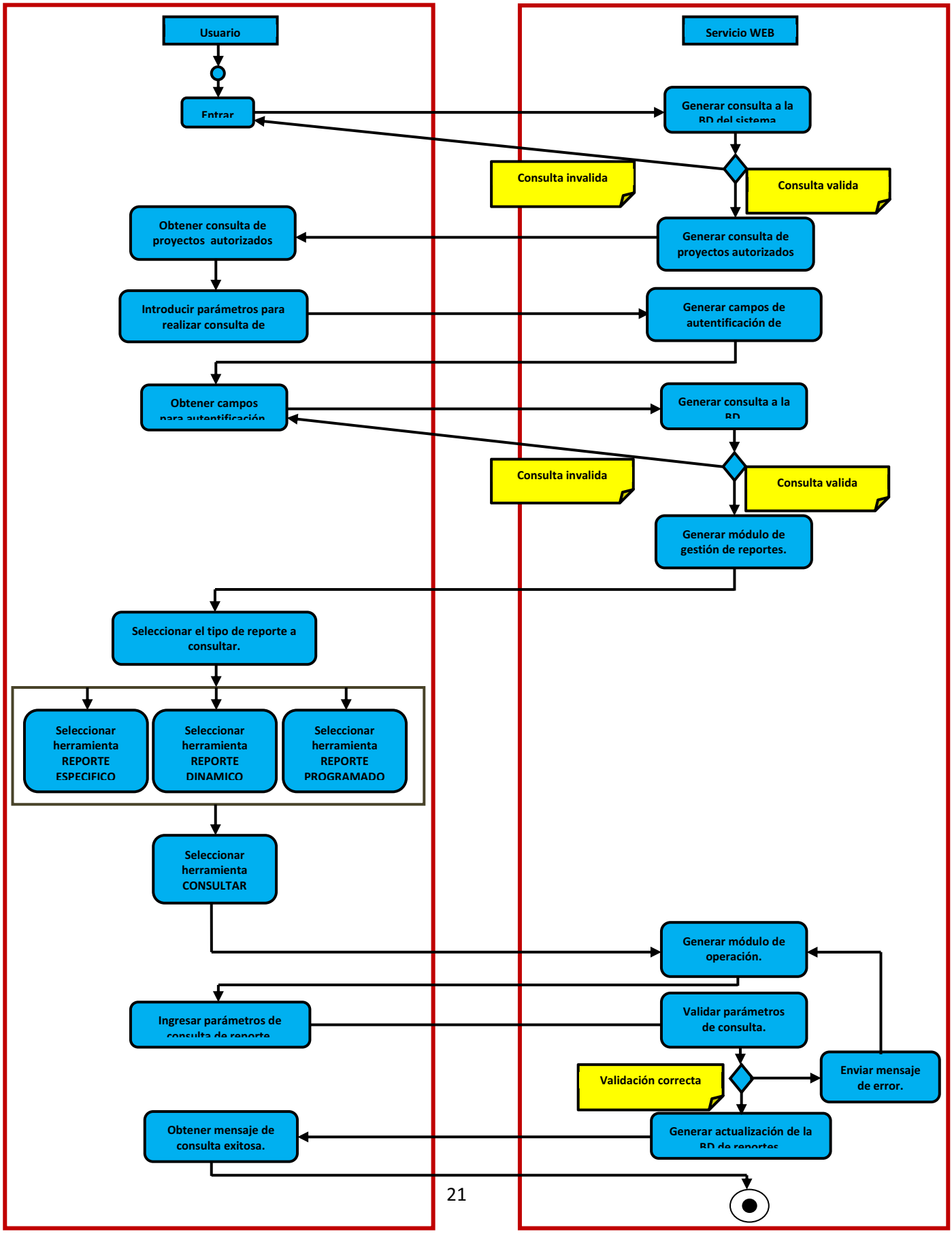
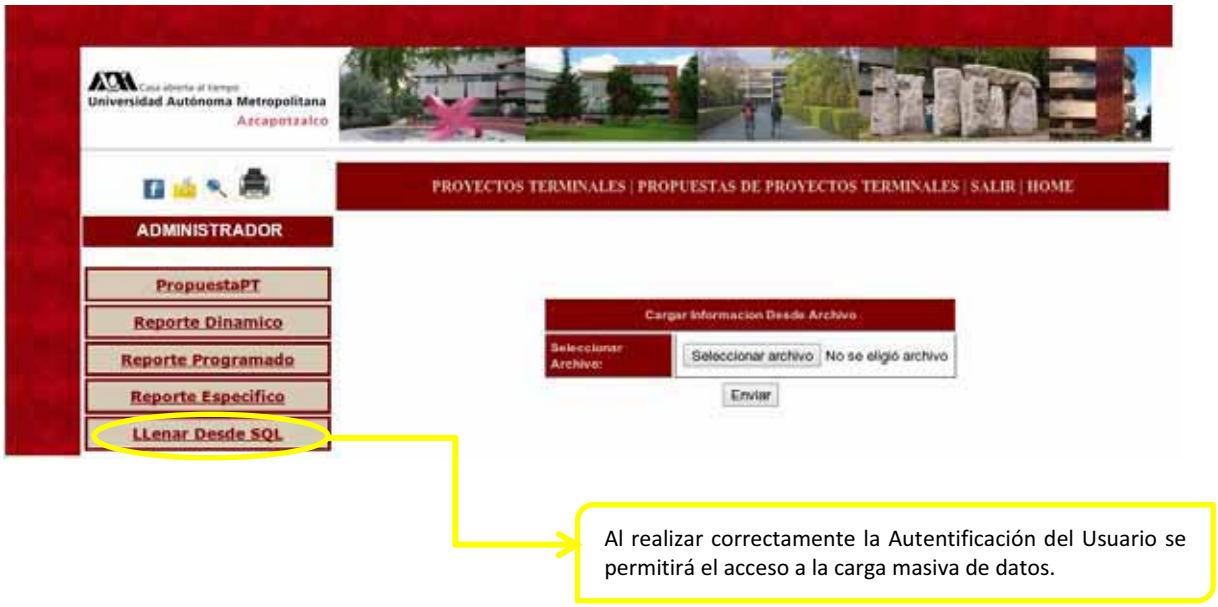


DIAGRAMA DE ACTIVIDAD DE USUARIO SECUNDARIO



MODULOS

El sistema se dividió en 6 módulos dividido en 3 secciones:



1. Llenar desde SQL

   1.1 Módulo de Carga Masiva de Datos

Permite el ingreso de datos consistentes a los módulos en el espacio virtual a través de un s*cript*. El diseño e implementación y repositorio de DWH se muestra en la figura 1.

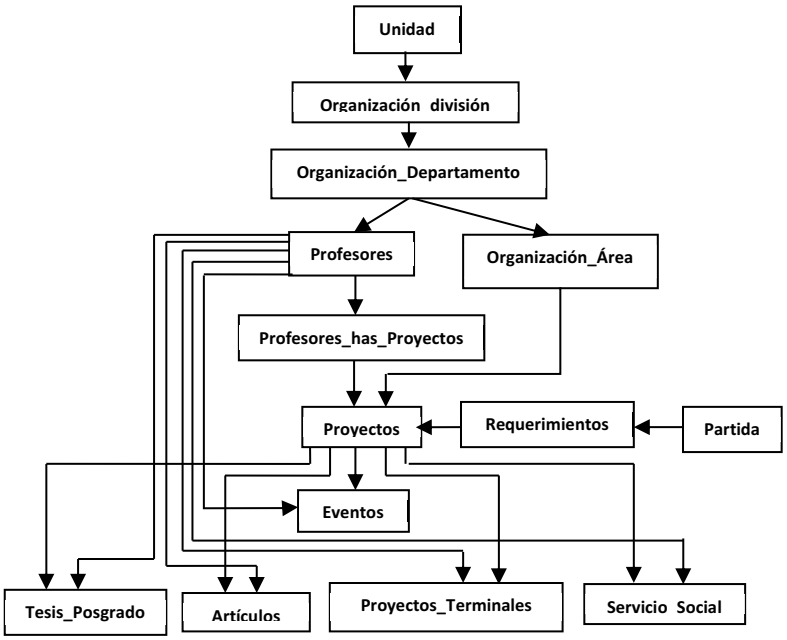


Figura 1. Relación de las entidades que integran la Base de Datos.

El Diseño e implementación del proceso para realizar la carga masiva de información a la base de datos se muestra en la figura 2.

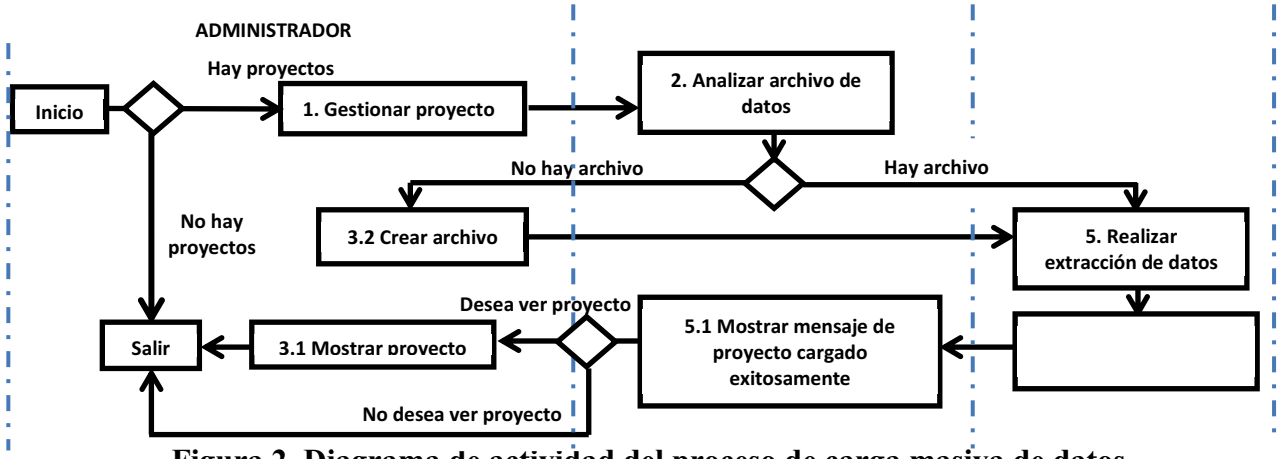


Figura 2. Diagrama de actividad del proceso de carga masiva de datos.

2. Reportes

2.1 Módulo de gestión de Reportes Programados

Permite gestionar los reportes que serán programados en distintos lapsos de tiempo (trimestral y anual), de acuerdo a ciertas características establecidas para el reporte. Como se muestra en la figura 3, el reporte será creado por el administrador de manera automatizada, por lo cual el jefe de departamento solo podrá consultar y eliminar el reporte, mientras que el jefe de área y el profesor investigador solo podrán realizar consultas de los reportes generados.

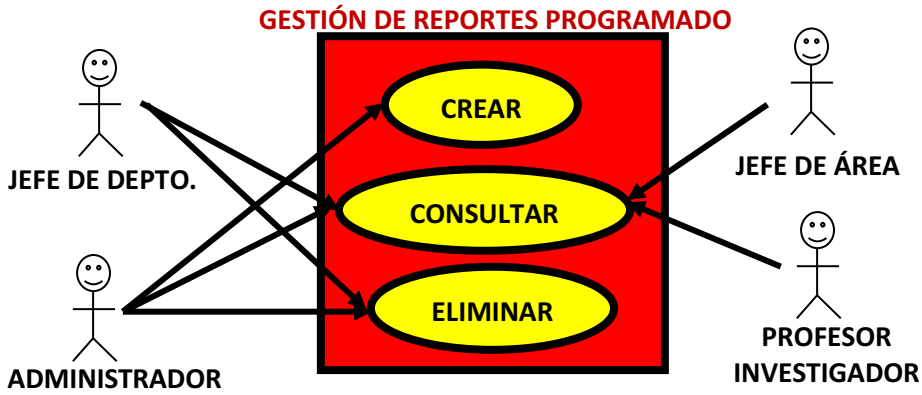


Figura 3. Diagrama de casos de uso de la gestión de reportes programados.

El diseño del proceso para la generación de reportes programados se muestra en la figura 4.

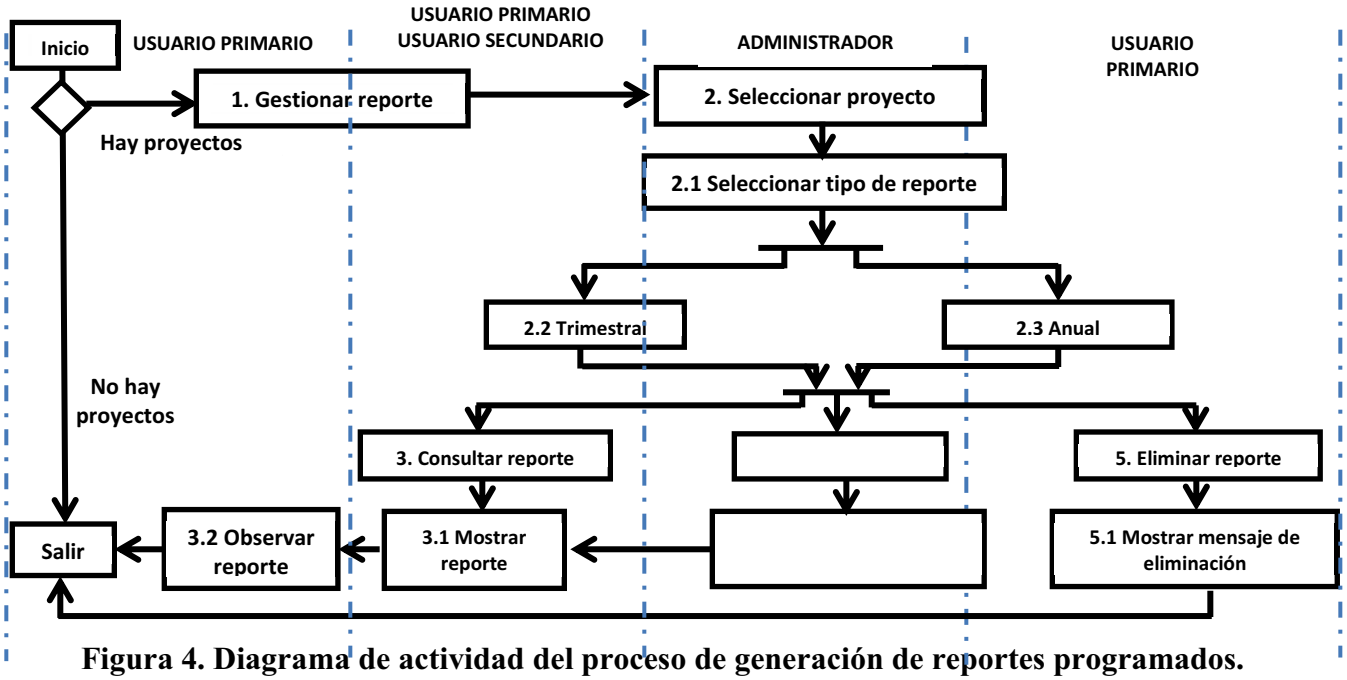


Figura 4. Diagrama de actividad del proceso de generación de reportes programados.

2.2 Módulo de gestión de Reportes Dinámicos

Permite al usuario seleccionar que elementos de la base de datos requiere para su reporte. Como se observa en la figura 5, el administrador y el jefe de departamento podrán crear, eliminar y consultar los reportes dinámicos, mientras que el jefe de área y el investigador solo podrán consultar el reporte.

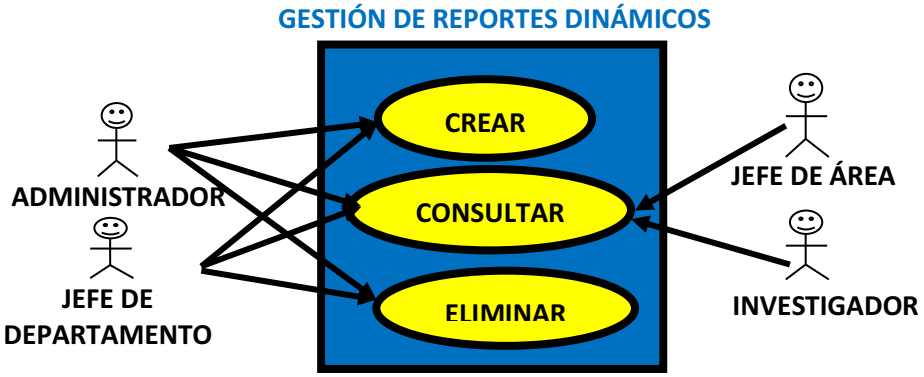


Figura 5. Diagrama de casos de uso de la gestión de reportes dinámicos.

El diseño del proceso para la generación de reportes dinámicos se muestra en la figura 6.

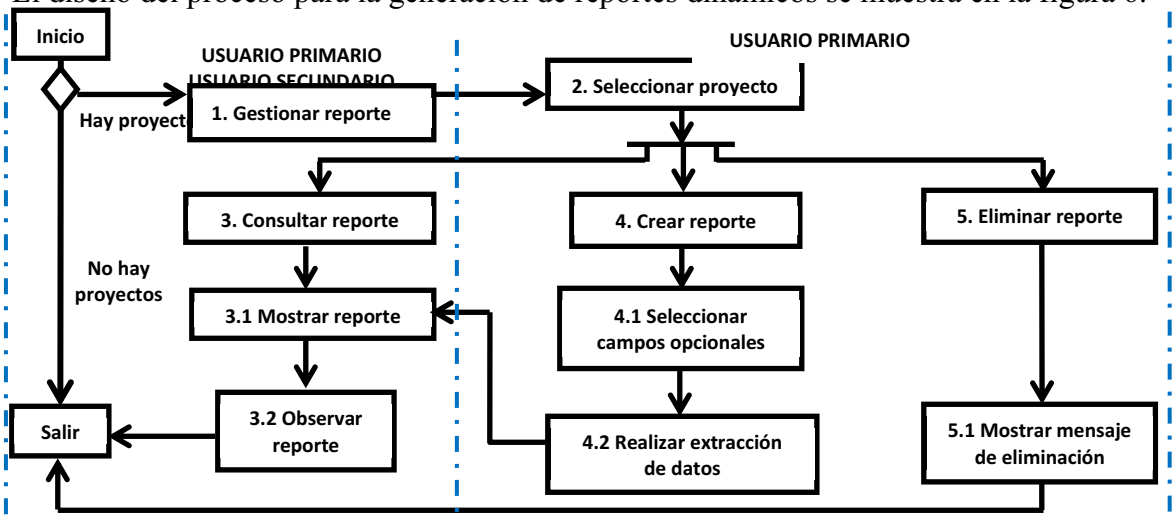


Figura 6. Diagrama de actividad del proceso de generación de reportes dinámicos.

2.3 Módulo de gestión de Reportes Específicos

Permite al usuario seleccionar elementos de ciertos módulos del sistema dentro de la base de datos como se muestra en la figura 7.

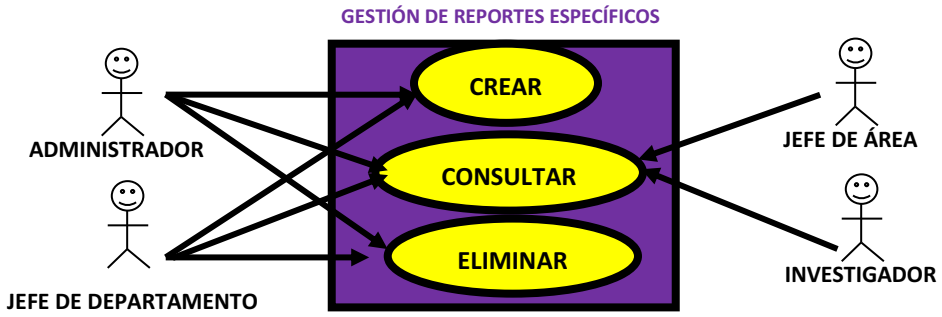


Figura 7. Diagrama de casos de uso de la gestión de reportes específicos.

Diseño del proceso para la generación de reportes específicos se muestra en la figura 8.

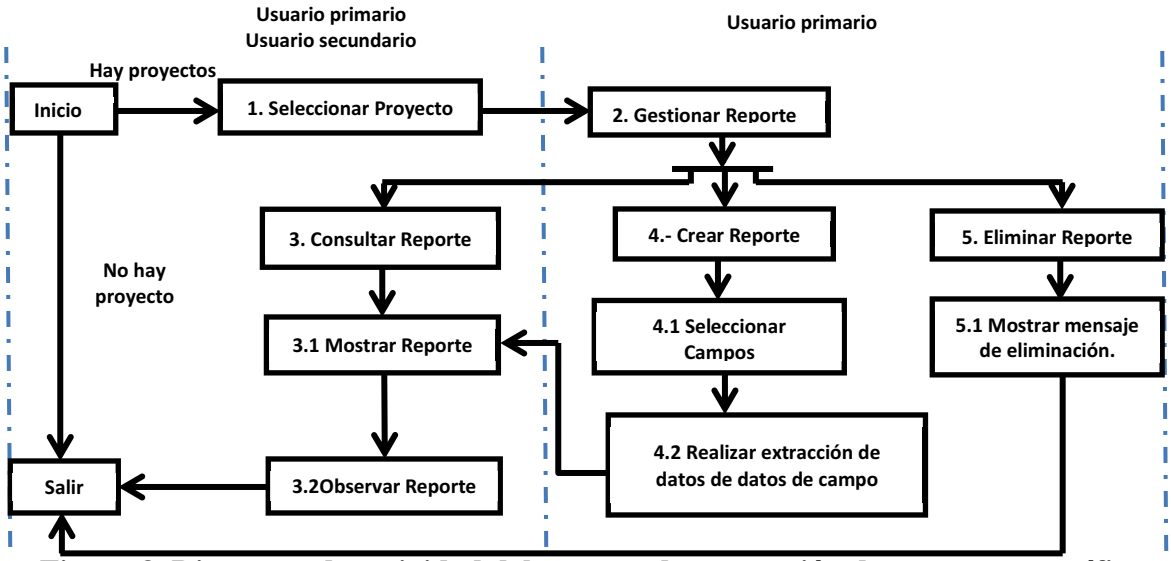


Figura 8. Diagrama de actividad del proceso de generación de reportes específicos.

3. Propuestas de Proyectos Terminales

3.1 Módulo de gestión de Propuestas de Proyectos Terminales

Permite a los profesores realizar la gestión de sus propuestas de proyectos terminales, con la finalidad de que los alumnos tengan acceso a dichas propuestas para los fines que a ellos convenga (medio de información para que los alumnos conozcan las propuestas de proyectos terminales que los profesores sugieren). Como se observa en la figura 9, los profesores podrán realizar altas, bajas, cambios y consultas de sus propuestas para proyectos terminales, mientras que los alumnos solo podrán realizar consultas de las propuestas, estas consultas podrán realizarse de acuerdo al nombre del profesor y al área de investigación, es decir, serán consultas generales, a diferencia de las consultas que realizará el profesor tal y como se muestra en la figura 10.

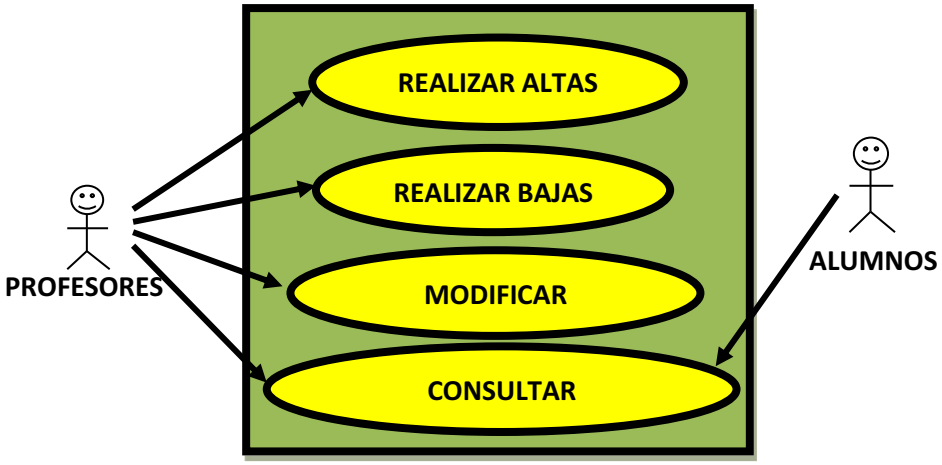


Figura 9. Diagrama de casos de uso de la gestión de propuestas de proyectos terminales registradas por los profesores.

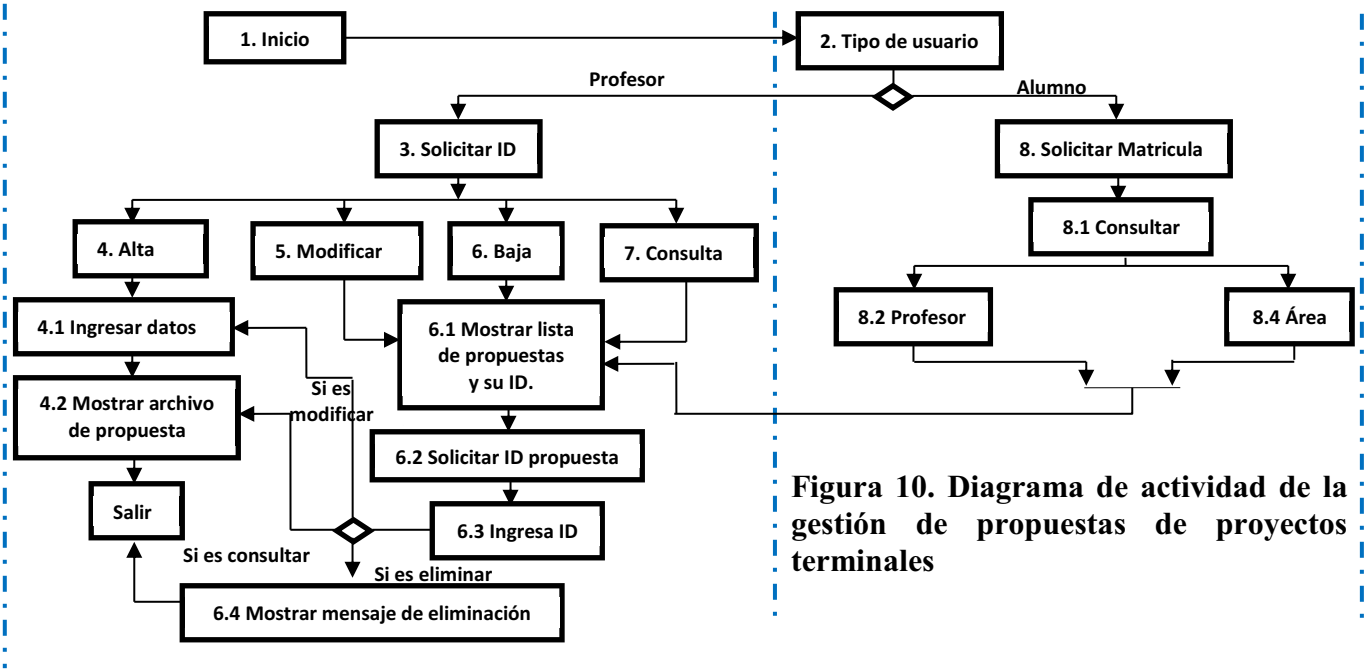


Figura 10. Diagrama de actividad de la gestión de propuestas de proyectos terminales

4. Proyectos Terminales

4.1 Módulo de gestión de Proyectos Terminales

Permite a los alumnos desarrolladores (usuario primario) realizar la gestión de su proyecto terminal finalizado, con la finalidad de que el usuario secundario del sistema (profesores, alumnos) pueda observar información sobre dicho proyecto terminal concluido. Como se observa en la figura 11, este módulo permitirá al alumno(s) desarrollador(es) del proyecto realizar las operaciones de alta y baja de su proyecto terminal concluido, así como consultas generales (podrá consultar los proyectos terminales existentes en la base de datos), mientras que el usuario secundario solo podrá realizar la consulta de dichos proyectos terminales, estas consultas podrán realizarse de acuerdo al departamento y al área de investigación al que pertenece el proyecto terminal tal y como se muestra en la figura 12.

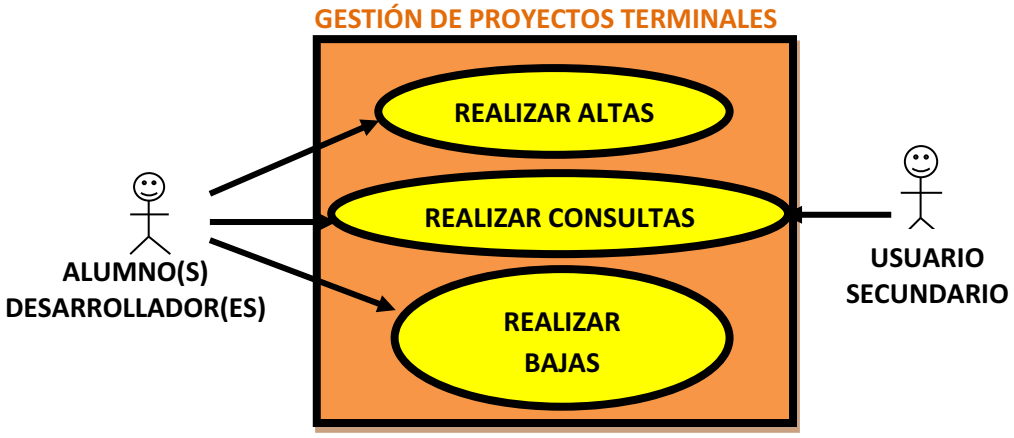


Figura 11. Diagrama de casos de uso de la gestión de proyectos terminales.

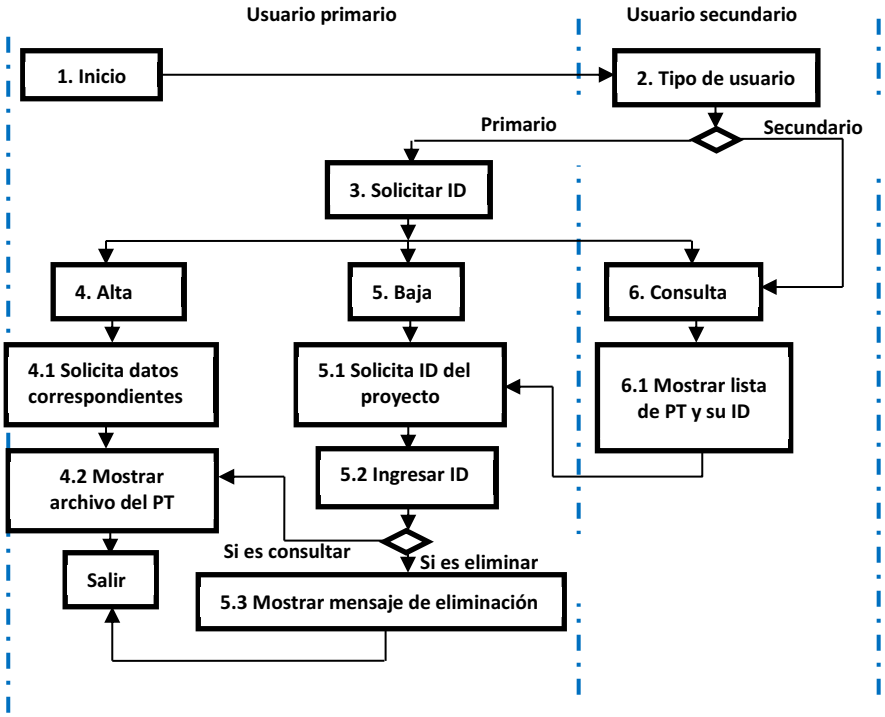


Figura 12. Diagrama de actividad de la gestión de proyectos terminales

Especificación Técnica

Comunicación entre bloques:

1. El módulo de carga de datos recibe como entrada un archivo que contiene la información con los datos que se deben cargar en la base de datos mediante un script que contendrá instrucciones SQL para realizar la carga.

2. El módulo de gestión de reportes programados generará de manera periódica (trimestral y anual) los reportes indicados en la Tabla 2. Recibirá como entrada la información relacionada con un proyecto de investigación después de haber transcurrido las fechas establecidas. Los datos que incluye el reporte son:

LISTA DE REPORTES	DATOS INCLUIDOS EN EL REPORTE
• REPORTE POR DEPTO. • REPORTE POR ÁREA DE INVESTIGACIÓN. • REPORTE POR PROYECTO. • REPORTE POR PROFESOR INVESTIGADOR.	• ID del reporte. • Tipo de reporte (especificará que se trata de un reporte programado). • Fecha de creación del reporte. • Periodo que abarca el reporte. • ID del proyecto. • Nombre del proyecto. • Fecha de creación del proyecto. • Área de investigación del proyecto. • Profesor responsable del proyecto. • Colaboradores. • Vigencia del proyecto. • Descripción del proyecto. • Área de trabajo (profesor titular y colaboradores). • Categoría (profesor titular y colaboradores). • e- Mail (profesor titular y colaboradores). • Cubículo (profesor titular y colaboradores). • Nombre de la unidad. • Nombre de la división. • Nombre del departamento. • Productos del trabajo generados en el periodo.

Tabla 2. Tabla de lista de reportes y campos que los integran.

3. El módulo de gestión de reportes dinámicos estará disponible para que el administrador y el jefe del departamento de Sistemas puedan: realizar la extracción de datos de los distintos campos de información que requieran para su reporte. Los campos de entrada se clasifican en dos grupos: obligatorios y opcionales, como se muestra en la tabla 3.

CAMPOS OBLIGATORIOS	CAMPOS OPCIONALES
• ID del reporte. • Tipo de reporte (especificará que se trata de un reporte dinámico). • Fecha de creación del reporte. • Periodo que abarca el reporte.	• ID del proyecto. • Nombre del proyecto. • Fecha de creación del proyecto. • Área de desarrollo del proyecto. • Profesor titular del proyecto. • Colaboradores. • Vigencia del proyecto. • Descripción del proyecto. • Número económico (profesor titular y colaboradores). • Área de trabajo (profesor titular y colaboradores). • Categoría (profesor titular y colaboradores). • E- Mail (profesor titular y colaboradores). • Cubículo (profesor titular y colaboradores). • Nombre de la unidad. • Nombre de la división. • Nombre del departamento • Productos de trabajo.

Tabla 3. Tabla de campos obligatorios y opcionales que integran un reporte dinámico.

4. El módulo de gestión de reportes específicos estará disponible para que el administrador y el jefe del departamento de Sistemas puedan: realizar extracción de datos enfocándose en ciertos campos en particular, los cuales determinarán la información que integrará al reporte. Estos campos se clasifican en 3 tipos; obligatorios, determinantes y dependientes (ver tabla 4).

CAMPOS OBLIGATORIOS:	
• ID del reporte. • Tipo de reporte (especificará que se trata de un reporte dinámico). • Fecha de creación del reporte. • Periodo que abarca el reporte.	
CAMPOS DETERMINANTES	**CAMPOS DEPENDIENTES**
• Profesor	• Nombre de proyecto (s). • Nombre del congreso (s) al que ha asistido. • Colaboradores que ha tenido. • Área de trabajo. • Categoría. • Teléfono. • Dirección. • E-mail. • Cubículo. • Número económico.
• Temáticas.	• Disciplina. • Tema.

	• Subtema.
• Productos de trabajo.	• Artículo Revista indexada (Nacional e Internacional). • Asistencia a eventos (Congresos Nacionales e Internacionales). • Ponencias (Nacionales e Internacionales) • Proyectos Terminales. • Tesis de maestría. • Capítulo de Libro. • Libro. • Temática.
• División. • Departamento. • Área de trabajo.	• Nombre del profesor (es). • Área de trabajo del profesor (es). • Teléfono. • E-mail. • Cubículo. • Número económico. • Proyectos realizados. • Área de desarrollo del proyecto (s). • Descripción del proyecto (s). • Productos de trabajo. • Descripción del producto.

Tabla 4. Tabla de clasificación de campos para el módulo de gestión de reportes específicos.

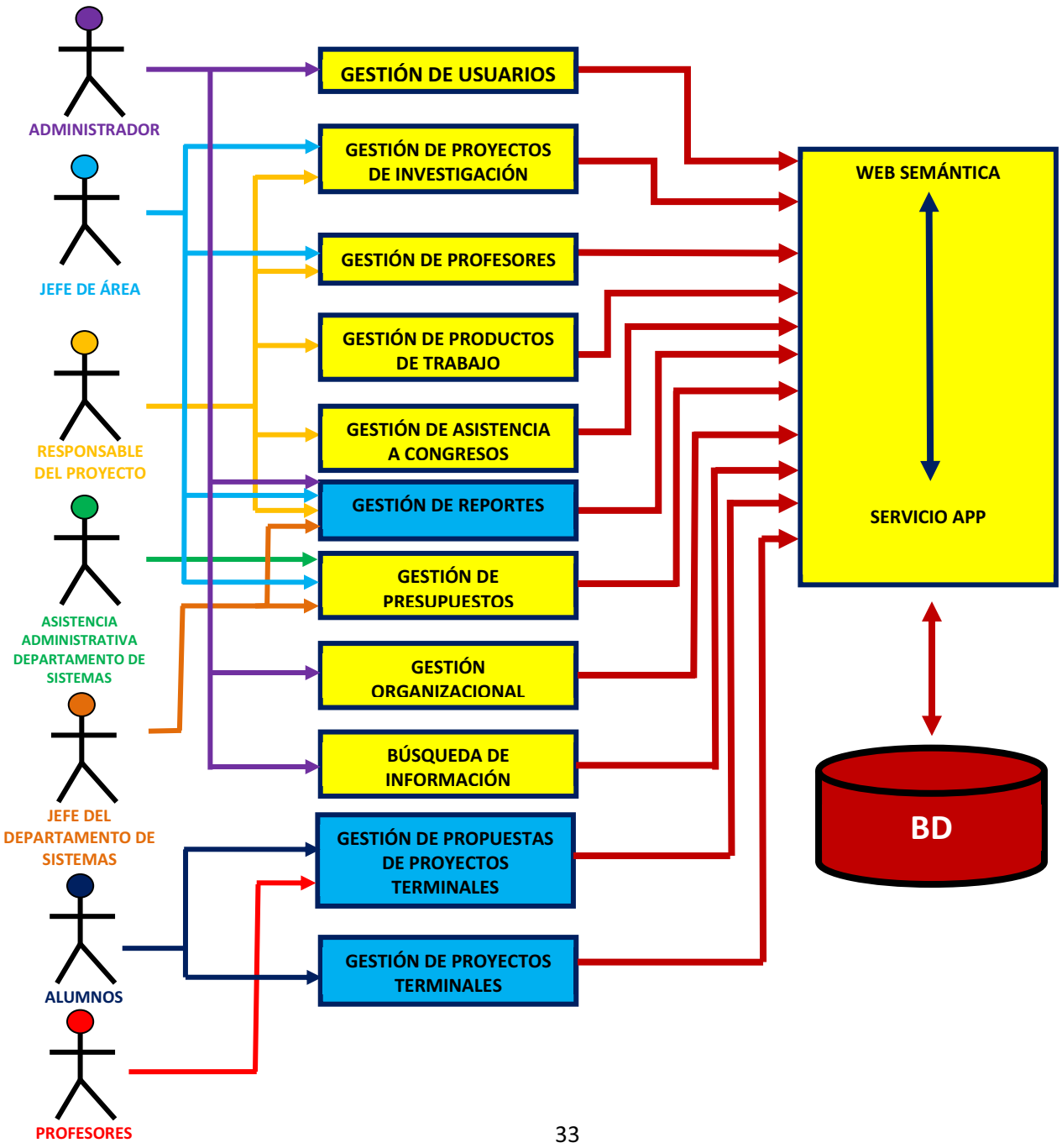
5. El módulo de gestión de propuestas de proyectos terminales estará disponible para que los profesores puedan: registrar sus propuestas con la finalidad de que los alumnos interesados tengan acceso a ellas, los datos de entrada de este módulo son:

- ID profesor.
- Nombre del profesor.
- Departamento.
- Área de trabajo.
- Cubículo.
- E-mail.
- ID Propuesta.
- Nombre de la propuesta
- Información de la propuesta.
- Matrícula del alumno.

6. El módulo de gestión de proyectos terminales estará disponible para que los alumnos con proyecto terminal concluido puedan: publicar dicho proyecto con la finalidad de que alumnos o profesores interesados tengan acceso a la información correspondiente a dicho proyecto, los datos de entrada del módulo son:

- ID del desarrollador del proyecto.
- Nombre de los participantes en el proyecto.
- Matrícula(s) del desarrollador(es).
- ID del proyecto.
- Nombre del proyecto.
- Área de desarrollo.
- Departamento.
- Información general del proyecto.

En la Figura 13 se esquematiza la comunicación entre los bloques del espacio virtual.



CARACTRISTICAS MINIMAS

Se considera que los jefes de departamento, así como el administrador (usuarios primarios), tendrán acceso a todas las operaciones de los distintos tipos de reportes. Por su parte, los profesores investigadores y jefes de área tienen acceso únicamente a la consulta de reportes de los proyectos de investigación.

Con respecto a la gestión de propuestas de proyectos terminales, los profesores podrán realizar las operaciones establecidas para este módulo: altas, bajas, cambios y consultas, sin embargo, las consultas serán única y exclusivamente de sus propuestas, mientras que los alumnos podrán realizar consultas de propuestas de cualquier profesor. Se considera que los alumnos desarrolladores de proyectos terminales podrán realizar todas las operaciones con respecto a la gestión de sus proyectos, por su parte, los usuarios secundarios (profesores y alumnos) solo podrán realizar consultas de dichos proyectos.

El proyecto se dará por concluido una vez que el usuario pueda realizar las operaciones de creación, consulta y eliminación de reportes de proyectos de investigación según sea el caso, así como la satisfactoria publicación de los proyectos terminales concluidos por parte de los alumnos y el registro de las propuestas de proyectos terminales sugeridas por los profesores en el observatorio Tecno - Educativo.

Esquema Físico

Unidad

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`unidad`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`unidad` (
  `idunidad` INT NOT NULL AUTO_INCREMENT ,
  `unidad` VARCHAR(45) NULL ,
  `rector` VARCHAR(45) NULL ,
  `secretario_unidad` VARCHAR(45) NULL ,
  `mail_unidad` VARCHAR(45) NULL ,
  `telefono_unidad` VARCHAR(45) NULL ,
  PRIMARY KEY (`idunidad`) )
ENGINE = InnoDB;
```

Division

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`division`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`division` (
  `iddivision` INT NOT NULL AUTO_INCREMENT ,
  `division` VARCHAR(45) NULL ,
  `unidad` INT NULL ,
  `director` VARCHAR(45) NULL ,
  `secretario` VARCHAR(45) NULL ,
  `mail` VARCHAR(45) NULL ,
  `telefono` VARCHAR(45) NULL ,
  PRIMARY KEY (`iddivision`) ,
  INDEX `unidad_idx` (`unidad` ASC) ,
  CONSTRAINT `unidad`
    FOREIGN KEY (`unidad` )
```

```
   REFERENCES `decicionProyectos`.`unidad` (`idunidad` )
   ON DELETE NO ACTION
   ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Departamento

```
-- -----------------------------------------------------------------------------------
-- Table `decicionProyectos`.`dapartamento`
-- -----------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`dapartamento` (
  `iddapartamento` INT NOT NULL AUTO_INCREMENT ,
  `dapartamento` VARCHAR(45) NULL ,
  `division` INT NULL ,
  PRIMARY KEY (`iddapartamento`) ,
  INDEX `division_idx` (`division` ASC) ,
  CONSTRAINT `division`
    FOREIGN KEY (`division` )
    REFERENCES `decicionProyectos`.`division` (`iddivision` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Rol

```
-- -----------------------------------------------------------------------------------
-- Table `decicionProyectos`.`rol`
-- -----------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`rol` (
  `Clave_Rol` INT NOT NULL AUTO_INCREMENT ,
  `Permiso` INT NULL ,
  `Nombre` VARCHAR(45) NULL ,
  PRIMARY KEY (`Clave_Rol`) )
ENGINE = InnoDB;
```

Usuario

```
-- -----------------------------------------------------------------------------------
-- Table `decicionProyectos`.`usuario`
-- -----------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`usuario` (
  `Clave_usuario` INT NOT NULL AUTO_INCREMENT ,
  `Rol_Clave_Rol` INT NULL ,
  `nombre` VARCHAR(45) NULL ,
  `usuario` VARCHAR(45) NULL ,
  `contrasenia` VARCHAR(45) NULL ,
  `telefono` VARCHAR(45) NULL ,
  `correo` VARCHAR(45) NULL ,
  PRIMARY KEY (`Clave_usuario`) ,
  INDEX `rolFK_idx` (`Rol_Clave_Rol` ASC) ,
  CONSTRAINT `rolFK`
    FOREIGN KEY (`Rol_Clave_Rol` )
    REFERENCES `decicionProyectos`.`rol` (`Clave_Rol` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Área de desarrollo

```
-- -----------------------------------------------------------------------------------------
-- Table `decicionProyectos`.`area_desarrollo`
-- -----------------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`area_desarrollo` (
  `idarea_desarrollo` INT NOT NULL AUTO_INCREMENT ,
  `area` VARCHAR(45) NULL ,
  `descripcion` VARCHAR(45) NULL ,
  `objetivos` VARCHAR(45) NULL ,
  `jefe` VARCHAR(45) NULL ,
  `mail` VARCHAR(45) NULL ,
  `telefono` VARCHAR(45) NULL ,
  PRIMARY KEY (`idarea_desarrollo`) )
ENGINE = InnoDB;
```

Proyecto Terminal

```
-- -----------------------------------------------------------------------------------------
-- Table `decicionProyectos`.`proyecto_terminal`
-- -----------------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`proyecto_terminal` (
  `idproyecto_terminal` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(255) NULL ,
  `urlarchivo` VARCHAR(255) NULL ,
  `profesor` INT NULL ,
  `alumnos` VARCHAR(255) NULL ,
  `estado` boolean NULL ,
  PRIMARY KEY (`idproyecto_terminal`) ,
  INDEX `profe_idx` (`profesor` ASC) ,
  CONSTRAINT `profesor`
    FOREIGN KEY (`profesor` )
    REFERENCES `decicionProyectos`.`profesor` (`idprofesor` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Alumno

```
-- -----------------------------------------------------------------------------------------
-- Table `decicionProyectos`.`alumno`
-- -----------------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`alumno` (
  `idalumno` INT NOT NULL AUTO_INCREMENT ,
  `usuario` INT NULL ,
  `id_pt` INT NULL ,
  `nombre` VARCHAR(45) NULL ,
  `matricula` VARCHAR(45) NULL ,
  `estadopt` boolean NULL ,
  PRIMARY KEY (`idalumno`)
  )
ENGINE = InnoDB;
```

Propuesta pt

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`propuestapt`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`propuestapt` (
  `idpropuestapt` INT NOT NULL AUTO_INCREMENT ,
  `descripcion` VARCHAR(255) NULL ,
  `nombre` VARCHAR(255) NULL ,
  `urlarchivo` VARCHAR(255) NULL ,
  `usuario` INT NULL ,
  `estado` boolean NULL ,
  PRIMARY KEY (`idpropuestapt`) ,
  INDEX `profe_idx` (`usuario` ASC) ,
  CONSTRAINT `usuario`
    FOREIGN KEY (`usuario` )
    REFERENCES `decicionProyectos`.`usuario` (`Clave_usuario` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Proyecto

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`proyecto`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`proyecto` (
  `idproyecto` INT NOT NULL AUTO_INCREMENT ,
  `proyecto` VARCHAR(45) NULL ,
  `fechacreacion` DATE NULL,
  `fechai` DATE NULL,
  `fechaf` DATE NULL,
  `descripcion` VARCHAR(255) NULL ,
   `area_de_desarrollo` INT NULL ,
   `profesor` INT NULL ,
   `departamento` INT NULL,

  PRIMARY KEY (`idproyecto` ),
  INDEX `areaaFK_idx` (`area_de_desarrollo` ASC) ,
  INDEX `profe_idx` (`profesor` ASC) ,
  INDEX `deptooFK_idx` (`departamento` ASC) ,
  CONSTRAINT `areaaFK2`
    FOREIGN KEY (`area_de_desarrollo` )
    REFERENCES `decicionProyectos`.`area_desarrollo` (`idarea_desarrollo` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `profesor2`
    FOREIGN KEY (`profesor` )
    REFERENCES `decicionProyectos`.`profesor` (`idprofesor` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
   CONSTRAINT `deptooFK2`
    FOREIGN KEY (`departamento` )
    REFERENCES `decicionProyectos`.`dapartamento` (`iddapartamento` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Proyecto colaboradores

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`proyectocolaborador`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`proyectocolaborador` (
  `idproyectoc` INT NOT NULL AUTO_INCREMENT ,
  `proyecto` INT NULL ,
  `profesor` INT NULL ,
  PRIMARY KEY (`idproyectoc`),
  INDEX `proy_idx` (`proyecto` ASC) ,
  INDEX `profe_idx` (`profesor` ASC) ,
CONSTRAINT `proyecto3`
    FOREIGN KEY (`proyecto` )
    REFERENCES `decicionProyectos`.`proyecto` (`idproyecto` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `profesor3`
    FOREIGN KEY (`profesor` )
    REFERENCES `decicionProyectos`.`profesor` (`idprofesor` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Informe dinamico

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`informedinamico`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`informedinamico` (
  `idinformedinamico` INT NOT NULL AUTO_INCREMENT ,
  `nombre` VARCHAR(255) NULL ,
  `fechainicio` DATE NULL ,
  `fechafin` DATE NULL ,
  `proyecto` INT NULL ,
  `tipo` VARCHAR(255) NULL ,
  `urlpdf` VARCHAR(255) NULL ,
  `estado` boolean ,
  PRIMARY KEY (`idinformedinamico` ),
   INDEX `proy_idx` (`proyecto` ASC),
   CONSTRAINT `proyecto4`
    FOREIGN KEY (`proyecto` )
    REFERENCES `decicionProyectos`.`proyecto` (`idproyecto` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Informe programado

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`informeProgramado`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`informeProgramado` (
  `idinformeProgramado` INT NOT NULL AUTO_INCREMENT ,
  `nombre` VARCHAR(255) NULL ,
  `fechainicio` DATE NULL ,
  `periodo` VARCHAR(255) NULL ,
  `tipo` VARCHAR(255) NULL ,
  `urlpdf` VARCHAR(255) NULL ,
```

```
  `estado` boolean ,
  PRIMARY KEY (`idinformeProgramado` ))
ENGINE = InnoDB;
```

Temática

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`tematica`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`tematica` (
  `idtematica` INT NOT NULL AUTO_INCREMENT ,
   `profesor` INT NULL ,
  `nombre` VARCHAR(255) NULL ,
   `tema` VARCHAR(255) NULL ,
   `subtema` VARCHAR(255) NULL ,
    `Diciplina` VARCHAR(255) NULL ,
  PRIMARY KEY (`idtematica` ),
  CONSTRAINT `profesor6`
    FOREIGN KEY (`profesor` )
    REFERENCES `decicionProyectos`.`profesor` (`idprofesor` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Producto trabajo

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`productotrabajo`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`productotrabajo` (
  `idproductotrabajo` INT NOT NULL AUTO_INCREMENT ,
   `profesor` INT NULL ,
  `nombre` VARCHAR(255) NULL ,
   `descripcion` VARCHAR(255) NULL ,
   `prealizados` VARCHAR(255) NULL ,
   `objetivo` VARCHAR(255) NULL ,
   `adesarrollo` VARCHAR(255) NULL ,
  PRIMARY KEY (`idproductotrabajo` ),
  CONSTRAINT `profesor5`
    FOREIGN KEY (`profesor` )
    REFERENCES `decicionProyectos`.`profesor` (`idprofesor` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Productos trabajo

```
-- -----------------------------------------------------------------------------
-- Table `decicionProyectos`.`productostrabajo`
-- -----------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`productostrabajo` (
  `idproductostrabajo` INT NOT NULL AUTO_INCREMENT ,
   `profesor` INT NULL ,
  `nombre` VARCHAR(255) NULL ,
   `arevistaindexada` VARCHAR(255) NULL ,
   `asistenciaeventos` VARCHAR(255) NULL ,
   `capitulodellibro` VARCHAR(255) NULL ,
   `proyectoterminales` VARCHAR(255) NULL ,
   `tesismaestria` VARCHAR(255) NULL ,
   `ponencia` VARCHAR(255) NULL ,
```

```
  `libro` VARCHAR(255) NULL ,
  `tematica` VARCHAR(255) NULL ,
 PRIMARY KEY (`idproductostrabajo` ),
 CONSTRAINT `profesor4`
  FOREIGN KEY (`profesor` )
  REFERENCES `decicionProyectos`.`profesor` (`idprofesor` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Informe especifico

```
-- -----------------------------------------------------------------------------------
-- Table `decicionProyectos`.`informeespecifico`
-- -----------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`informeespecifico` (
 `idinformeespecifico` INT NOT NULL AUTO_INCREMENT ,
 `nombre` VARCHAR(255) NULL ,
 `fechaCreacion` DATE NULL ,
 `fechainicio` DATE NULL ,
 `fechafin` DATE NULL ,
 `tipo` VARCHAR(255) NULL ,
 `urlpdf` VARCHAR(255) NULL ,
 `estado` boolean ,
 PRIMARY KEY (`idinformeespecifico`) )
ENGINE = InnoDB;
```

Profesor

```
-- -----------------------------------------------------------------------------------
-- Table `decicionProyectos`.`profesor`
-- -----------------------------------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `decicionProyectos`.`profesor` (
 `idprofesor` INT NOT NULL AUTO_INCREMENT ,
 `usuario` INT NULL ,
 `nombre` VARCHAR(45) NULL ,
 `cubiculo` VARCHAR(45) NULL,
  `neconomico` VARCHAR(45) NULL,
  `categoria` VARCHAR(45) NULL,
 `area_de_desarrollo` INT NULL ,
 `departamento` INT NULL ,
 PRIMARY KEY (`idprofesor`),
 INDEX `areaaFK_idx` (`area_de_desarrollo` ASC) ,
 INDEX `deptooFK_idx` (`departamento` ASC) ,
 CONSTRAINT `areaaFK`
  FOREIGN KEY (`area_de_desarrollo` )
  REFERENCES `decicionProyectos`.`area_desarrollo` (`idarea_desarrollo` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `deptooFK`
  FOREIGN KEY (`departamento` )
  REFERENCES `decicionProyectos`.`dapartamento` (`iddapartamento` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
 )
ENGINE = InnoDB;
```

Manual de Administración del sitio.

Semantic Web Builder es una plataforma para el desarrollo de aplicaciones y portales semánticos, donde el contenido del sistema se le define como semántica, de manera que puede ser interpretada y procesada por personas y/o sistemas, permitiendo el intercambio e integración de información entre diferentes organizaciones.

Requisitos de Instalación

JDK 1.6 o superior – compilación de programas en java.
Application Server (Tomcat, OAS, etc.).
Base de datos con soporte de RDF.
MySQL Server 5.0
WampServer
Phpmyadmin
Eclipse IDE EE Java

Bases de Datos Soportadas

Oracle 10g Including OracleXE.
Microsoft SQL Server 2005 Including MS SQL Express.
DB2 9 Including DB2 9 Express.
PostgreSQL v8.2.
MySQL v5.0.22.
Apache Derby v10.2.2.0.

Instalación

Paso 1

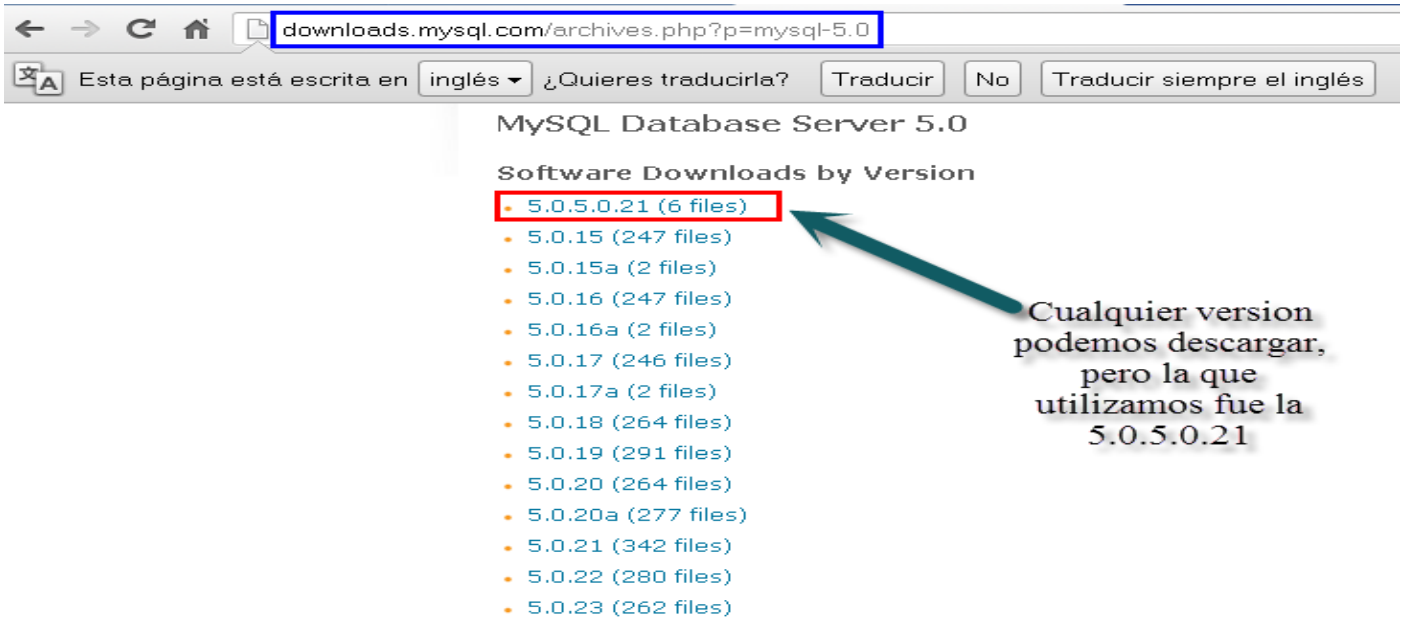
Ubicar la carpeta donde se encuentra instalado el JDK de Java.
Paso 2

Crear una nueva variable de entorno en opciones avanzadas de propiedades del sistema.
Paso 3

Dar Clic en nueva, nombre es JAVA_HOME y valor es la ruta donde está el JDK.
Paso 4

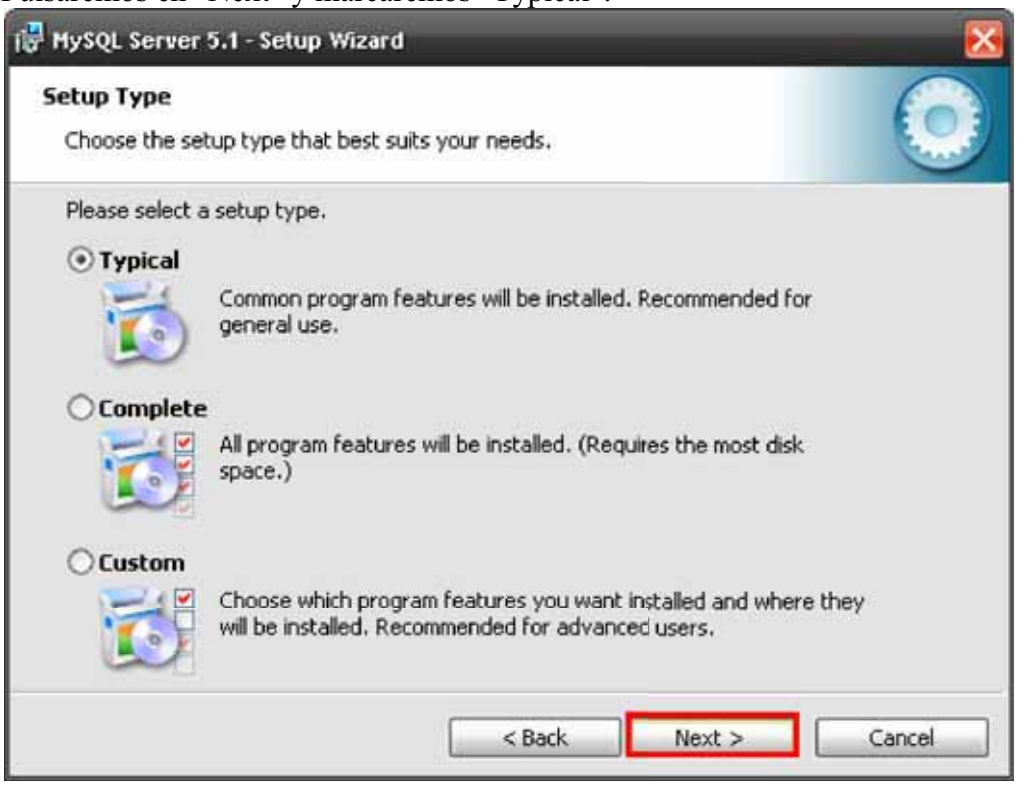
Dar doble Clic en la variable path y agregar en valor ";" y ruta de JDK hasta la carpeta de bin.
Paso 5

Descargar e instalar MySql Server 5.0 en la página oficial http://downloads.mysql.com/archives.php?p=mysql-5.0



Una vez descargado el programa de instalación de MySQL lo ejecutaremos y seguiremos las instrucciones que nos muestra el asistente de instalación:

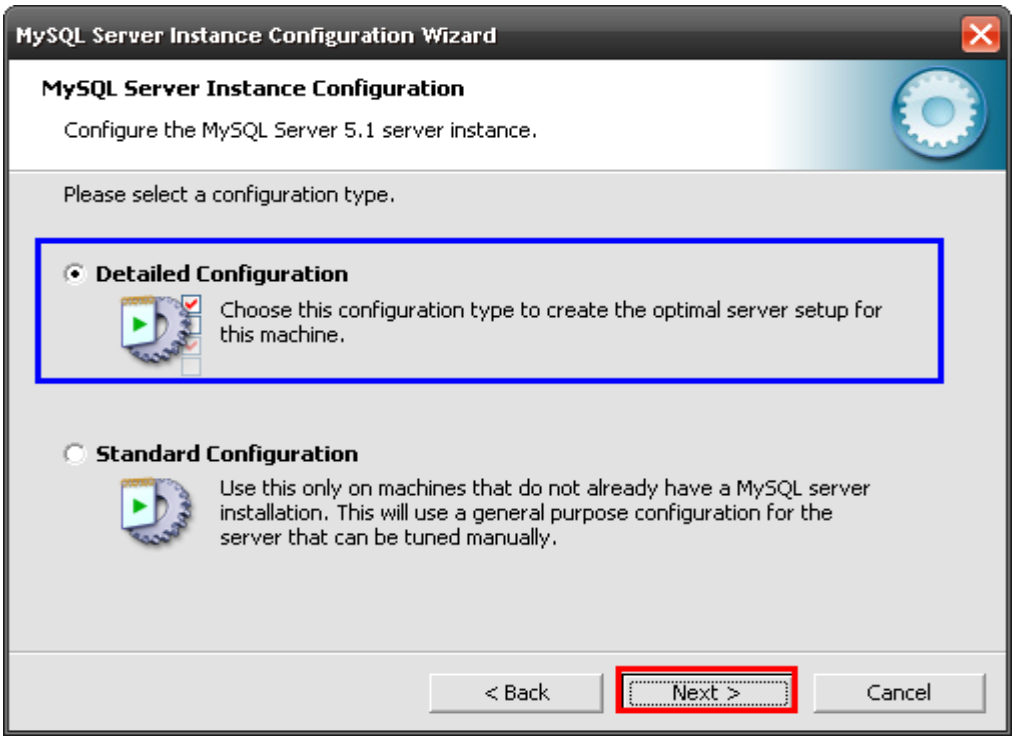
Pulsaremos en "Next" y marcaremos "Typical":



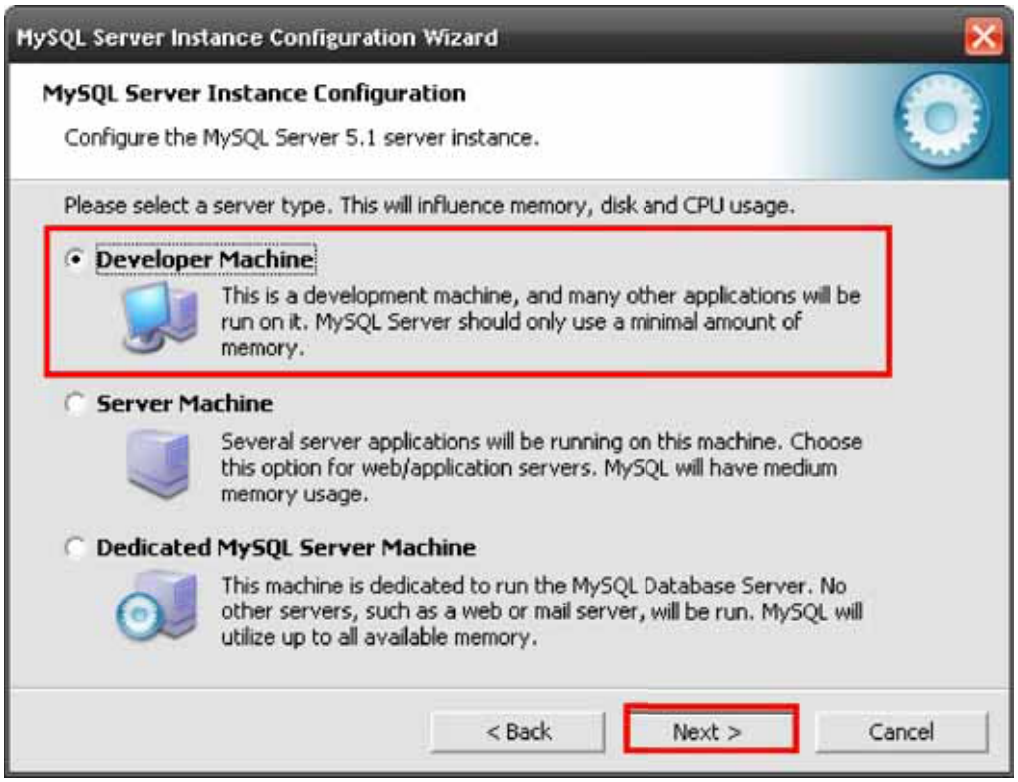
E instalamos:

Una vez finalizado le damos que queremos configurar MySQL server ahora y le damos Finish

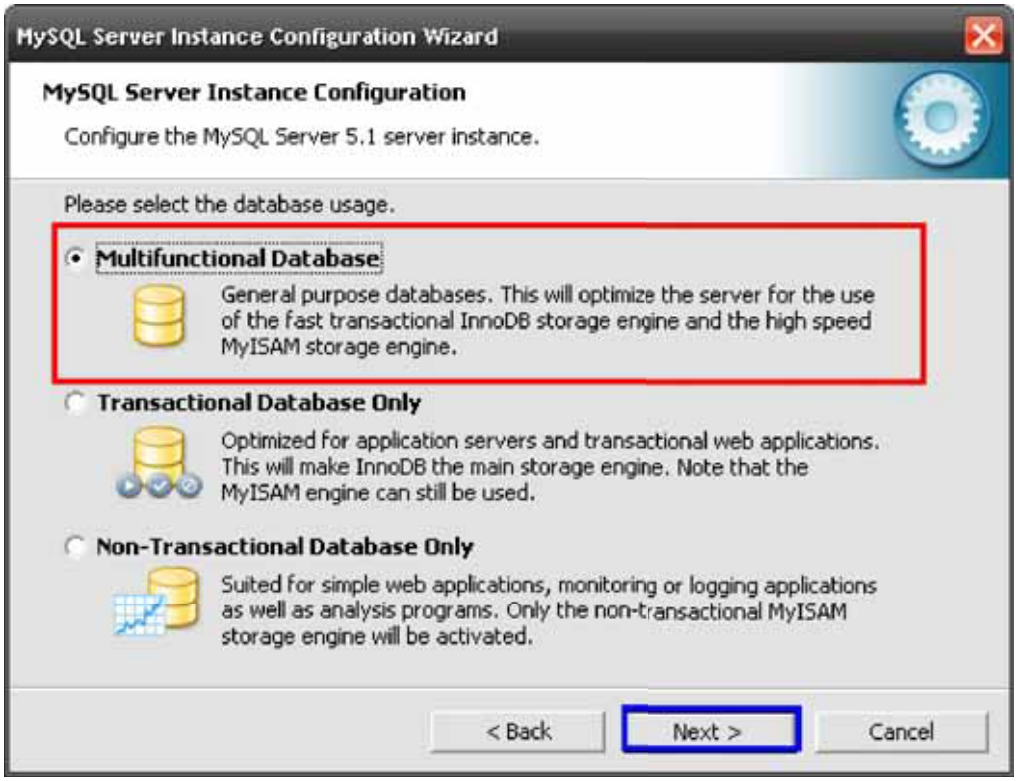
Le damos Detailed Configuration, Next:



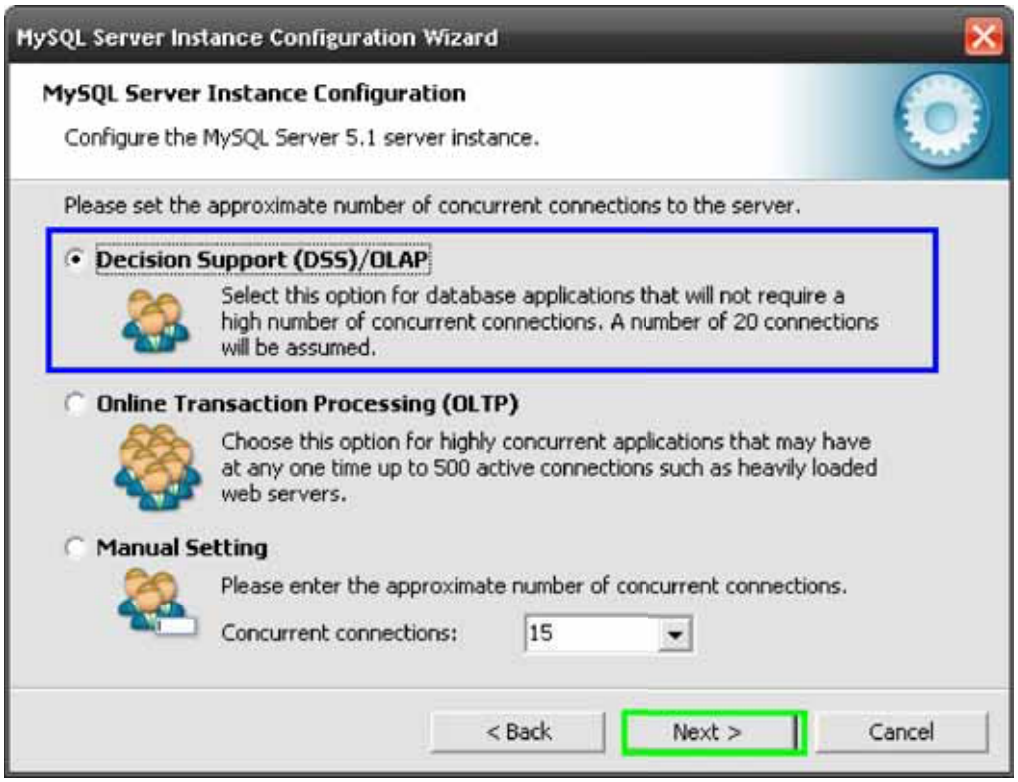
Presionamos Developer Machine y después Next:



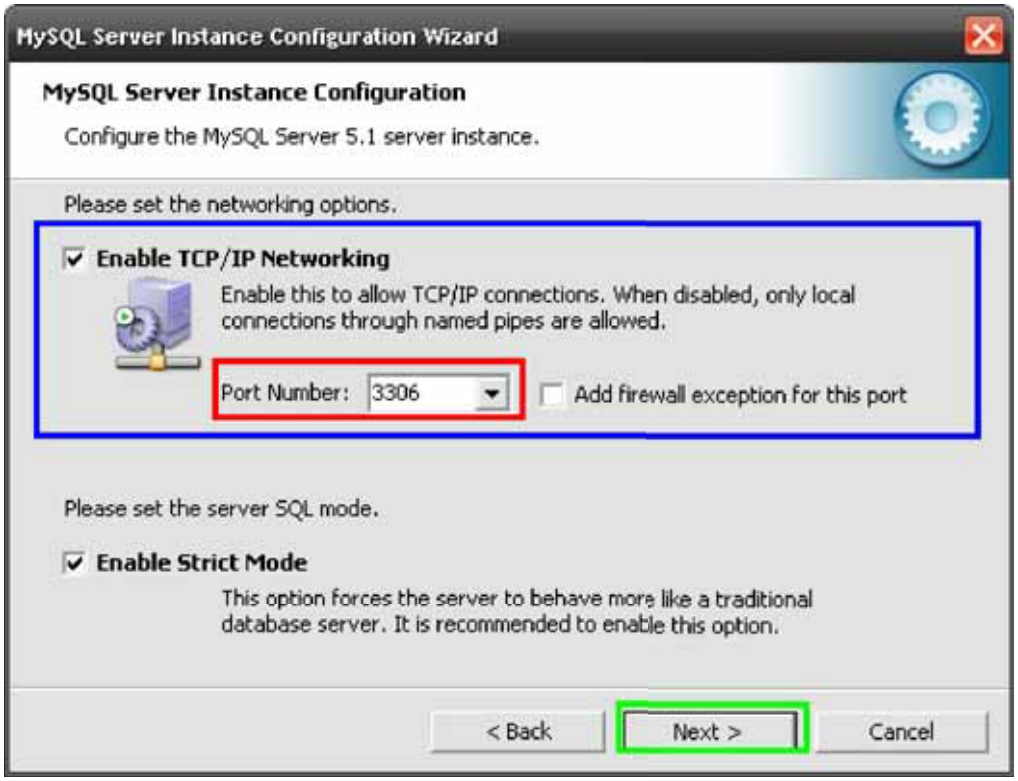
Elegimos la primer opción "Multifunctional Database" y le damos Next



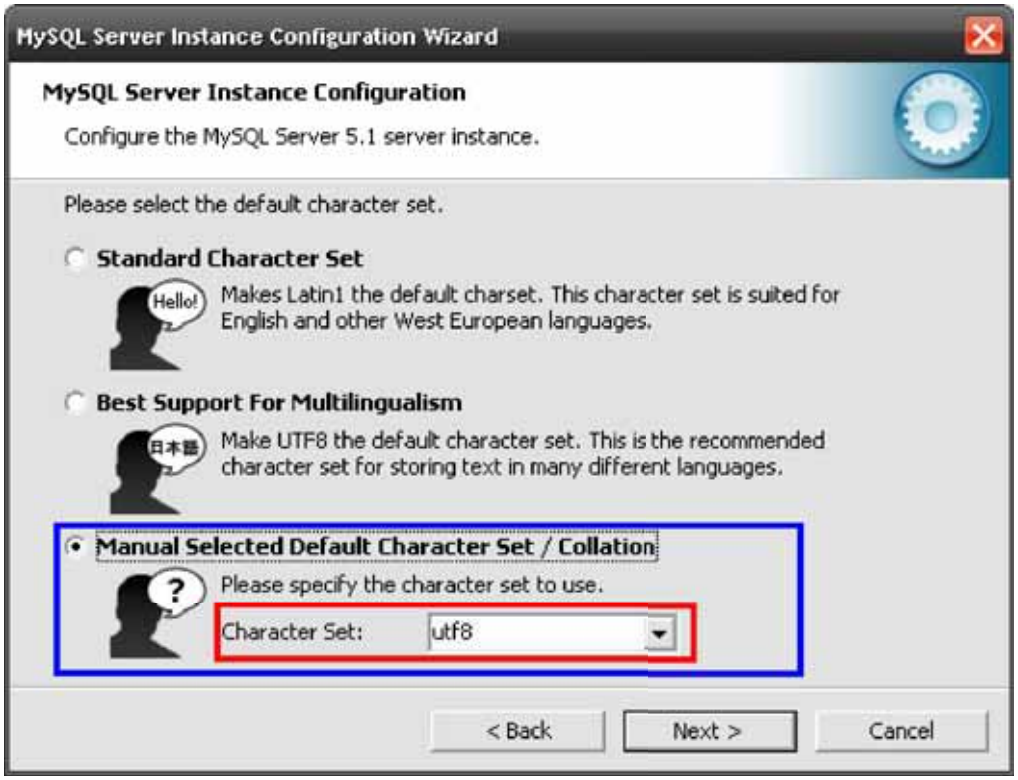
Le damos Next:



Aquí dejamos el puerto de por default 3306 y le damos Next



Aquí es importante elegir el Charecter Set por utf8 para que funciona correctamente nuestra Base de Datos.



Y le damos Next

Ponemos un password y le damos Next

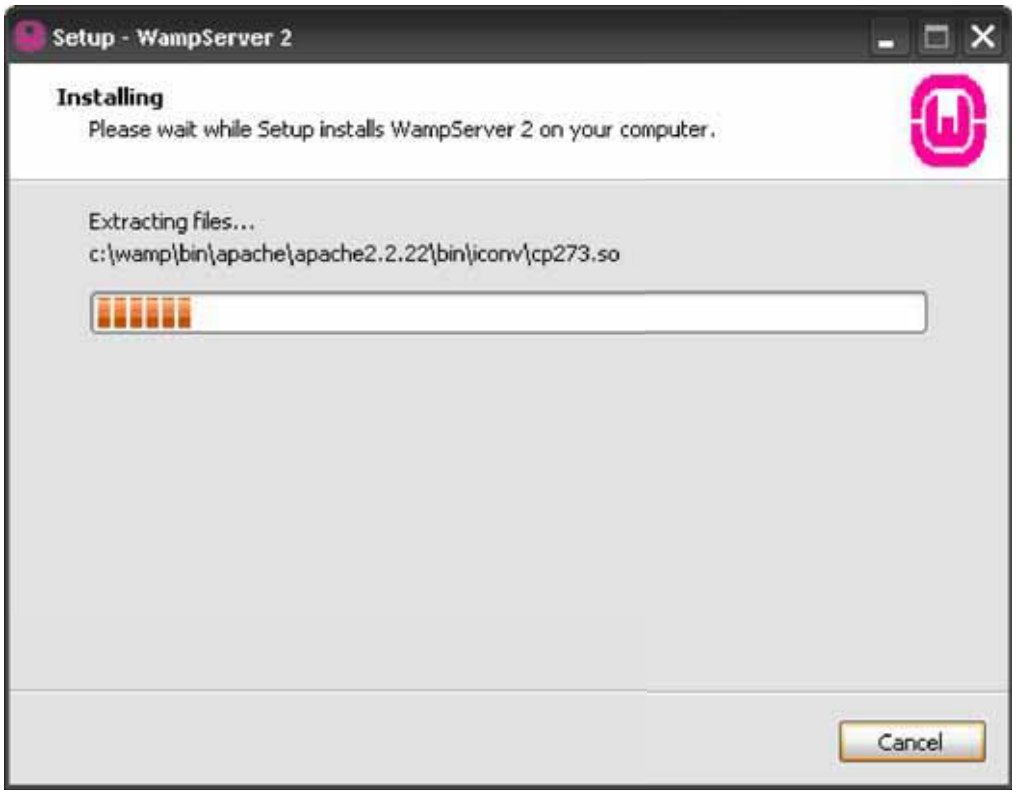
Y listo se instaló MySQL Server correctamente!!!!!

Paso 6

Descargamos un administrador de base de datos para ser más intuitivo y de fácil acceso a nuestra base de datos. En este caso instalaremos “PhpmyAdmin” en el cual podemos administrar fácilmente nuestra base de datos. Para eso bajaremos un programa llamado “wampserver” en el cual incluye “PhpmyAdmin”

Bajaremos “wampserver” de la siguiente página:
http://sourceforge.net/projects/wampserver/

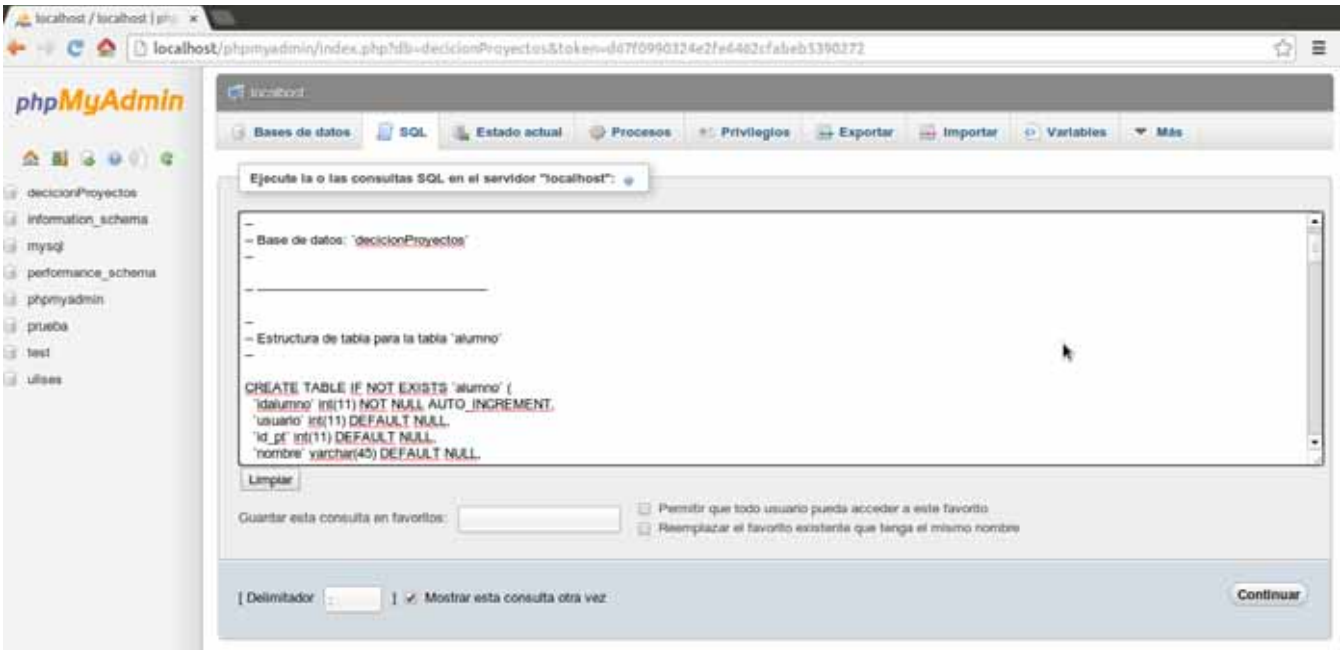
Y lo instalamos



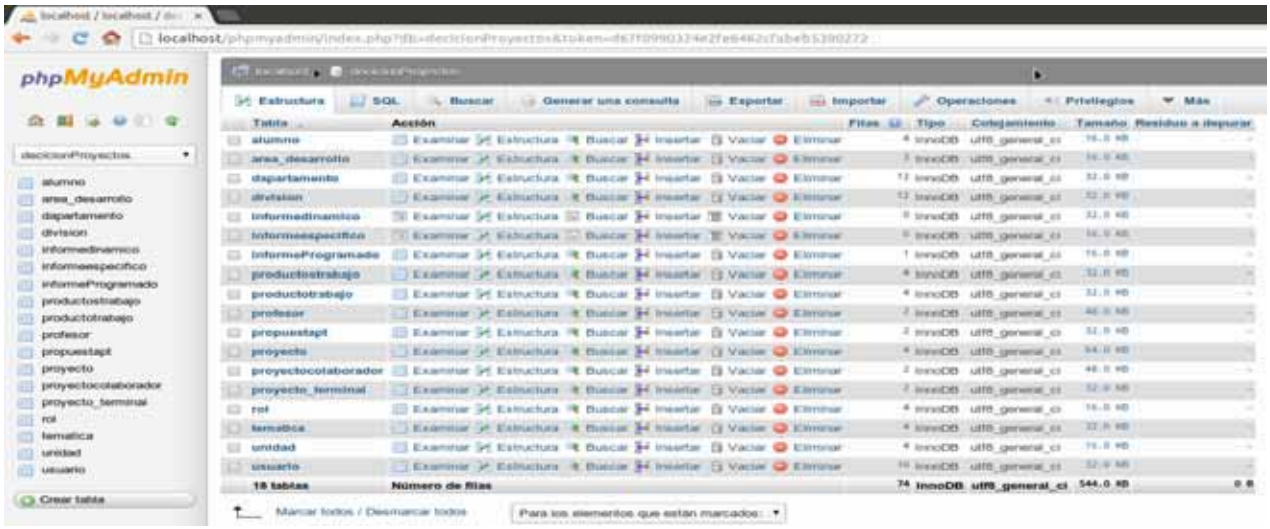
Y listo se instaló correctamente.

Ingresamos a nuestro navegador preferido y le damos esta ruta para que nos abra "phpmyadmin" http://localhost/phpmyadmin/ . Ingresamos el usuario y contraseña y le damos Continuar

Y nos dirigimos al apartado SQL en donde cargaremos nuestro Script previamente elaborado para poder cargar las tablas de nuestro proyecto. Una vez pegado el código SQL le damos Continuar.



Y si todo sale bien nuestra base de datos se generara y nos mostrara todas las tablas y campos que hay en ella.



Paso 7

Una vez que tengamos nuestra Base de Datos y nuestro herramienta para administrar y manipular la BD, proseguiremos a instalar las herramientas para cargar nuestro proyecto.

Nos dirigimos a la página de Tomcat apache http://tomcat.apache.org/download-70.cgi y bajamos el "apache-tomcat-7.0.42"

7.0.42

Please see the README file for packaging information. It explains what every distribution contains.

Binary Distributions

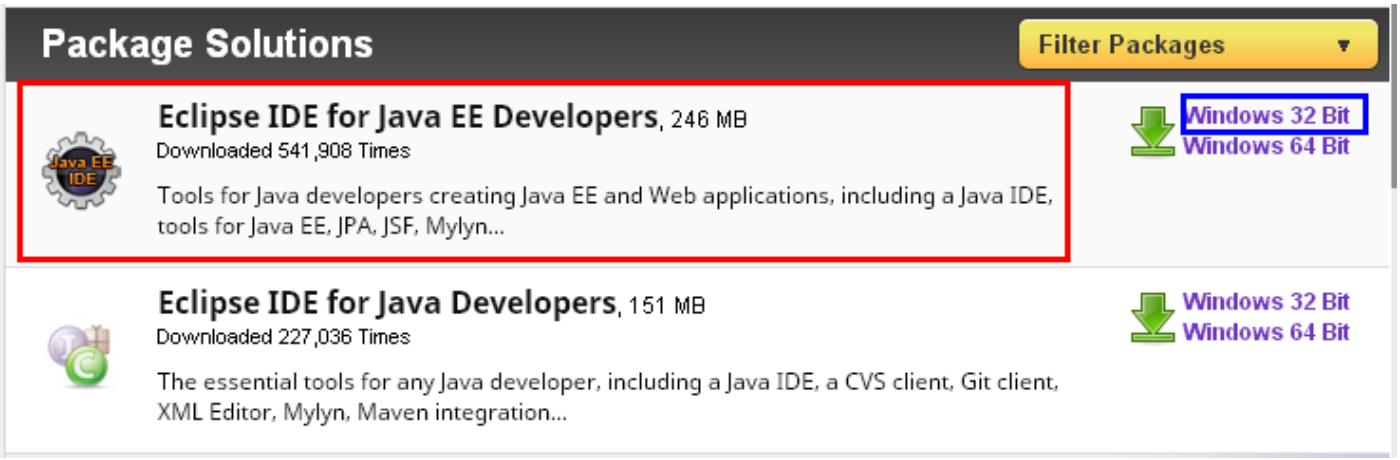
- Core:
  - zip (pgp, md5)
  - tar.gz (pgp, md5)
  - 32-bit Windows zip (pgp, md5)
  - 64-bit Windows zip (pgp, md5)
  - 64-bit Itanium Windows zip (pgp, md5)
  - 32-bit/64-bit Windows Service Installer (pgp, md5)

Cualquier versión que sea superior a 7.x.x nos sirve para poder correr nuestro proyecto.

Descomprimimos el archivo y listo, posteriormente lo utilizaremos en el siguiente paso.

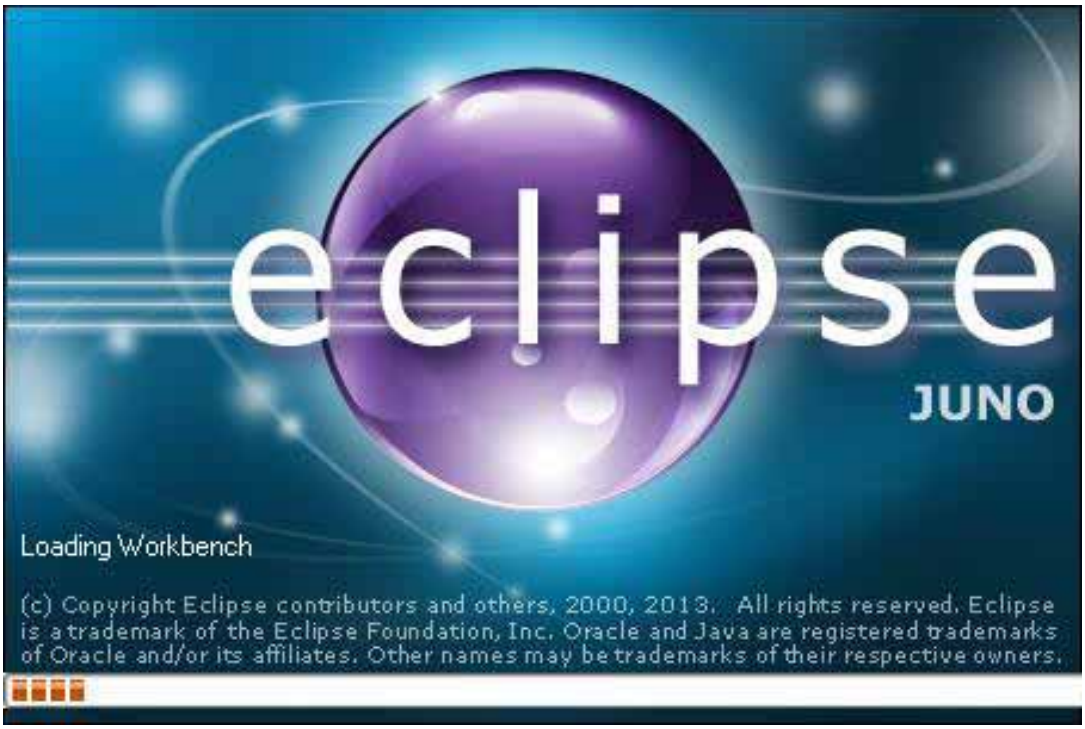
Paso 8

Descargaremos el "Eclipse IDE" de la siguiente página http://www.eclipse.org/downloads/ para poder correr nuestro proyecto.

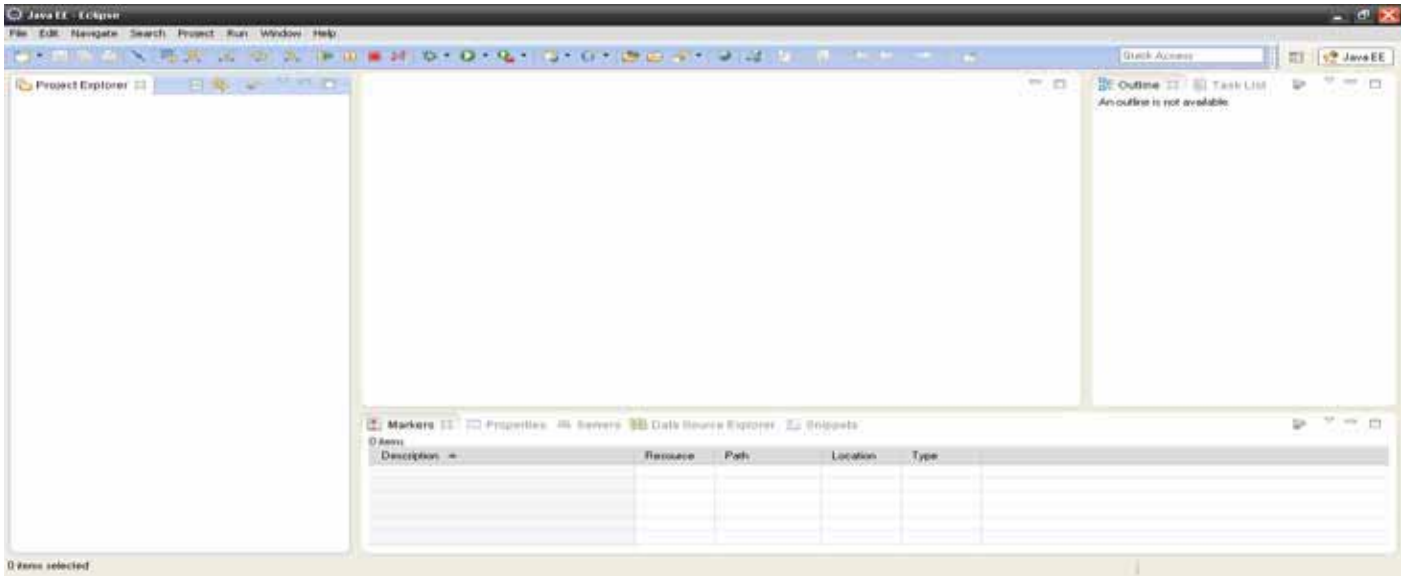


Descomprimimos el archivo y ejecutamos el icono de eclipse

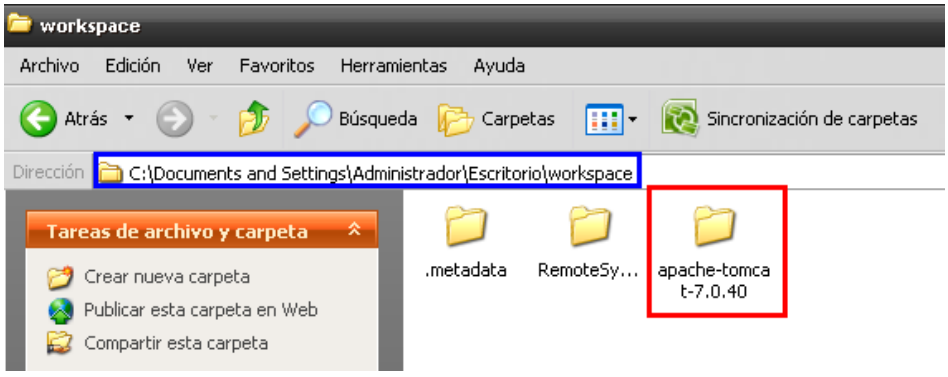
Nos abrirá el eclipse y le especificamos la ruta en donde quiere que guarde nuestros proyectos futuros.



Nos abrirá la IDE de eclipse y nos mostrara algo así:

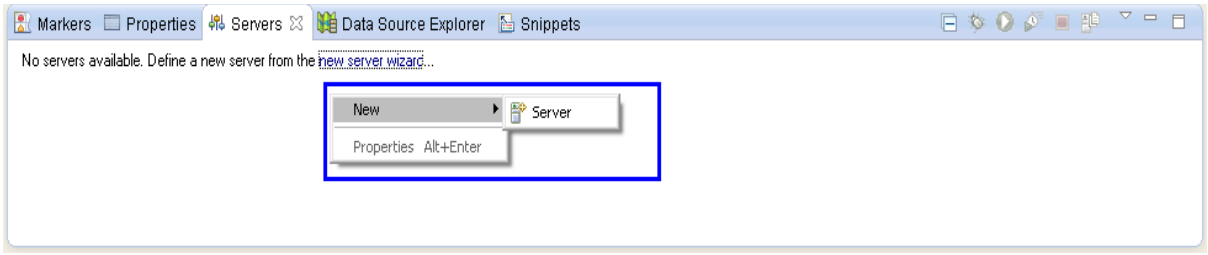


Antes que cualquier cosa nos dirigimos a la carpeta del apache Tomcat que descargamos y descomprimimos, la copiamos y la pegamos dentro de la carpeta workspace del eclipse:

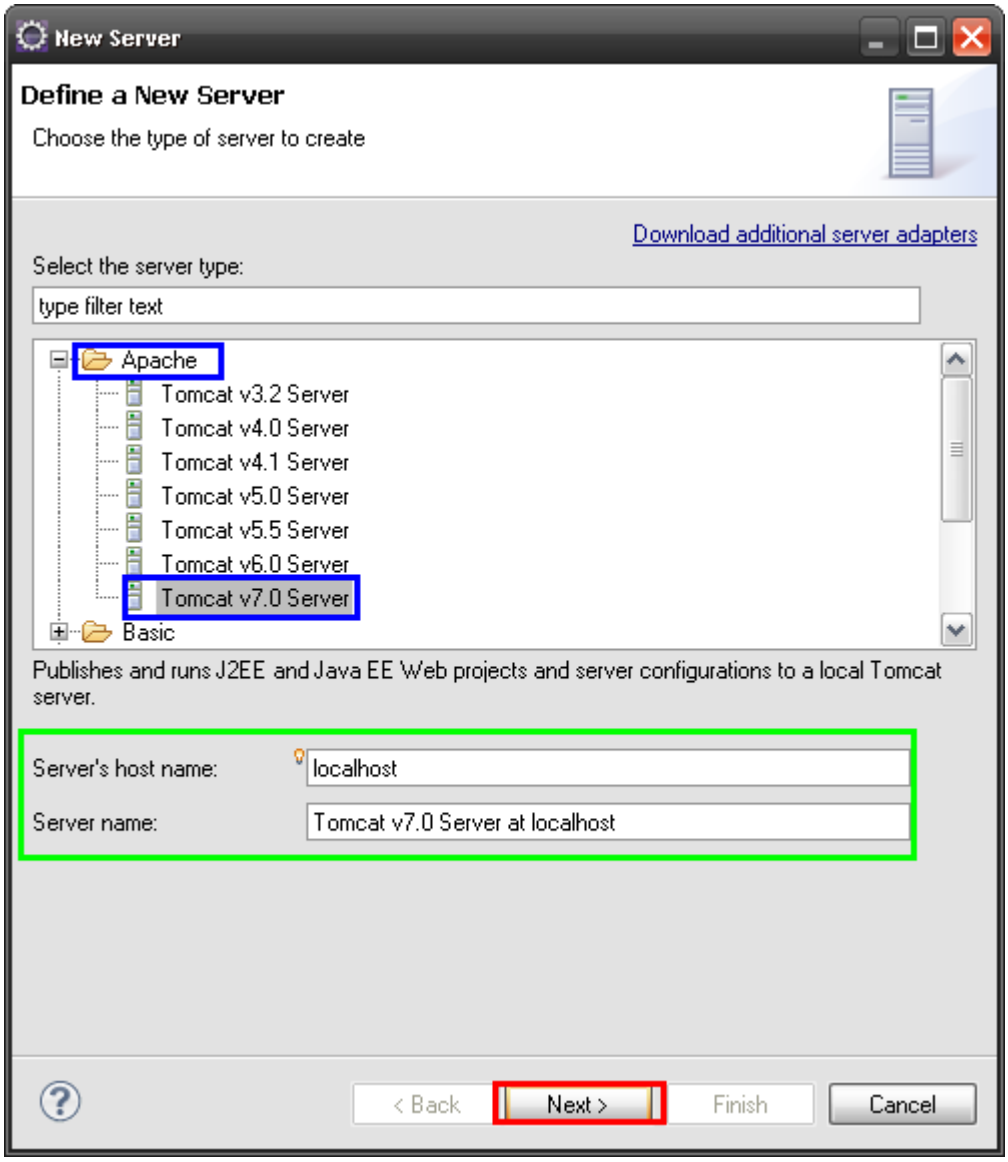


Antes de meter nuestro proyecto, primero agregaremos nuestro Server que en este caso va ser el apache Tomcat.

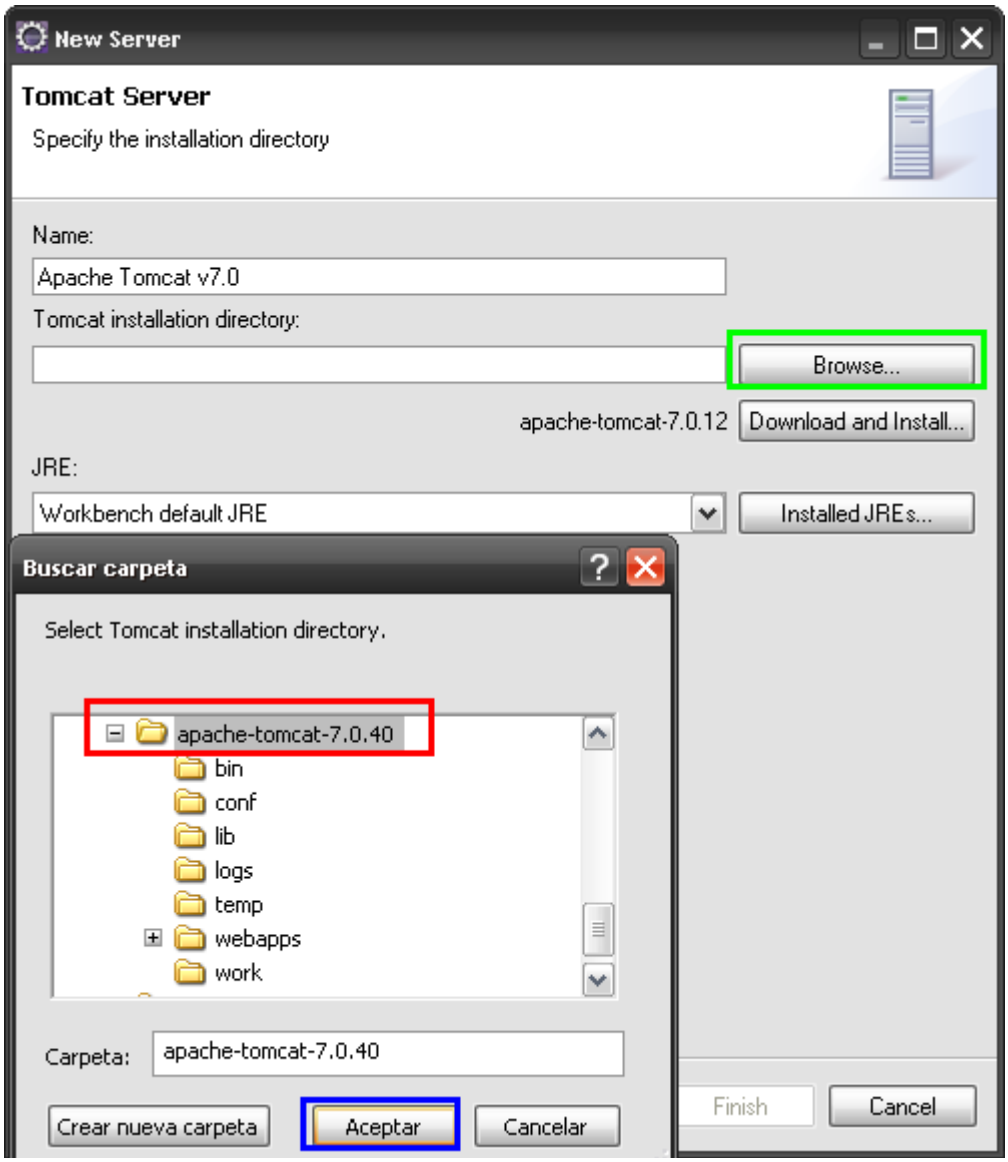
Nos dirigimos al eclipse parte inferior en la pestaña Servers y ahí le damos click derecho y le damos New→Server



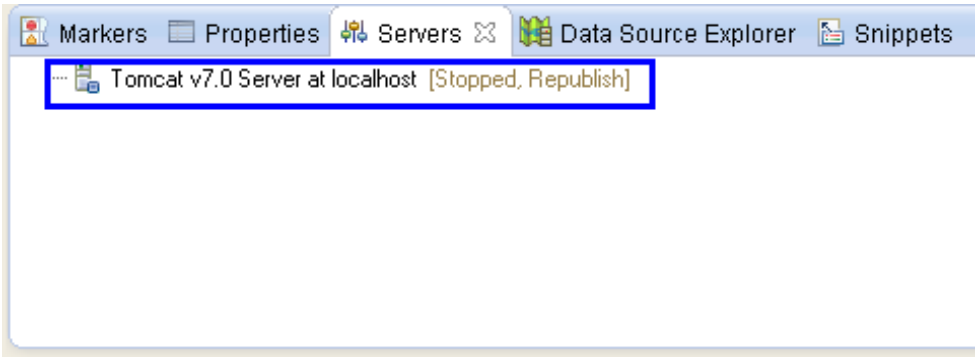
Nos aparecerá una ventana en donde tenemos que elegir la carpeta Apache➔Tomcat v7.0 Server y le damos Next



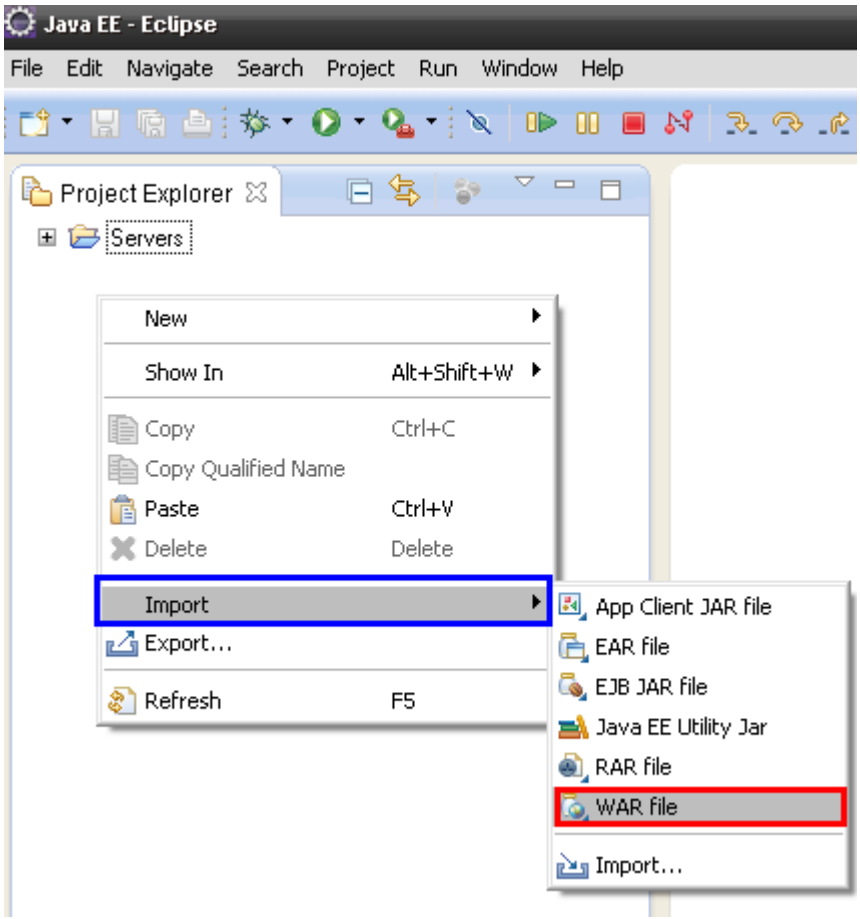
Le damos en el botón "Browse" y buscamos la carpeta el cual descargamos y descomprimimos el apache Tomcat y le damos aceptar y luego Finish.



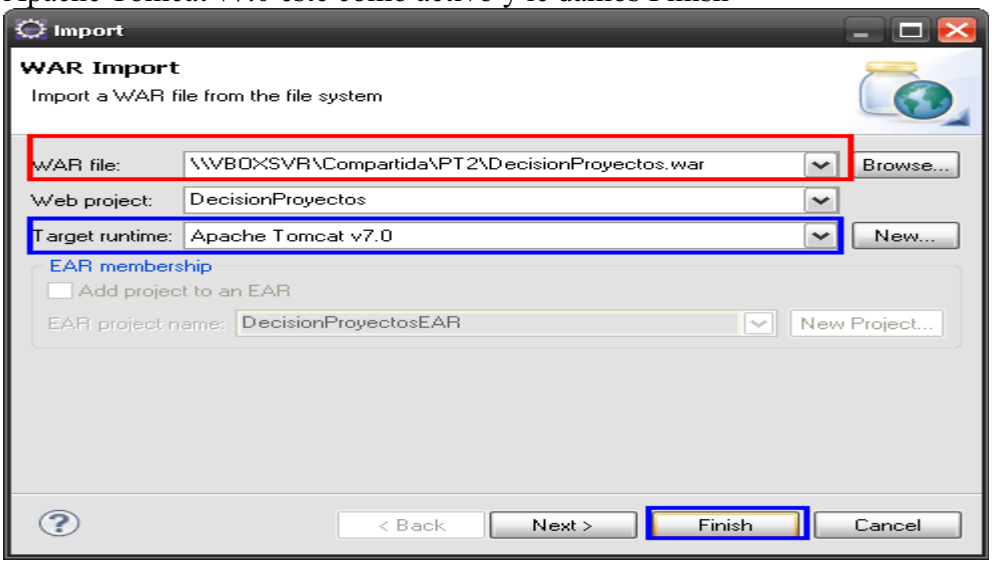
Verificamos que el Server se haya agregado bien al eclipse



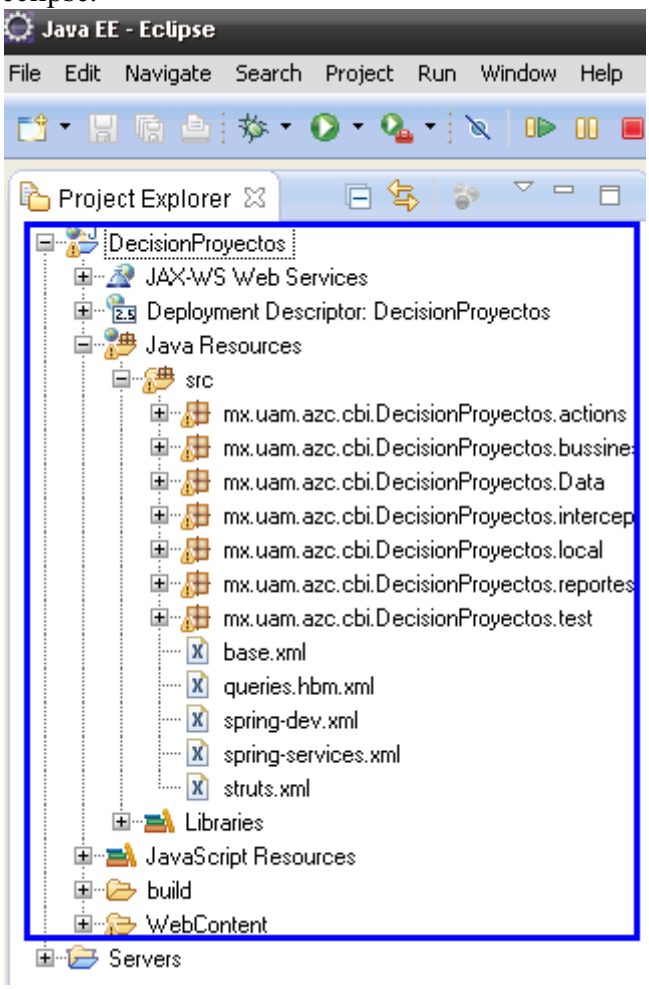
Ahora dentro de la herramienta eclipse meteremos nuestro proyecto con extensión .war llamado “DecisionProyectos.war”. Nos dirigimos en la panel izquierdo, le damos click derecho ➔Import➔ WAR file



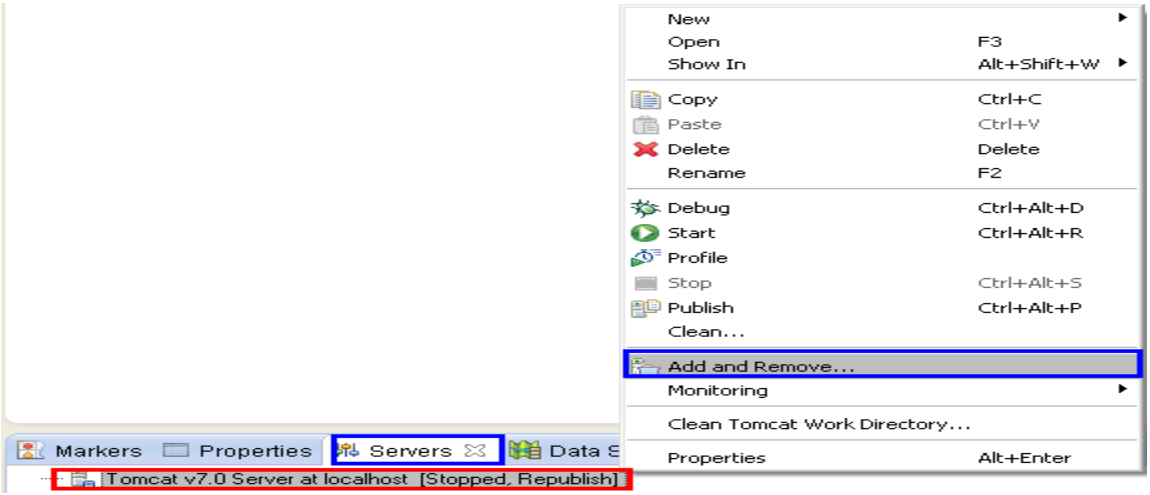
Le damos en Browse y buscamos el proyecto “DecisionProyectos.war”, verificamos que Apache Tomcat v7.0 este como activo y le damos Finish



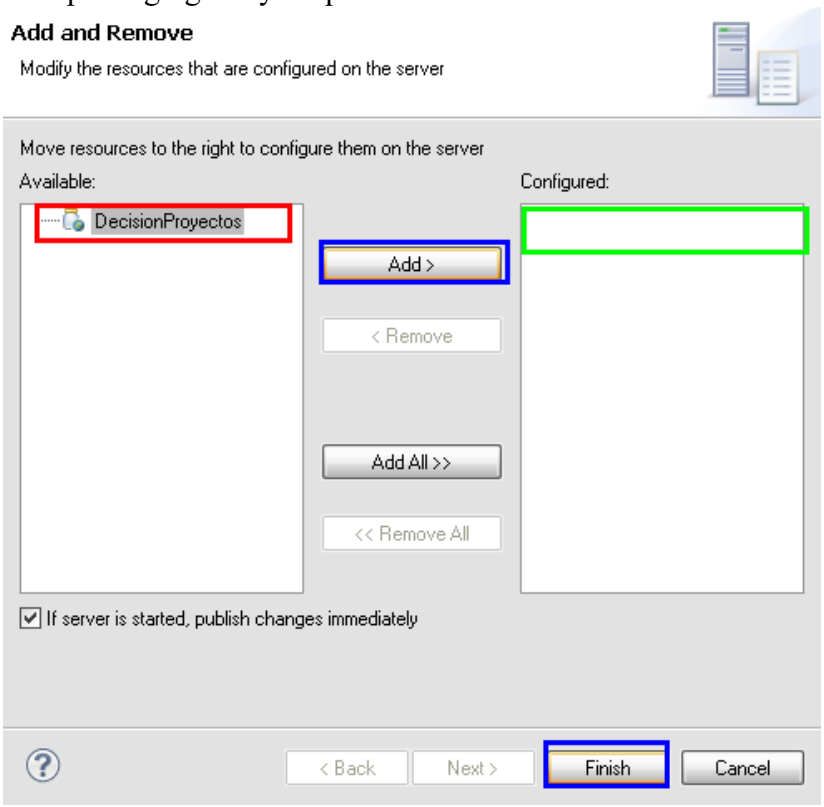
Y Observamos como el proyecto fue importado con éxito a nuestro “Project Explorer” del eclipse.



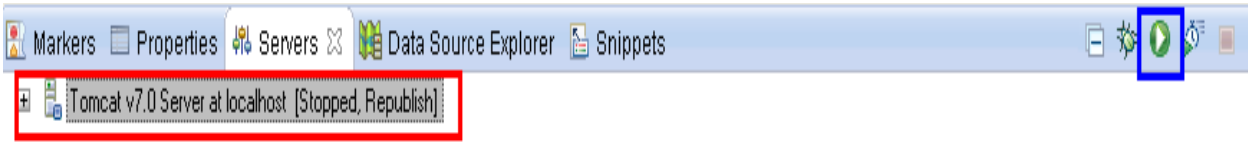
Nos dirigimos nuevamente a la pestaña de Server parte inferior y le damos click derecho➔Add and Remove



Elegimos nuestros recursos de nuestro proyecto “DecisionProyectos” y le damos al botón Add para agregarlo y después le damos Finish.



Una vez hecho esta parte solo le damos al botón de play del Server para levantar los servicios de nuestro proyecto



Y verificamos que los servicios del proyecto se hayan levantado correctamente. Nos dirigimos a http://localhost:8080/DecisionProyectos/login_principal.action y observamos que abra la página y si lo hace es que todo la instalación fue realizada con éxito.

RECURSOS
Para poder llevar a cabo el desarrollo de esta aplicación se utilizarán los siguientes dispositivos:

1. Una computadora portátil:
Nombre del Modelo: Dell vostro
Procesador: Intel Core Duo
Memoria: 2 GB DDR2 667 MHZ
Disco Duro: 250 GB
Sistema operativo: Linux Ubuntu 12.04

2. Una computadora portatil:
Nombre del Modelo: HP
Procesador Intel: Core Duo
Memoria: 2 GB DDR2 567 MHZ
Disco Duro: 250GB
Sistema operativo: Windows 7 Home Premium

MÓDULOS DE PROGRAMACIÓN

Para el desarrollo de este sitio web se planeó el diseño de cinco principales módulos de programación:

Propuestas de proyectos terminales

Modulo diseñado para los profesores y alumnos. En el cual el profesor puede subir todas las propuestas de proyectos terminales para que los alumnos puedan consultarlas.

Proyectos terminales

Modulo diseñado para alumnos que concluyeron su proyecto terminal y suben su documento del proyecto terminal ya concluido.

Reporte Dinámico

Modulo diseñado para profesores y jefe de departamento en el cual generen reportes sobre datos en los cuales están presentes en algún proyecto en un intervalo de fechas.

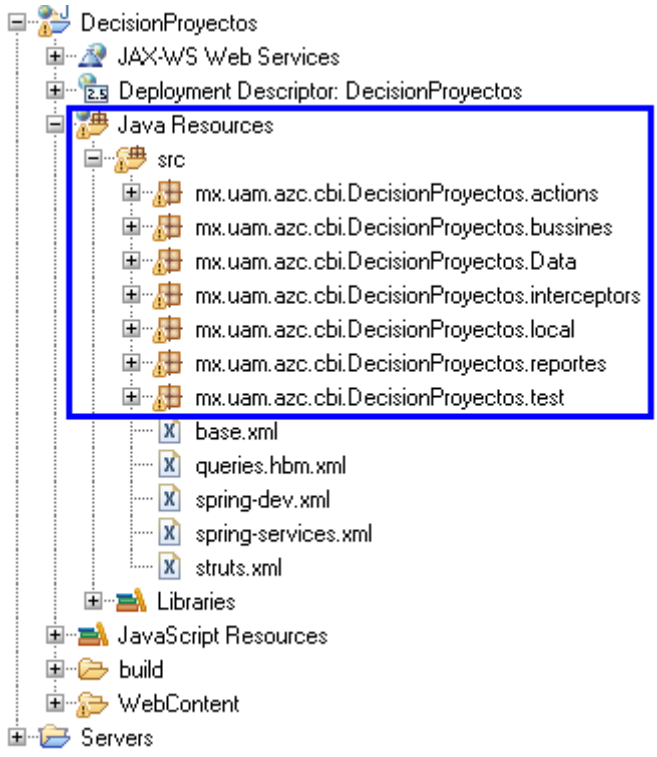
Reporte Programado

Modulo diseñado para profesores y jefes de departamento en el cual generan reportes sobre datos de proyectos existentes, generado ya sea por trimestre o por año.

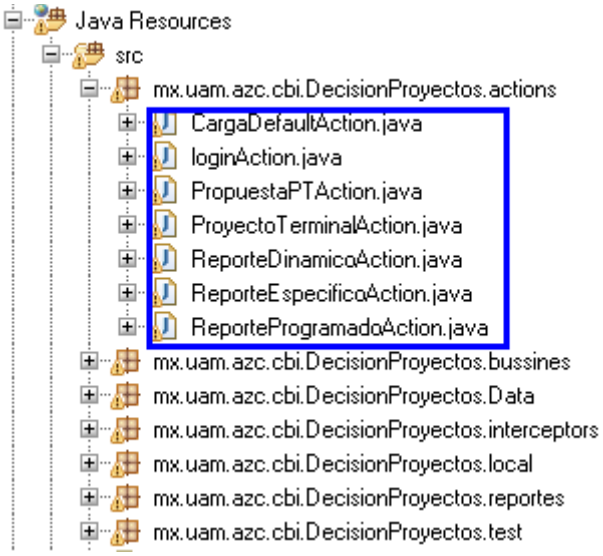
Reporte Específico

Modulo diseñado para profesores y jefes de departamento en el cual generan reportes sobre datos de proyectos existentes divididos por datos de proyecto, temáticos o productos de trabajo.

En nuestro proyecto raíz tenemos varios paquetes en los cuales desarrollan una función diferente pero a su vez se complementan para funcionar adecuadamente el sistema en el sitio.



En el paquete de **mx.uam.azc.cbi.DecisionProyectos.action** encontramos varios archivos de código en el cual su funcionamiento es para las acciones de los botones y ligas de cada módulo.



CargaDefaultAction.- Para la carga masiva de datos en la base de datos.

```
public class CargaDefaultAction extends ActionSupport implements SessionAware {

        // session del navegador
        private Map _sessionData;

   private String path,file,mensaje;
   private LlenaDAO llenaImpl;

   //fileds para subir archivo
         private File fileUpload;
         private String fileUploadContentType;
         private String fileUploadFileName;
         private static final String UPLOAD_FOLDER = "upload";

         //listas de registros
         List <Unidad> lunidad;

        public String formCargaDefault()
        {

                return "formCargaDefault";
        }

        //Subir el archivo de la carga masiva
        public String subirArchivo()
        {
                String redirection= "subirArchivo";

                System.out.println("nombre del archivo : " + fileUploadFileName);

                if (fileUploadFileName.indexOf(".sql") != -1)
                {


                path = "fuentes/";
                String path2 = ContextLoader.getCurrentWebApplicationContext()
                                .getServletContext().getRealPath(path);

                File origen = new File(path2);

                if (origen.exists() != true)
                        origen.mkdirs();

                //Si existe el archivo se crea el path y lo sube si no manda un mensaje de error
                if (origen.exists() == true)
                {

                        origen = new File(path2 + "/"+ fileUploadFileName);

                        try
                   {
                     FileUtils.copyFile(fileUpload, origen);
                     //System.out.println(origen.getPath());
                    _sessionData.put("fileD", origen);
                   redirection = "lista";
                   }
                   catch ( IOException e )
                   {
                        mensaje = "ERROR. no se pudo subir el archivo";
```

loginAction .- Verifica que el usuario ingresado sea alguno que este en la BD

```
public String login() {
//El sistema validara si el usuario ingresado es el correcto
        usuario = usuariosIMPL.leerUsuarioLoginPassword(usuario);
        if (usuario == null) {
                mensaje = "Usuario Incorrecto";

                return "error";
        }

        _sessionData.put("usuario", usuario);
//Si el usuario es correcto obtendrá sus respectivos roles
        if (usuario.getRol().getPermiso() == 1)
        {
                _sessionData.put("lperfil", "admin");
                return "admin";
        }


        if (usuario.getRol().getPermiso() == 2)
        {
                _sessionData.put("lperfil", "profesor");

                return "profesor";
        }

        if (usuario.getRol().getPermiso() == 3)
        {
                _sessionData.put("lperfil", "alumno");
                return "alumno";
        }


        else
        {
                return "error";
        }
```

PropuestaPTAction.- Revisa el usuario logueado y dependiendo del rol le mostrara la lista de propuestas de proyectos terminales

```
//Obtendra primero el usuario logueado con su respectivo rol
public String formPropuesta()
    {
        usuario = (Usuario)_sessionData.get("usuario");

        //Si el usuario tiene el rol 2, le dara una lista de propuestas
        if (usuario.getRol().getPermiso() ==2)
        {
            listapropuestas =
propuestaptIMPL.consultarPropuestasProfessor(usuario.getId());
        }
        else
        {
    // De lo contrario solo le mostrara el modulo en odnde puede subir su propuesta
            listapropuestas = propuestaptIMPL.consultarPropuestas();
            lprofesores = usuariosIMPL.leerProfesor();
        }

        return "formPropuesta";
    }
```

ProyectoTerminalAction.- Desplegara la lista de alumnos con algún proyecto terminal o el módulo de alta de proyecto terminales.

```
//Muestra la lista de profesores y de proyectos terminales que hay en la BD
public String formProyectoTerminal()
        {
                lprofesores = new ArrayList<Profesor>();
                lproyectos = new ArrayList<ProyectoTerminal> ();

//Validara el usuario logueado y dependiendo del usuario verá si tiene algún proyecto o no
tiene proyecto.
                usuario = (Usuario)_sessionData.get("usuario");
            alumno = usuariosIMPL.leerAlumno(usuario.getId());
            if (!alumno.isEstadopt())
            {
                lproyectos = proyectoTerminalIMPL.leerProyectoTerminal();
                lprofesores = usuariosIMPL.leerProfesor();
            }
            else
            {
        proyectoterminal = proyectoTerminalIMPL.leerProyectoTerminal(alumno.getIdpt());

                if (proyectoterminal.isEstado())
                lalumnos = usuariosIMPL.leerAlumnoPT(proyectoterminal.getId());
                else proyectoterminal = new ProyectoTerminal();
            }
                return "formProyectoTerminal";
         }
```

ReporteDinamicoAction.- Genera las acciones correspondiente para que los reportes dinamicos se generen correctamente.

```
//Muestra la lista de reportes dinamicos
public String formReporteDinamico()
        {
                //Las acciones de los botones del campo de unidades
                inicia ();
                lunidades = catalogosImpl.leerUnidad();

                //Las acciones de los botones del campo de divisiones
                ldivicion = (List<Division>)_sessionData.get("ldiviciones");
                if (ldivicion == null)
                        ldivicion = new ArrayList<Division>();
                //Las acciones de los botones del campo de departamentos
                ldepartamentos = (List<Departamento>)_sessionData.get("ldepartamentos");
                if (ldepartamentos == null)
                        ldepartamentos = new ArrayList<Departamento>();
                //Las acciones de los botones del campo de proyectos
                lproyectos = (List<Proyecto>)_sessionData.get("lproyectos");
                if (lproyectos == null)
                        lproyectos = new ArrayList<Proyecto>();

                lreportes = reportesImpl.leerReporteDinamico();

                return "formReporteDinamico";
        }
```

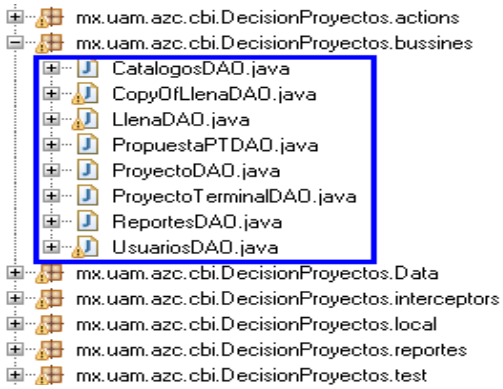
ReporteEspecificoAction.- Genera las acciones correspondientes para que los reportes específicos se generen correctamente.

```
public String formReporteEspecifico()
        {
          //Las acciones de los botones del campo de unidades

                inicia ();
                lunidades = catalogosImpl.leerUnidad();

                //Las acciones de los botones del campo de profesores

                lprofesor = usuariosIMPL.leerProfesor();

                ltematicas = catalogosImpl.leerTematica();
                lptrabajo = catalogosImpl.leerProductosDeTrabajos();
                lreportes = reportesImpl.leerReporteEspecifico();
                //Las acciones de los botones del campo de divisiones


                ldivicion = (List<Division>)_sessionData.get("ldiviciones");
                if (ldivicion == null)
                        ldivicion = new ArrayList<Division>();
                //Las acciones de los botones del campo de departamentos

                ldepartamentos = (List<Departamento>)_sessionData.get("ldepartamentos");
                if (ldepartamentos == null)
                        ldepartamentos = new ArrayList<Departamento>();

                //Obtiene la lista de proyectos en los botones del campo de proyecto

                lproyectos = (List<Proyecto>)_sessionData.get("lproyectos");
                if (lproyectos == null)
                        lproyectos = new ArrayList<Proyecto>();

                return "formReporteEspecifico";
        }
```

ReporteProgramadoAction.- Genera las acciones correspondiente para que los reportes programados se generen correctamente.

```
public String formReporteProgramado()
        {
                //Lee los proyectos existente
                lproyectos = proyectoIMPL.leerProyecto();
                lreportes = reportesImpl.leerReporteProgramado();
                return "formReporteProgramado";     }
public String generar ()throws JRException
        {
        //Genera una lista de reportes programados dependiendo de si es trimestral o anual
                tipo = (Integer)_sessionData.get("tipo");
                ReporteProgramado rprog = new ReporteProgramado();
                final Map<String, Object> parameters = new HashMap<String, Object>();
                prog.setTipo("Programado");

                if (tipo == 1)
                {
                        rprog.setPeriodo("trimestral");
                }
                if (tipo ==2)
                {
                        rprog.setPeriodo("anual");
                }
                //Y muestra el reporte programado en el modulo
                rprog.setNombre(nombrer) ;
                rprog.setId(reportesImpl.insertarReporteProgramado(rprog));

                return "generar";
        }
```

En el paquete de **mx.uam.azc.cbi.DecisionProyectos.bussines** encontramos varios archivos de código en el es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.



CatalogosDAO.- Muestra el catálogo de las interfaces que se van a utilizar para el acceso a los datos.

```
public interface CatalogosDAO {
        // unidades
        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public Unidad leerUnidad(int id);
        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public List<Unidad> leerUnidad();
        // areas
        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public AreaDesarrollo leerAreaDesarrollo(int id);

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public List<AreaDesarrollo> leerAreaDesarrollo();
        // divicion
        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public Division leerDivision(int id);

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public List<Division> leerDivision();

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public List<Division> buscarDivision(int id);
        // departamentos
        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public Departamento leerDepartamento(int id);

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public List<Departamento> leerDepartamento();

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public List<Departamento> buscarDepartamento(int id);
        //tematicas

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public Tematica leerTematica(int id);

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public List<Tematica> leerTematica();

        @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
        public Tematica leerTematicap( int idprofesor);
}
```

CopyOfLlenaDAO.- La interfaz que manda a llamar para el llenado de los objetos presentes.

```
public interface CopyOfLlenaDAO {

        // unidad
        public Unidad cUnidad (String registro);
        public int iUnidad (Unidad unidad);

        // divicion
        public Division cdivicion (String registro);
        public int idivicion (Division divicion);

        // departamento
        public Departamento cdepartamento (String registro);
        public int idepartamento (Departamento departamento);

        // rol
        public Rol crol (String registro);
        public int irol (Rol rol);

        // usuario
        public Usuario cusuario (String registro);
        public int iusuario (Usuario usuario);

        // areadesarrollo
        public AreaDesarrollo careadesarrollo (String registro);
        public int iareadesarrollo (AreaDesarrollo areadesarrollo);

        // alumno
        public Alumno calumno (String registro);
        public int ialumno (Alumno alumno);

        // proyecto
        public Proyecto cproyecto (String registro);
        public int iproyecto (Proyecto proyecto);

        // proyectocolaborador
        public ProyectoColaborador cproyectocolaborador (String registro);
        public int iproyectocolaborador (ProyectoColaborador proyectocolaborador);

        // tematica
        public Tematica ctematica (String registro);
        public int itematica (Tematica tematica);

        // productotrabajo
        public ProductoDeTrabajo cproductotrabajo (String registro);
        public int iproductotrabajo (ProductoDeTrabajo productotrabajo);

        // productostrabajo
        public ProductosDeTrabajo cproductostrabajo (String registro);
        public int cproductostrabajo (ProductosDeTrabajo productostrabajo);

        // profesor
        public Profesor cprofesor (String registro);
        public int iprofesor (Profesor profesor);

        // lee origen

        public List<String> lee(File origen) throws FileNotFoundException, IOException;
}
```

LlenaDAO.- La interfaz que manda a llamar para el llenado de los objetos presentes.

```
public interface LlenaDAO {

        // unidad
        public boolean cUnidad (String registro);


        // divicion
        public boolean cdivicion (String registro);


        // departamento
        public boolean cdepartamento (String registro);


        // rol
        public boolean crol (String registro);


        // usuario
        public boolean cusuario (String registro);


        // areadesarrollo
        public boolean careadesarrollo (String registro);


        // alumno
        public boolean calumno (String registro);


        // proyecto
        public boolean cproyecto (String registro);


        // proyectocolaborador
        public boolean cproyectocolaborador (String registro);


        // tematica
        public boolean ctematica (String registro);


        // productotrabajo
        public boolean cproductotrabajo (String registro);


        // productostrabajo
        public boolean cproductostrabajo (String registro);


        // profesor
        public boolean cprofesor (String registro);


        // lee origen

        public List<String> lee(File origen) throws FileNotFoundException, IOException;
}
```

PropuestaPTDAO.- La interfaz de objetos de acceso a los datos de la propuesta del pt.

```
public interface PropuestaPTDAO
{
//Interface para leer la propuestaPT
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public Propuestapt leerPropuestaPT( int id );
  //Interface para consultar la propuestaPT
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<Propuestapt> consultarPropuestasByProfesor( Usuario usuario);
  //Interface para leer la lista la propuestaPT
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<Propuestapt> consultarPropuestas();

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<Propuestapt> consultarPropuestasProfessor(int idprofesor);
  //inserta propuesta  PT

  public int insertarPropuestaPT( Propuestapt propuestaPT );

  public int cambiarPropuestaPT( Propuestapt propuestaPT );

}
```

ProyectoDAO.- La interfaz de objetos de acceso a los datos del Proyecto.

```
//Interfaces para el acceso a los datos del los objetos del proyecto
public interface ProyectoDAO
{
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public Proyecto leerProyecto( int id );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<Proyecto> leerProyecto( );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<Proyecto> leerProyectoPeriodo(int periodo);

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<Proyecto> buscarProyecto(int id);

  public int insertarProyecto( Proyecto proyecto );

  public int cambiarProyecto( Proyecto proyecto );

}
```

ProyectoTerminalDAO.- La interfaz de objetos de acceso a los datos del Proyecto Terminal.

```
//Interfaces para el acceso a los datos del los objetos del proyecto terminal
public interface ProyectoTerminalDAO
{
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public ProyectoTerminal leerProyectoTerminal( int id );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<ProyectoTerminal> leerProyectoTerminal( );

  public int insertarProyectoTerminal( ProyectoTerminal proyectoTerminal );

  public int cambiarProyectoTerminal( ProyectoTerminal proyectoTerminal );
}
```

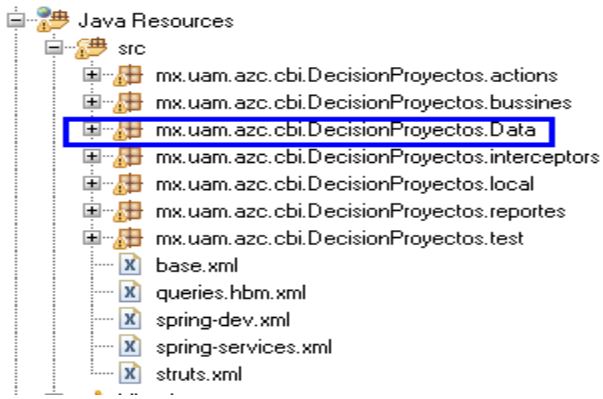
ProyectoTerminalDAO.- La interfaz de objetos de acceso a los datos de los reportes.

```
//Interfaces para el acceso a los datos de los objetos del proyecto terminal
public interface ReportesDAO
{
// dinamico
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public ReporteDinamico leerReporteDinamico( int id );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<ReporteDinamico> leerReporteDinamico();

  public int insertarReporteDinamico( ReporteDinamico reporteDinamico );

  public int cambiarReporteDinamico(ReporteDinamico reporteDinamico );
//  programado
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public ReporteProgramado leerReporteProgramado( int id );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<ReporteProgramado> leerReporteProgramado();

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<ReporteProgramado> leerReporteProgramadoPeriodo(int periodo);

  public int insertarReporteProgramado( ReporteProgramado reporteProgramado);

  public int cambiarReporteProgramado(ReporteProgramado reporteProgramado );
// especifico
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public ReporteEspecifico leerReporteEspecifico( int id );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<ReporteEspecifico> leerReporteEspecifico();

  public int insertarReporteEspecifico( ReporteEspecifico reporteEspecifico );

  public int cambiarReporteEspecifico(ReporteEspecifico reporteEspecifico );
}
```

UsuariosDAO.- La interfaz de objetos de acceso a los datos de los usuarios.

```
//Interfaces para el acceso a los datos de los objetos del proyecto terminal
public interface UsuariosDAO
{

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public Usuario leerUsuario( int id );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public Usuario leerUsuarioLoginPassword( Usuario usuario );

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List<Usuario> leerUsuarioRol( int rol );

  public int insertarUsuario(Usuario usuario );
  public int cambiarUsuario(Usuario usuario );

  //lee profesor

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public Profesor leerProfesorAD(int id);

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public Profesor leerProfesor(int id);

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List <Profesor> leerProfesor();

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List <ProyectoColaborador> leerProyectoColaborador(int idproyecto);

  //lee alumno
  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public Alumno leerAlumno(int id);

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List <Alumno> leerAlumnoPT(int idalumno);

  @Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  public List <Alumno> leerAlumnos();

  public int cambiarAlumno(Alumno alumno );

}
```

En el paquete de **mx.uam.azc.cbi.DecisionProyectos.data** encontramos varios archivos de código en donde se implementa más que nada las principales funciones del sistema.



propuestaPT.- Es donde se implementa todas las funciones de los datos de la propuesta del proyecto terminal.

```
public class Propuestapt implements java.io.Serializable {

	private int idpropuestapt;
	private String nombre;
	private String descripcion;
	private String urlarchivo;
	private Usuario usuario;
  private boolean estado;

	public Propuestapt() {
		this.estado= true;
	}
	@Id
	@GeneratedValue(strategy = IDENTITY)
	@Column(name = "idpropuestapt", unique = true, nullable = false)
	public int getIdpropuestapt() {
		return idpropuestapt;
	}
	public void setIdpropuestapt(int idpropuestapt) {
		this.idpropuestapt = idpropuestapt;
	}
	@Column(name = "nombre")
	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	@Column(name = "descripcion")
	public String getDescripcion() {
		return this.descripcion;
	}

	public void setDescripcion(String descripcion) {
		this.descripcion = descripcion;
	}
	@Column(name = "urlarchivo")
	public String getUrlarchivo() {
		return this.urlarchivo;
	}

	public void setUrlarchivo(String urlarchivo) {
		this.urlarchivo = urlarchivo;
	}
	@OneToOne @JoinColumn( name = "usuario" )
	public Usuario getUsuario() {
		return usuario;
}
```

ProyectoTerminal.- Se definen los parámetros para que pueda extraer datos de la base de datos y así poder utilizar los datos para generar los archivos del proyecto terminal

```
public class ProyectoTerminal implements Serializable {
	private int id;
	private String nombre;
	private String url;
	private Profesor profesor;
	private boolean estado;
	private String alumnos;
  public ProyectoTerminal ()
  {
	  this.estado = true;
	  this.url = "";
	  this.alumnos ="";
  }
	@Id
	@Column(name = "idproyecto_terminal")
	@GeneratedValue(strategy = IDENTITY)
	public int getId() {
		return id;
	}
	@Column(name = "nombre")
	public String getNombre() {
		return nombre;   }
	@Column(name = "urlarchivo")
	public String getUrl() {
		return url;
	}
	@ManyToOne
	@JoinColumn(name = "profesor")
	public Profesor getProfesor() {
		return profesor;
	}
	@Column(name = "estado")
	public boolean isEstado() {
		return estado;
	}
	@Column(name = "alumnos")
	public String getAlumnos() {
		return alumnos;
	}
	public void setId(int id) {
		this.id = id;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public void setUrl(String url) {
		this.url = url;
	}
	public void setProfesor(Profesor profesor) {
		this.profesor = profesor;
	}
	public void setEstado(boolean estado) {
		this.estado = estado;
	}
	public void setAlumnos(String alumnos) {
		this.alumnos = alumnos;
	}
}
```

Reporte Dinámico.- Se definen los parámetros para que una cuando se genere el reporte dinámico extraiga los datos correspondientes, dependiendo de la selección del usuario.

```
public class ReporteDinamico implements Serializable {
   private int id;
   private String nombre;
        private Calendar Ifecha = Calendar.getInstance();
        private Calendar ffecha = Calendar.getInstance();
        private Proyecto proyecto;
        private int profesor;
        private String tipo;
        private String urlpdf;
        private boolean estado;
        public ReporteDinamico ()
        {
                this.urlpdf = "";
                this.estado = true;
        }
        @Id
        @Column(name = "idinformedinamico")
        @GeneratedValue(strategy = IDENTITY)
        public int getId() {
                return id;              }
        @Column(name = "nombre")
        public String getNombre() {
                return nombre;      }
        @Column(name = "fechainicio")
        public Calendar getIfecha() {
                return Ifecha;        }
        @Column(name = "fechafin")
        public Calendar getFfecha() {
                return ffecha;        }
        @ManyToOne
        @JoinColumn(name = "proyecto")
        public Proyecto getProyecto() {
                return proyecto;     }
        @Column(name = "tipo")
        public String getTipo() {
                return tipo;            }
        @Column(name = "urlpdf")
        public String getUrlpdf() {
                return urlpdf;        }
        @Column(name = "estado")
        public boolean isEstado() {
                return estado;       }
        public void setId(int id) {
                this.id = id;
        }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public void setIfecha(Calendar ifecha) {
                Ifecha = ifecha;
        }
        public void setFfecha(Calendar ffecha) {
                this.ffecha = ffecha;
        }
        public void setProyecto(Proyecto proyecto) {
                this.proyecto = proyecto;
        }
        public void setTipo(String tipo) {
                this.tipo = tipo;
        }
        public void setUrlpdf(String urlpdf) {
                this.urlpdf = urlpdf;             }
        public void setEstado(boolean estado) {
                this.estado = estado;           }
}
```

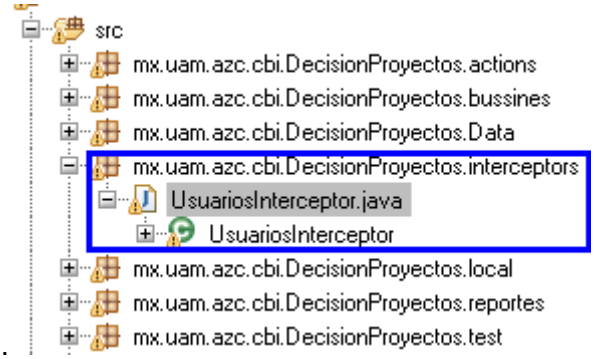
Reporte Programado.- Se definen los parámetros para que una cuando se genere el reporte programado extraiga los datos correspondientes, dependiendo de la selección del usuario.

```
public class ReporteProgramado implements Serializable {
//Se definen los parámetros
   private int id;
   private String nombre;
        private Calendar Ifecha = Calendar.getInstance();
        private String periodo;
        private String tipo;
        private String urlpdf;
        private boolean estado;
        public ReporteProgramado ()
        {
                this.urlpdf = "";
                this.estado = true;
        }
        @Id
        @Column(name = "idinformeProgramado")
        @GeneratedValue(strategy = IDENTITY)
        public int getId() {
                return id;          }
        @Column(name = "nombre")
        public String getNombre() {
                return nombre;  }
        @Column(name = "fechainicio")
        public Calendar getIfecha() {
                return Ifecha;    }
        @Column(name = "periodo")
        public String getPeriodo() {
                return periodo;  }
        @Column(name = "tipo")
        public String getTipo() {
                return tipo;        }
        @Column(name = "urlpdf")
        public String getUrlpdf() {
                return urlpdf;    }
        @Column(name = "estado")
        public boolean isEstado() {
                return estado;   }
        public void setId(int id) {
                this.id = id;       }
        public void setNombre(String nombre) {
                this.nombre = nombre;
        }
        public void setIfecha(Calendar ifecha) {
                Ifecha = ifecha;
        }
        public void setPeriodo(String periodo) {
                this.periodo = periodo;
        }
        public void setTipo(String tipo) {
                this.tipo = tipo;
        }
        public void setUrlpdf(String urlpdf) {
                this.urlpdf = urlpdf;
        }
        public void setEstado(boolean estado) {
                this.estado = estado;
        }
}
```

Reporte Específico.- Se definen los parámetros para que una cuando se genere el reporte específico extraiga los datos correspondientes, dependiendo de la selección del usuario.

```
public class ReporteEspecifico implements Serializable {
    private int id;
    private String nombre;
    private Calendar cfecha = Calendar.getInstance();
    private Calendar Ifecha = Calendar.getInstance();
            private Calendar Ffecha = Calendar.getInstance();
            private String tipo;
            private String urlpdf;
            private boolean estado;
            public ReporteEspecifico ()
            {
                    this.urlpdf = "";
                    this.estado = true;
            }
            @Id
            @Column(name = "idinformeespecifico")
            @GeneratedValue(strategy = IDENTITY)
            public int getId() {
                    return id;          }
            @Column(name = "nombre")
            public String getNombre() {
                    return nombre;    }
            @Column(name = "fechaCreacion")
            public Calendar getCfecha() {
                    return cfecha;     }
            @Column(name = "fechainicio")
            public Calendar getIfecha() {
                    return Ifecha;     }
            @Column(name = "fechafin")
            public Calendar getFfecha() {
                    return Ffecha;
            }
            @Column(name = "tipo")
            public String getTipo() {
                    return tipo;
            }
            @Column(name = "urlpdf")
            public String getUrlpdf() {
                    return urlpdf;
            }
            @Column(name = "estado")
            public boolean isEstado() {
                    return estado;
            }
            public void setId(int id) {
                    this.id = id;
            }
            public void setNombre(String nombre) {
                    this.nombre = nombre;
            }
            public void setIfecha(Calendar ifecha) {
                    Ifecha = ifecha;
            }
            public void setTipo(String tipo) {
                    this.tipo = tipo;    }
            public void setUrlpdf(String urlpdf) {
                    this.urlpdf = urlpdf;
            }
            public void setEstado(boolean estado) {
                    this.estado = estado;
            }
}
```

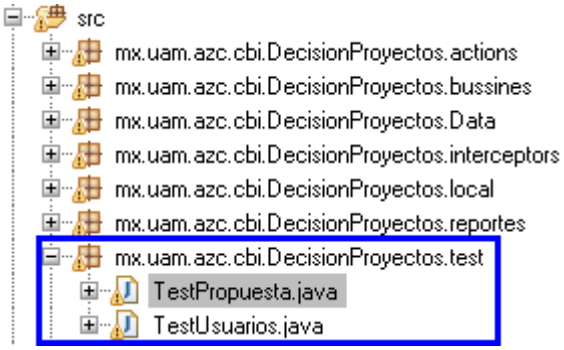
En el paquete de **mx.uam.azc.cbi.DecisionProyectos.interceptors** encontramos varios archivos de código en donde se implementa métodos que actúan al detectar determinado evento dependiente del estado de un objeto. Actúan mediante anotaciones" en el código interceptando las acciones entre los objetos y posibilitando el control de las mismas.



UsuarioInterceptor.- El usuario interceptor se crea para interceptar las acciones del objeto en este caso para el usuario.

```
//Se implementa el usuario interceptor para el usuario
public class UsuariosInterceptor implements Interceptor
{
  public void destroy()
  {
    System.out.println( "Destruyendo UsuariosInterceptor..." );
  }

  public void init()
  {
    System.out.println( "Inicializando UsuariosInterceptor..." );
  }

  public String intercept( ActionInvocation invocation ) throws Exception
  {



    Map<?, ?> session = invocation.getInvocationContext().getSession();
    Usuario usuario = ( Usuario )session.get( "usuario" );

    if ( usuario != null )
    {
      if ( usuario.getRol().getPermiso() == 1
          || usuario.getRol().getPermiso() == 2 )
      {
        return invocation.invoke();
      }
      else
        return "error";

    }
    else
      return "error";

  }
}
```

En el paquete de **mx.uam.azc.cbi.DecisionProyectos.test** encontramos varios archivos de código en donde se hacen algunas pruebas sobre como insertar, buscar y editar la propuesta de proyectos terminales. Este módulo se hizo antes de la implementación de los demás modulo solo para probar que funcionara los DAO (Data Access Object).



TestPropuesta.- Se probaron algunos datos para verificar que funcionaran correctamente los DAO (Data Access Object) para las propuestas.

```
// CLASE PARA PROBAR LOS DAO DEL SISTEMA
public class TestPropuesta
{
  public static void main( String[] args )
  {
    try
    {
      new TestPropuesta().execute();
    }
    catch ( Exception e )
    {
      e.printStackTrace();
    }
  }

  private void execute() throws Exception
  {
    usuarios();

    System.out.println( "Done!!" );
  }

  private void usuarios()
  {
     // insercion
//     AdministradorUsuarios administrador = getAdministradorUsuarios();
//     RolDTO rol = new RolDTO();
//     rol.setId( 2 );
//
//     UsuarioDTO usuario = new UsuarioDTO();
//     usuario.setNombre( "ulises" );
//     usuario.setNombreUsuario( "usuario" );
//     usuario.setContrasenia( "usuario" );
//     usuario.setCorreo( "usuario@hotmail.com" );
//     usuario.setTelefono( "5538183685" );
//     usuario.setRol( rol );
//     administrador.insertarUsuario( usuario );
    }
```

TestUsuarios.- Se probaron algunos datos para verificar que funcionaran correctamente los DAO (Data Access Object) para los usuarios.

```
// CLASE PARA PRBAR LOS DAO DEL SISTEMA
public class TestUsuarios
{
  public static void main( String[] args )
  {
    try
    {
      new TestUsuarios().execute();
    }
    catch ( Exception e )
    {
      e.printStackTrace();
    }
  }

  private void execute() throws Exception
  {
    usuarios();

    System.out.println( "Done!!" );
  }

  private void usuarios()
  {
 // insercion
//     AdministradorUsuarios administrador = getAdministradorUsuarios();
//     RolDTO rol = new RolDTO();
//     rol.setId( 2 );
//
//     UsuarioDTO usuario = new UsuarioDTO();
//
//     usuario.setNombre( "israel" );
//     usuario.setNombreUsuario( "usuario" );
//     usuario.setContrasenia( "usuario" );
//     usuario.setCorreo( "usuario@hotmail.com" );
//     usuario.setTelefono( "5538183685" );
//     usuario.setRol( rol );
//     administrador.insertarUsuario( usuario );

  }
```

MANUAL DE USUARIO:

Este documento contiene la descripción de los pasos a seguir para utilizar y gestionar correctamente las herramientas con las que cuenta el espacio virtual Tecno-Educativo, al igual que permite conocer el funcionamiento interno de esta aplicación.

¿Qué se necesita para accesar al sitio?

Es necesario tener instalado un navegador ó explorador de internet para buscar y mostrar la página como por ejemplo: Internet Explorer, Mozilla Firefox, Safari, Opera, Google Chrome, Netscape, Konqueror, etc.

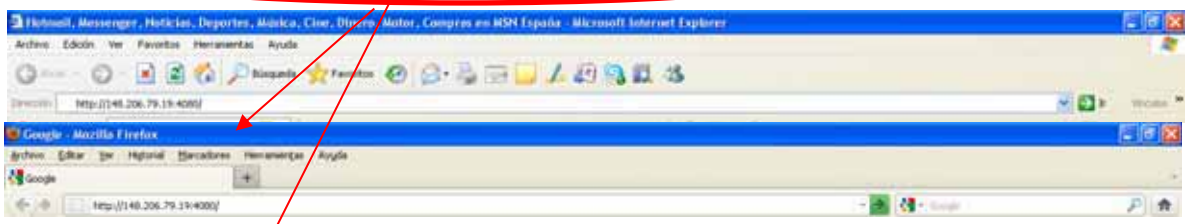
¿Cómo entrar al sitio?

Deberá ingresar en la barra de navegación la siguiente dirección electrónica:

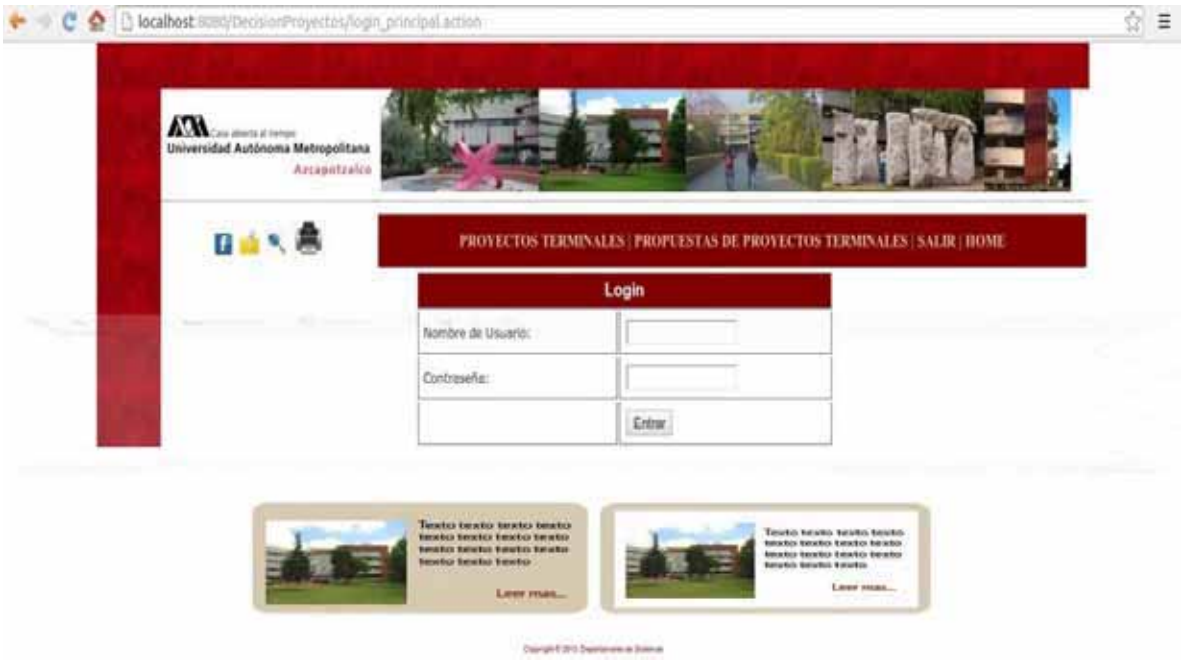
saisis.azc.uam.mx:6080/DecisionProyectos/login_principal.action

Ejemplo para ingresar la dirección en explorador Mozilla Firefox e Internet Explorer:

saisis.azc.uam.mx:6080/DecisionProyectos/login_principal.action

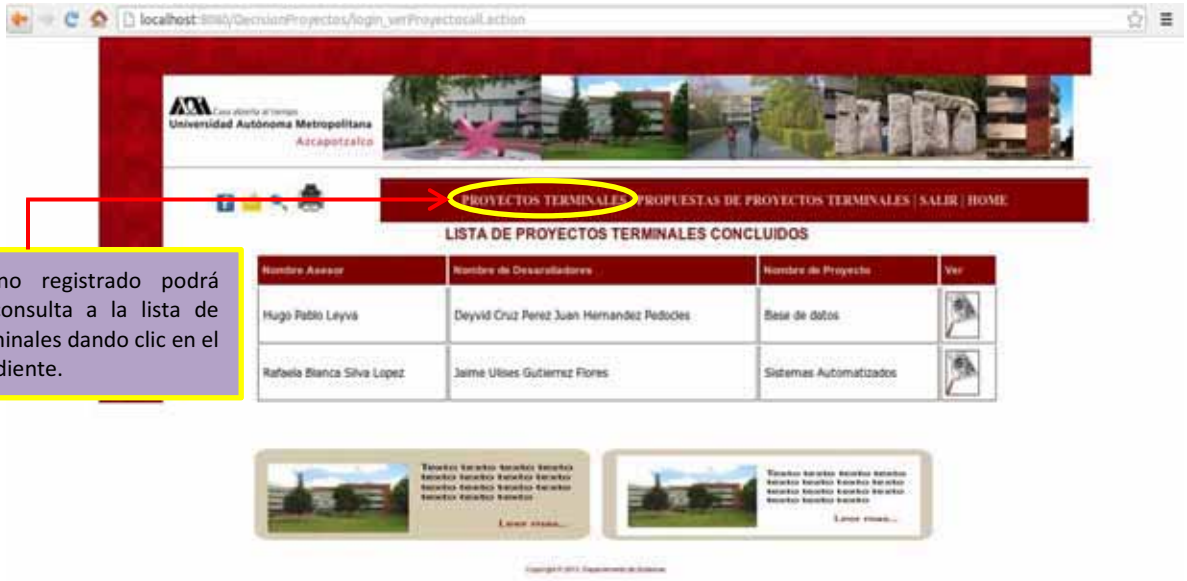


Al haber ingresado correctamente la dirección se desplegara una pantalla como la siguiente:



Sí el usuario del Sistema ya se encuentra registrado en la Base de datos podrá accesar su usuario y contraseña y hacer uso del sistema de acuerdo a los privilegios correspondientes a su tipo de usuario, de lo contrario tendrá que recurrir al administrador para que sea dado de alta y así poder realizar las gestiones correspondientes.

En caso de que no se cuente con un usuario y contraseña solo se podrá realizar la consulta de Proyectos Terminales y consulta de Propuestas de Proyectos Terminales, esto sin necesidad de estar registrado en la Base de Datos, tal y como se muestra en la siguiente imagen:



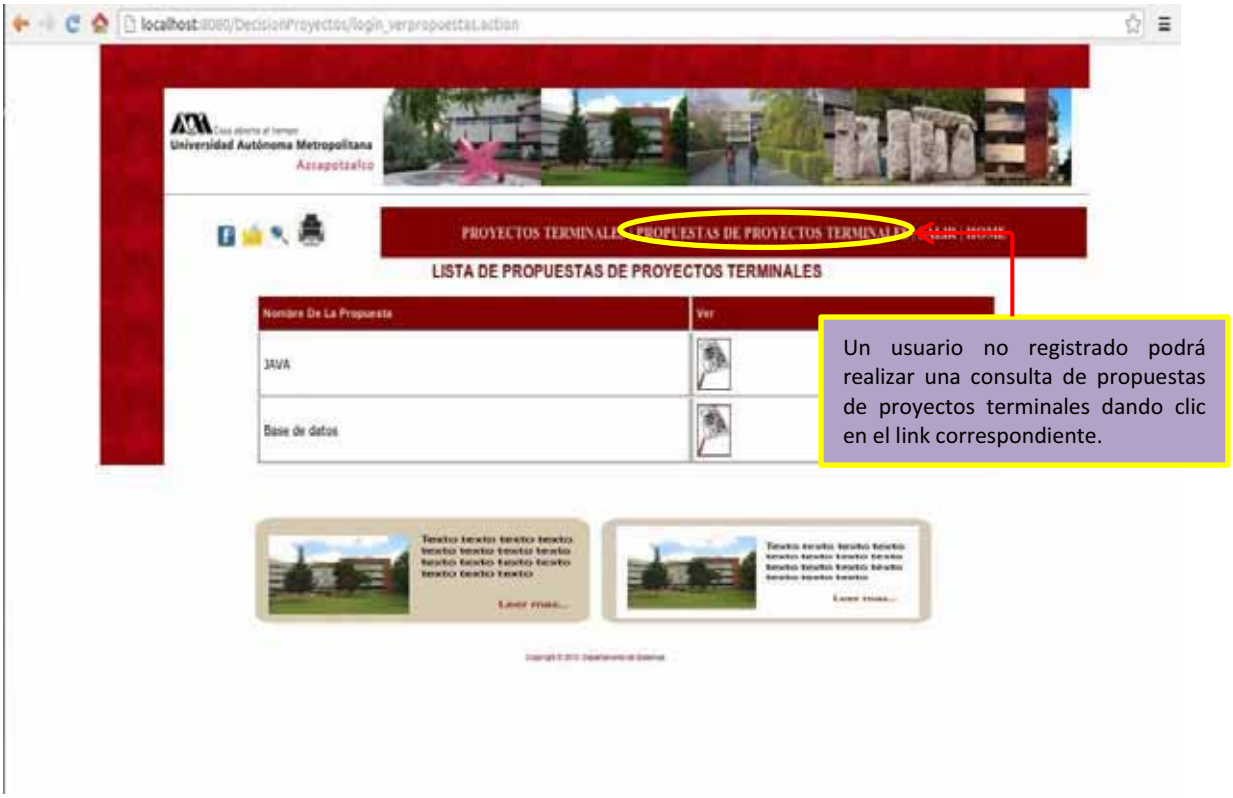
Un usuario no registrado podrá realizar una consulta a la lista de proyectos terminales dando clic en el link correspondiente.

Para poder realizar una consulta a cualquier proyecto terminal se deberá dar clic en el icono representado con una lupa tal y como se muestra en la imagen siguiente:

Como se puede observar en la lista de proyectos terminales se indican los datos correspondientes al nombre del asesor, el nombre del proyecto y el nombre de los desarrolladores del proyecto.

Si se da clic en el icono marcado podrá visualizarse el archivo correspondiente al proyecto terminal, este archivo corresponde a un documente con extensión PDF.

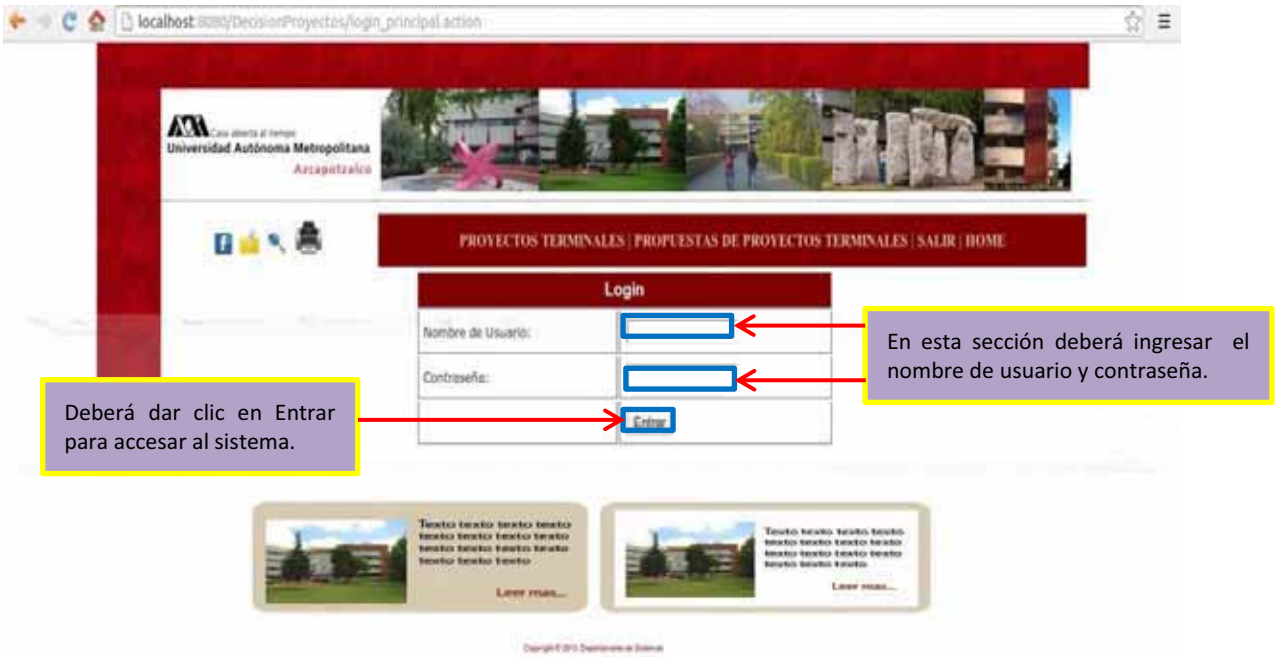
Así mismo podrá realizar las consultas de propuestas de proyectos terminales que los profesores han registrado en el sistema:



Para poder realizar una consulta a cualquier propuesta de proyecto terminal se deberá dar clic en el icono representado con una lupa tal y como se muestra en la imagen siguiente:

Como se puede observar en la lista de propuestas de proyectos terminales se muestran los datos correspondientes al nombre de la propuesta y se muestra la imagen correspondiente al link de visualización del archivo, este archivo corresponde a un PDF.

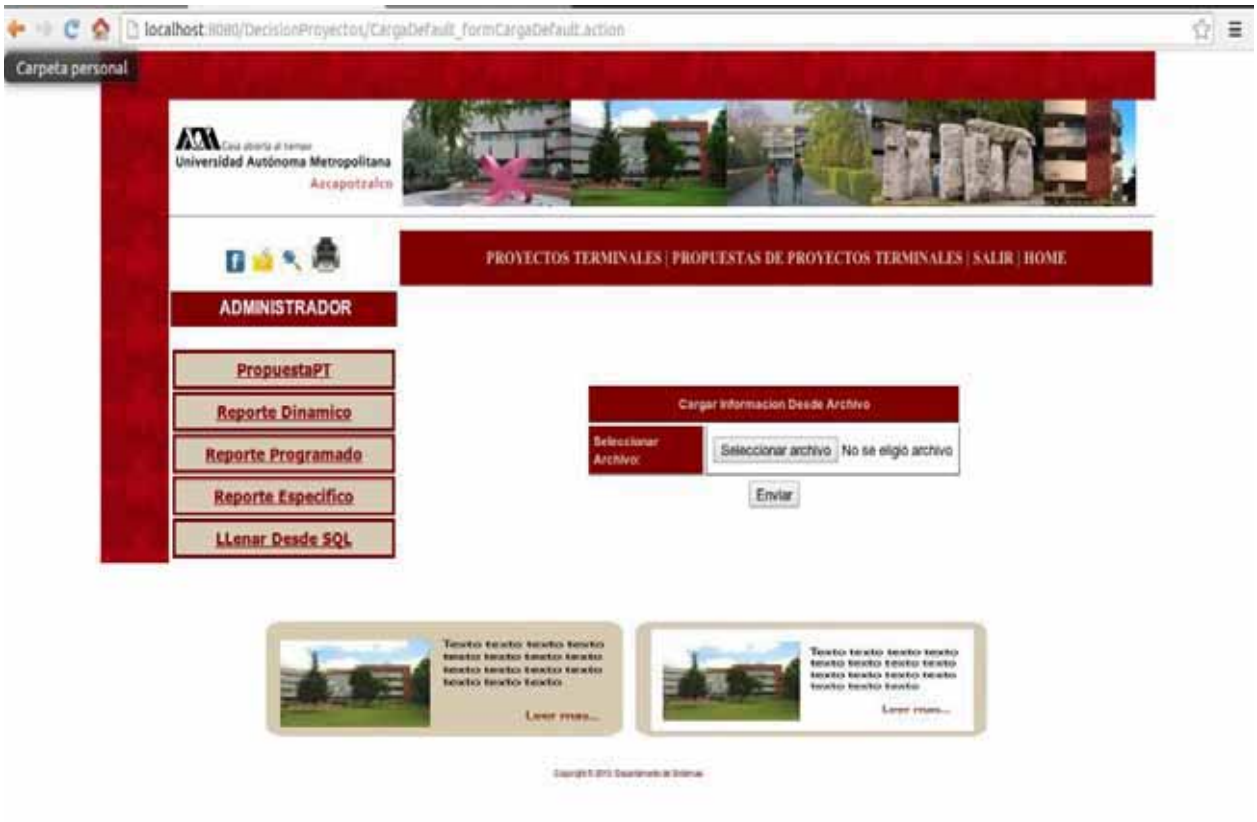
Si el usuario se encuentra registrado podrá accesar al sistema y hacer uso del mismo ingresando su usuario y contraseña correspondiente, así como también deberá dar clic en el botón entrar, tal y como se muestra en la siguiente imagen:



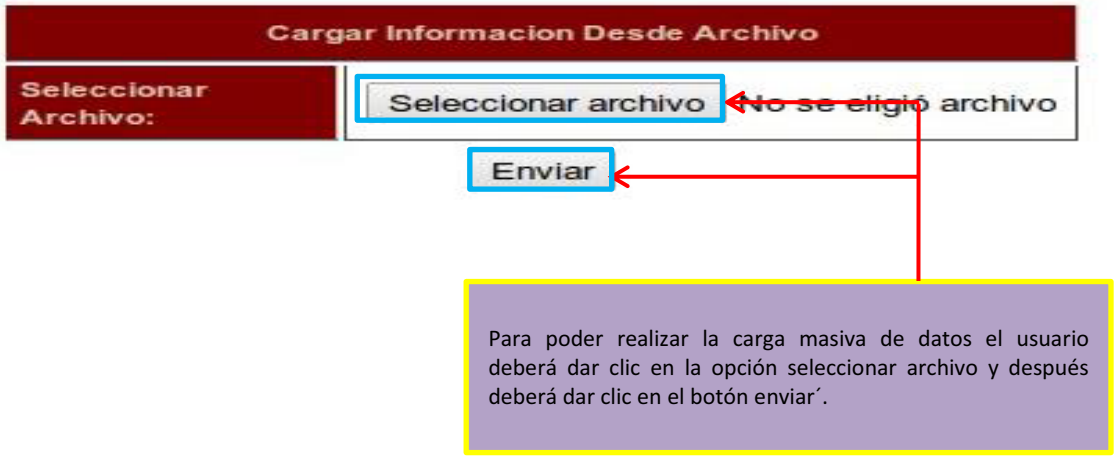
Sí el usuario registrado es administrador el sistema mostrará la siguiente pantalla:

Como podemos observar el usuario administrador cuenta con todos los privilegios sobre el sistema y podrá realizar cualquier tipo operación que ofrece el mismo.

Tal y como se muestra en la imagen anterior el usuario administrador puede seleccionar cualquiera de las opciones presentadas por el sistema, para poder agregar más datos a la Base de Datos el usuario deberá seleccionar la opción LlenarDesdeSQL y por consiguiente la pantalla que mostrará el sistema será la siguiente:



Como se observa en la imagen anterior el usuario solo deberá seleccionar el archivo correspondiente a los datos para el llenado de la BD desde una ruta alterna en su equipo.



El archivo seleccionado para el llenado de la BD corresponderá a una extensión SQL y deberá ser llenada como se muestra en la imagen siguiente.

```
scriptZ.sql
/*UNIDAD*/
unidad>1/Azcapotzalco/Paloma Jimenez Rodriguez/Arturo Puerta/azc@uam.mx/5566778899
unidad>2/Iztapalapa/Gerardo Garcia Perez/Beatriz Olivares/izt@uam.mx/57876598
unidad>3/Cuajimalpa/Penelope Avilar Estrada/Miranda Galvez/cua@uam.mx/57652133
unidad>4/Xochimilco/Jaime Morales Flores/Florencia Garcia/Xoc@uam.mx/34568130
/*-----------------------------------------------------------------------------------------------------*/
/*DIVISION*/
division>1/CBI/1/Jorge Valdano Vazquez/Arturo Solis/cbi@azc.uam.mx/54676567
division>2/CSH/1/Blanca Rosa Pampin/Elena Fernandez Garcia/csh@azc.uam.mx/58547654
division>3/CyAD/1/Oscar Mederos Mesa/Mirtha Alarcon de Quesada/cyad@azc.uam.mx/58734123
division>4/CBI/2/Teresa Lara Junco/Juan Carlos Alfonso Fraga/cbi@izt.uam.mx/53456788
division>5/CSH/2/Maria Eugenia Garcia/Pedro Gonzalez Rivas/csh@izt.uam.mx/59651211
division>6/CyAD/2/Antonio Vega Perz/Neyda Garcia Campos/cyad@izt.uam.mx/58011234
division>7/CBI/3/Maira Mena Correa/Maria del Carmen Franco/cbi@cua.uam.mx/55634543
division>8/CSH/3/Maria del Carmen Zamora/Guillermo Leganoa Martinez/csh@cua.uam.mx/55421328
division>9/CyAD/3/Evelyn Martinez Mendoza/Tomas M. Gonzalez Zorrilla/cyad@cua.uam.mx/59034123
division>10/CBI/4/Maricela Reina/Ileana Feliciano/cbi@xoc.uam.mx/58044512
division>11/CSH/4/Anabel Osorio/Amada Iglesias/csh@xoc.uam.mx/56780923
division>12/CyAD/4/Sissi Zamora/Cecilia Calderon/cyad@xoc.uam.mx/58769065
/*-----------------------------------------------------------------------------------------------------*/
/*DEPARTAMENTO*/
departamento>1/Sistemas/1
departamento>2/Derecho/2
departamento>3/Diseno Grafico/3
departamento>4/Mecanica/4
departamento>5/Administracion/5
departamento>6/Arquitectura/6
departamento>7/Industrial/7
departamento>8/Sociologia/8
departamento>9/Diseno Industrial/9
departamento>10/Fisica/10
departamento>11/Economia/11
departamento>12/Diseno Mecanica/12
/*-----------------------------------------------------------------------------------------------------*/
/*ROLES*/
rol>2/2/Profesor
rol>3/3/Alumno
rol>4/1/Jefe de departamento
/*-----------------------------------------------------------------------------------------------------*/
/*USUARIO*/
usuario>2/3/Alumno/al/al/55456787/usuario@hotmail.com
usuario>3/2/Rafaela Blanca Silva Lopez/blanca/blanca/67678654/blanca@azc.uam.mx
usuario>4/1/Administrador2/admin2/admin2/55456787/admin2@hotmail.com
usuario>5/2/Hugo Pablo Leyva/hugo/hugo/56578987/hugo@hotmail.com
usuario>6/3/Deyvid Cruz Perez/deyvid/deyvid/56787687/deyvid@hotmail.com
usuario>7/4/Jefe de Departamento/jefe/jefe/58765678/jefe@hotmail.com
usuario>8/1/Administrador3/admin3/admin3/58987890/admin3@hotmail.com
usuario>10/3/Juan Hernandez Pedocles/juan/juan/56765450/juan@hotmail.com
usuario>11/3/Jaime Ulises Gutierrez Flores/ulises/ulises/5514362041/ulises_830188@hotmail.com

/*-----------------------------------------------------------------------------------------------------*/
/*AREA DE DESARROLLO*/
areadesarrollo>1/Sistenas/Sistemas/Procesar sistenas/Rafaela Blanca Silva Lopez/sistenas@uam.mx/66558765
areadesarrollo>2/Fibra Optica/Diseno de fibras opticas/Hacer fibras opticas/Antonio Lopez/foptica@uam.mx/56765450
areadesarrollo>3/Base de Datos/Soporte a BD/Mantener en buen estado la BD/Hugo Pablo Leyva/BD@uam.mx/56765678
/*-----------------------------------------------------------------------------------------------------*/
/*PROYECTO TERMINAL*/
/*-----------------------------------------------------------------------------------------------------*/
/*ALUMNO*/
alumno>1/2/0/Alumno/206787654
alumno>2/6/0/Deyvid Cruz Perez/207200035
alumno>3/10/0/Juan Hernandez Pedocles/207898765
alumno>4/11/0/Jaime Ulises Gutierrez Flores/207200132
```

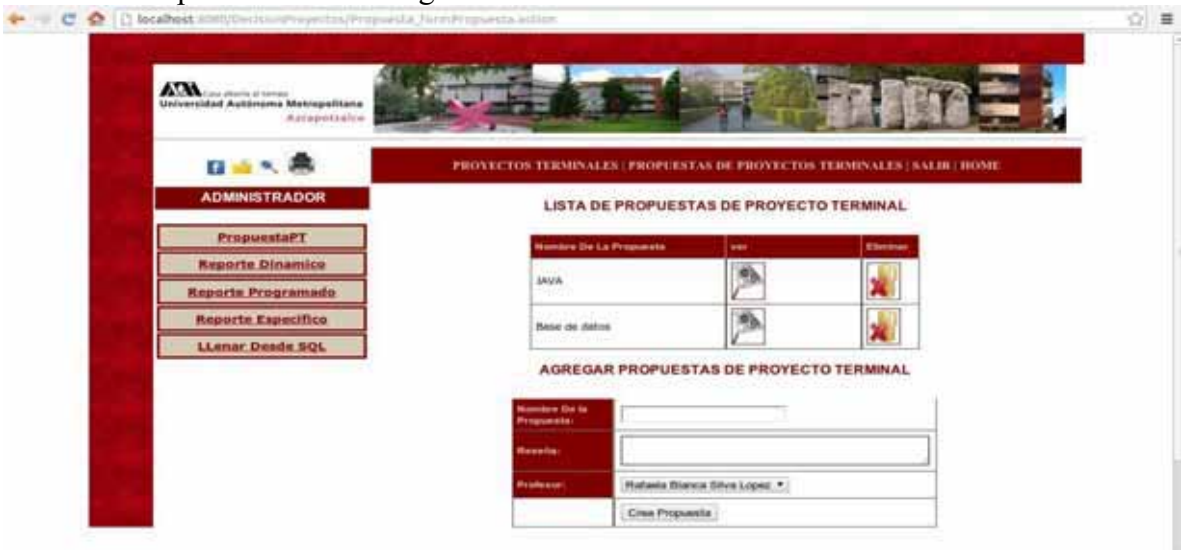
En caso de que este seleccione la opción PropuestaPT la pantalla que mostrara el sistema será la siguiente:

Como se puede observar el administrador deberá ingresar el Nombre de la Propuesta, una reseña correspondiente a dicha propuesta, así como también deberá seleccionar el nombre del profesor al cual corresponderá la propuesta. Por último deberá dar clic en el botón de crear propuesta con la finalidad de generar la propuesta requerida.

En el costado izquierdo de la pantalla se mantendrá en menú principal en caso de que el usuario desee realizar otra operación

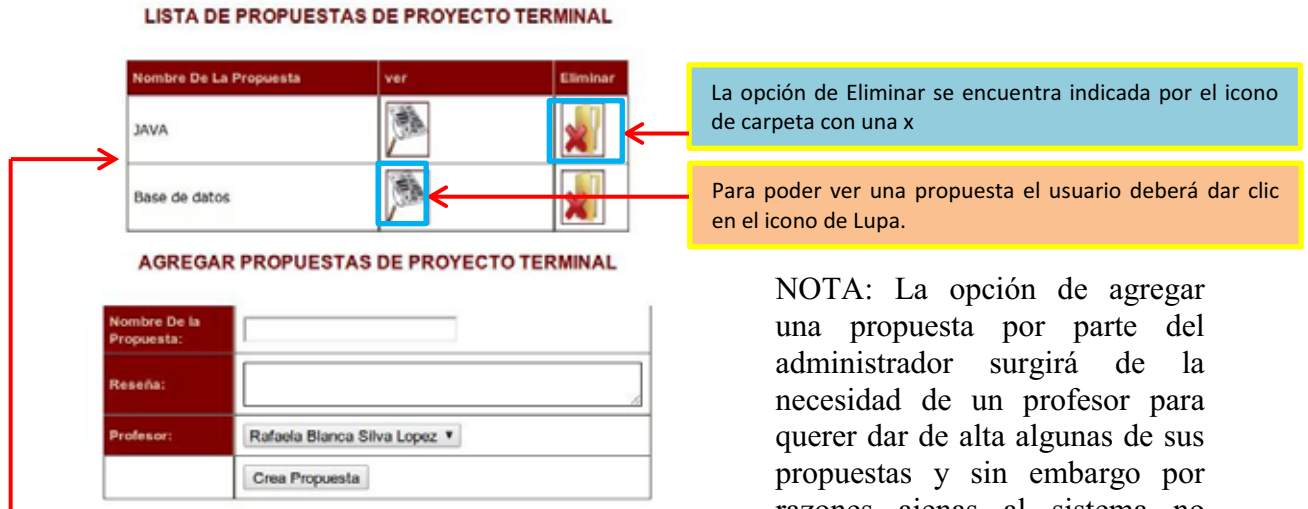
En la parte superior de la pantalla se mostrará la lista de las propuestas que se agregaran correctamente por el administrador.

Sí el usuario administrador ingresa de manera correcta la propuesta deseada, el sistema mostrará una pantalla como la siguiente:



Como se puede observar la lista de propuestas que fueron agregadas correctamente se muestran en la parte superior de la pantalla.

En caso de que el usuario Administrador quisiera agregar otra propuesta en el sistema deberá realizar los pasos anteriormente mencionados.



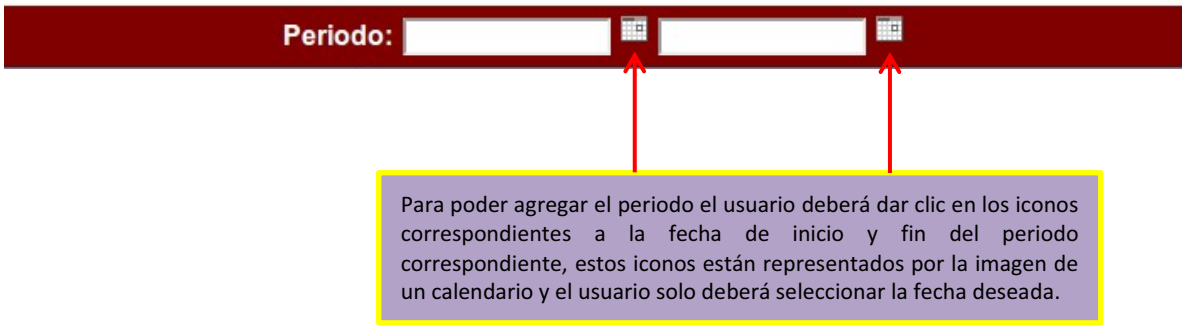
NOTA: La opción de agregar una propuesta por parte del administrador surgirá de la necesidad de un profesor para querer dar de alta algunas de sus propuestas y sin embargo por razones ajenas al sistema no podrá hacerlo, por lo cual acudirá al administrador para que este de alta dicha propuesta en su lugar, cabe mencionar que la propuesta quedará agregada al sistema de tal forma que parezca que fue el profesor titular de la propuesta quien la agregó y no el administrador.

La tabla correspondiente a la lista de Propuestas de Proyectos Terminales muestra el nombre de la Propuesta, la opción para poder ver la propuesta (Icono de Lupa) y la opción de Eliminar, esto se debe a los privilegios correspondientes a un usuario Administrador.

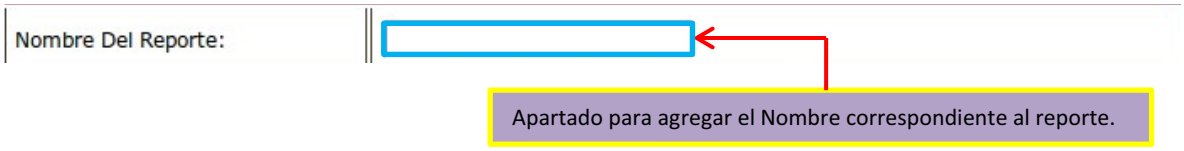
Sí el usuario selecciona la opción de gestionar un reporte dinámico deberá dar clic en el link ReporteDinamico mostrándose la siguiente pantalla:

Para llevar a cabo la correcta gestión de un reporte dinámico se proseguirá de la siguiente manera:

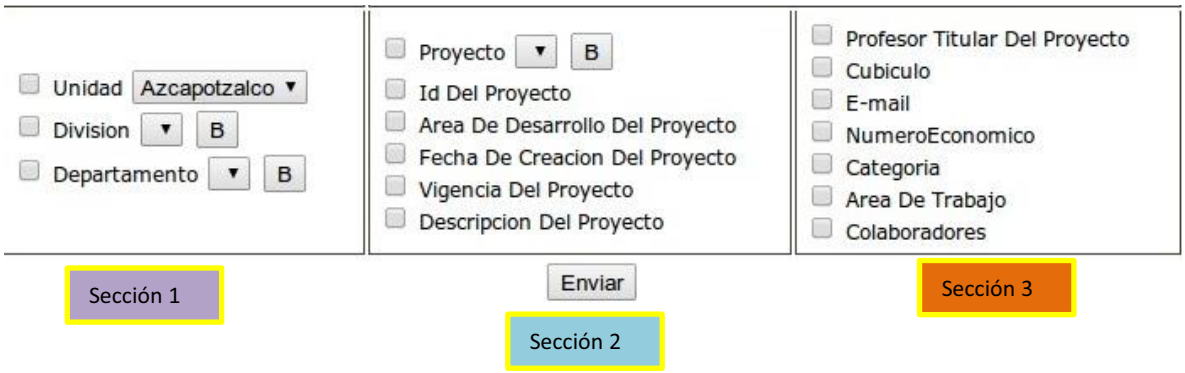
1) Se agregará el periodo correspondiente al reporte dinámico en el apartado correspondiente, tal y como se muestra en la imagen siguiente:



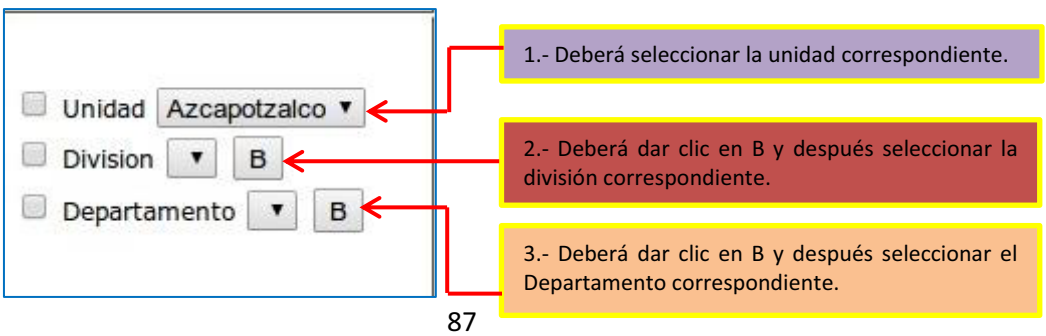
2) Por consiguiente se deberá agregar el nombre correspondiente al reporte dinámico:



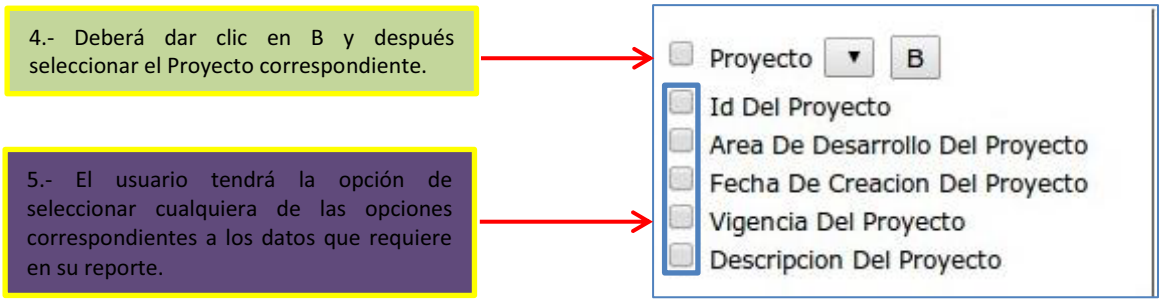
3) El siguiente paso consiste en seleccionar los datos que integrarán el reporte dinámico:



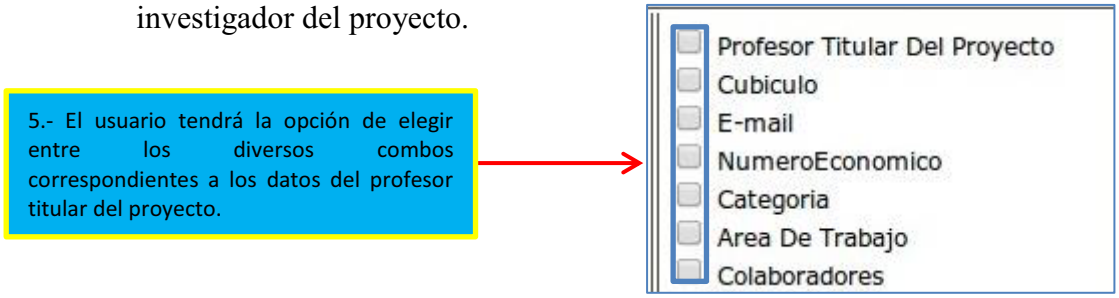
1) El usuario deberá seleccionar en primer lugar el combo correspondiente a la unidad.
2) Después deberá dar clic en el botón B para poder actualizar los datos correspondientes a la división perteneciente a la unidad seleccionada en el paso 3.1
3) Por consiguiente deberá realizar el mismo procedimiento para seleccionar el departamento perteneciente a la división seleccionada en el paso 3.2



4) En la siguiente sección el usuario deberá seleccionar los combos correspondientes a los datos que requiere en el Reporte Dinámico, para ello deberá seleccionar en primer lugar el Reporte correspondiente al Departamento seleccionado pues este será el dato preponderante del cual dependerán los datos de la sección 2 y 3 de la gestión del Reporte Dinámico, esta sección corresponde a los datos referentes al proyecto de investigación.



5) El usuario continuará seleccionando los datos que requiere para su reporte dinámico, esta sección corresponde a los datos referentes al profesor investigador del proyecto.



6) Una vez realizado esto el usuario deberá dar clic en el botón Enviar

Como es posible observar una vez que un reporte dinámico ha sido agregado de manera correcta la lista de reportes aparecerá en la parte superior de la pantalla tal y como se muestra la siguiente imagen:

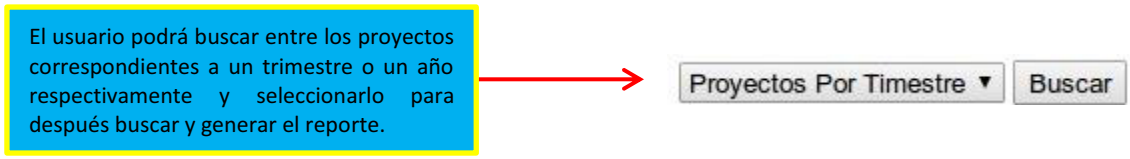
Una vez agregado un Reporte Dinámico, el usuario podrá visualizar el reporte dando clic en la imagen correspondiente y el reporte obtenido se mostrará como en la imagen siguiente:

Para dejar de ver el reporte el usuario puede dar clic en HOME colocado en la Barra roja ubicada en la parte superior de la pantalla o si desea hacer otra gestión puede dar clic en cualquiera de las opciones que ofrece el sistema ubicadas al costado izquierdo de la misma.

El reporte dinámico se muestra en un PDF donde el usuario tiene la opción de guardar el PDF en cualquier localidad de su equipo si así lo desea e imprimir el reporte.

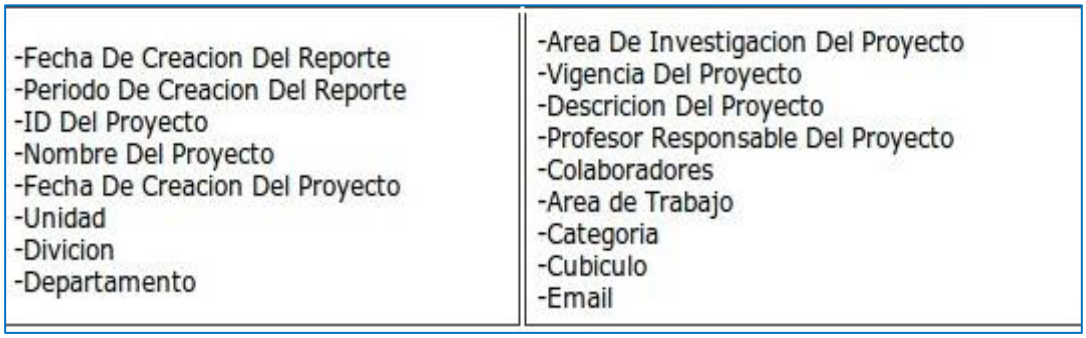
Sí el usuario selecciona la opción de gestionar un Reporte Programado deberá dar clic en el link Reporte Programado mostrándose la siguiente pantalla:

El reporte Programado consiste en dos tipos de reportes; un reporte por trimestre y un reporte anual, en la parte superior de la imagen anterior el usuario podrá buscar entre ambos tipos de reportes, para ello deberá elegir el tipo de reporte que desea buscar y después solo deberá dar clic en Buscar . Es importante mencionar que los reportes dependen de los proyectos es por ello que un reporte por trimestre al igual que un reporte anual se deberá elaborar de acuerdo a los proyectos correspondientes a ese trimestre o a ese año respectivamente.

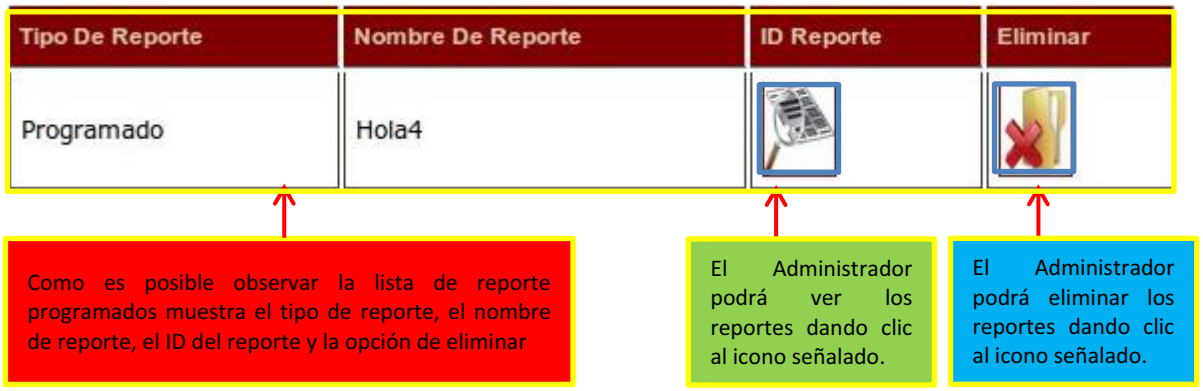


Una vez seleccionado que tipo de proyecto (Por Trimestre o Anual) el usuario podrá generar el Reporte Programado tal y como se muestra en la imagen:

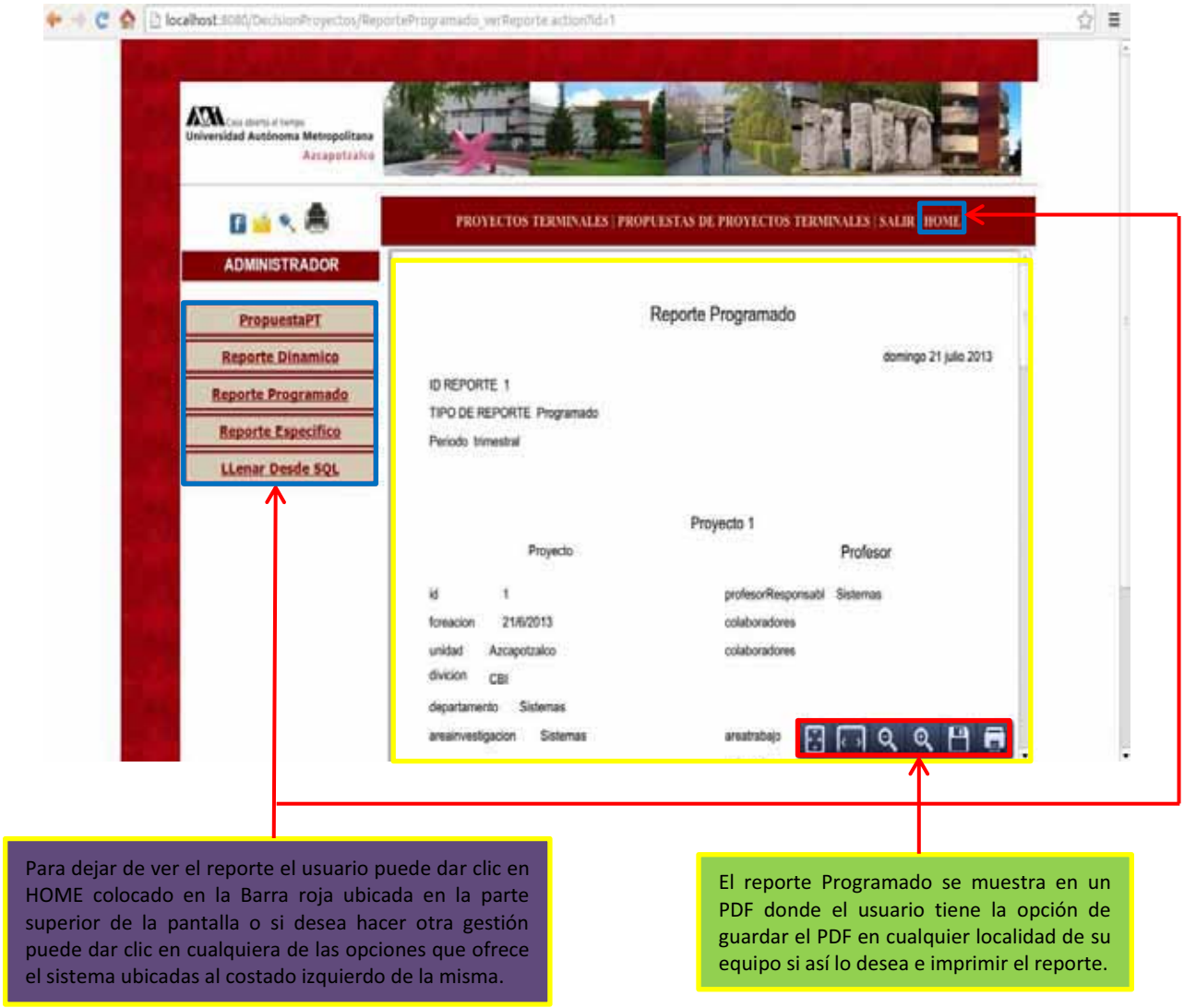
En el Reporte Programado los datos que se incluyen en él se encuentras establecidos por default por el sistema, por lo cual el usuario administrador no tendrá que seleccionar que datos se incluyen en el reporte, dicho usuario solo deberá elegir el tipo de proyecto que integrará el reporte, el proyecto en sí y escribir el nombre del reporte.



Sí el usuario genera de manera correcta el Reporte Programado, dicho reporte se agregará a la lista de reportes tal y como se muestra en la siguiente imagen.



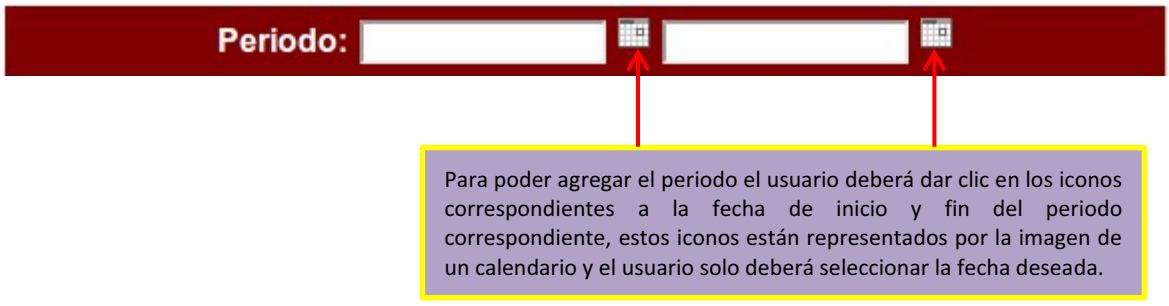
El reporte generado será un archivo con extensión PDF y se mostrará como en la siguiente imagen:



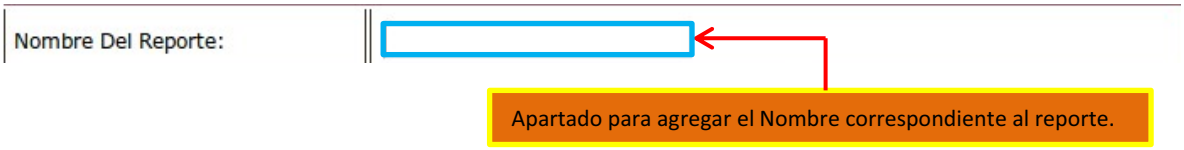
Sí el usuario selecciona la opción de gestionar un Reporte Específico deberá dar clic en el link Reporte Específico mostrándose la siguiente pantalla:

Para poder generar un Reporte Específico el usuario deberá seguir los siguientes pasos:

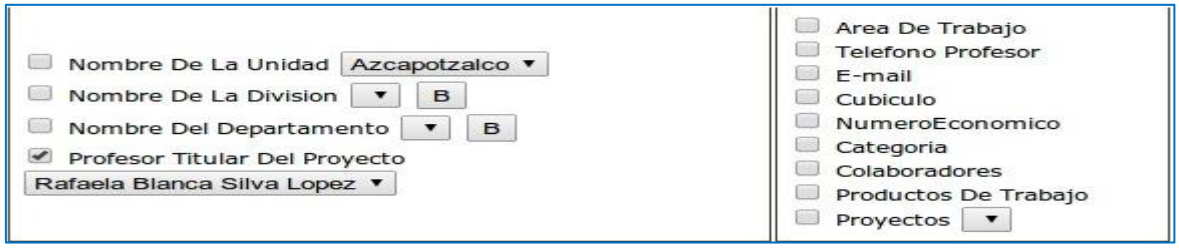
1) Deberá ingresar el periodo que abarcará el reporte:

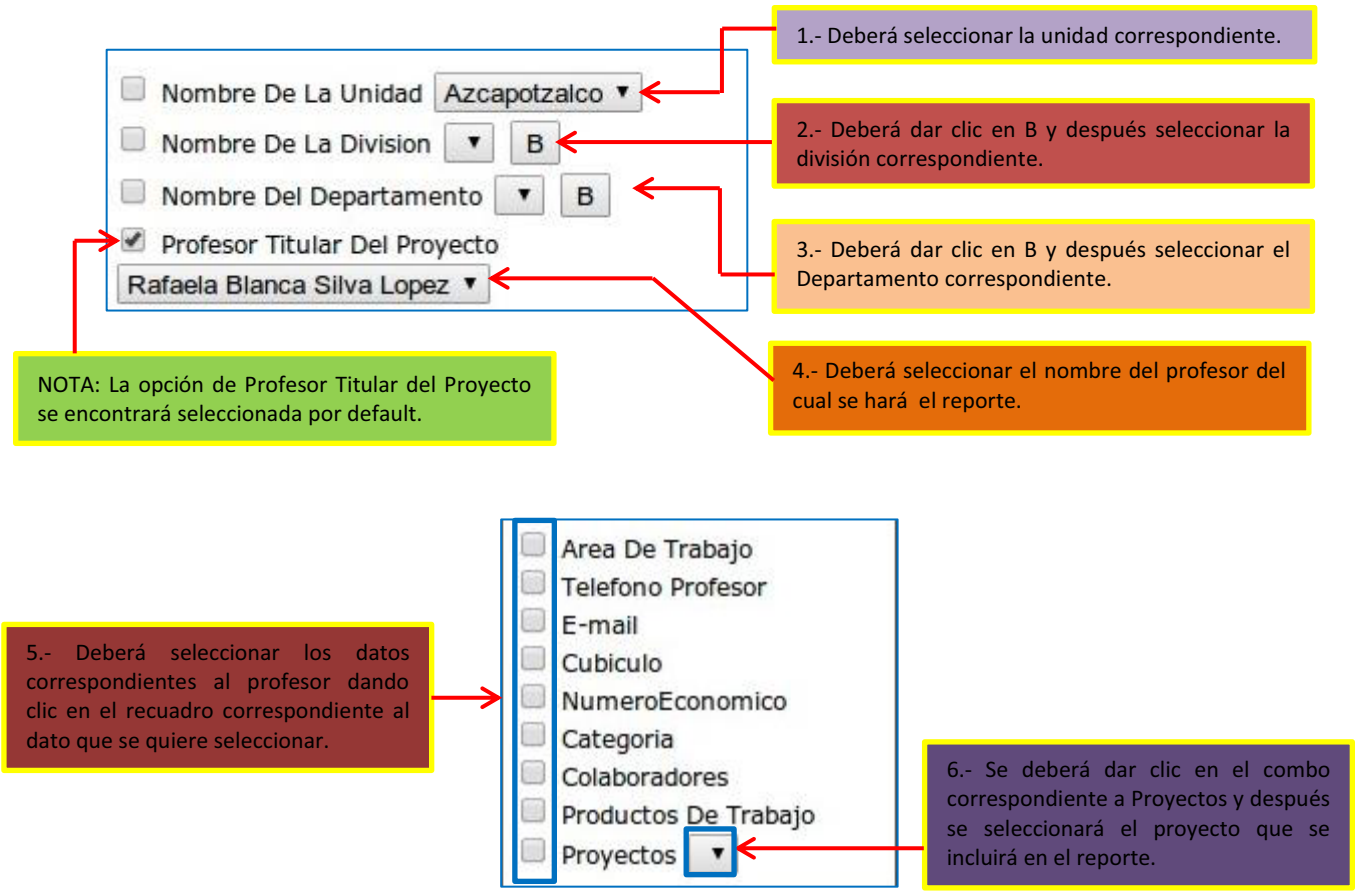


2) Deberá ingresar el nombre del reporte:



3) Por consiguiente el usuario deberá seleccionar los datos que se incluirán en el reporte.
   1) Se deberán seleccionar los datos de la sección correspondiente al profesor:



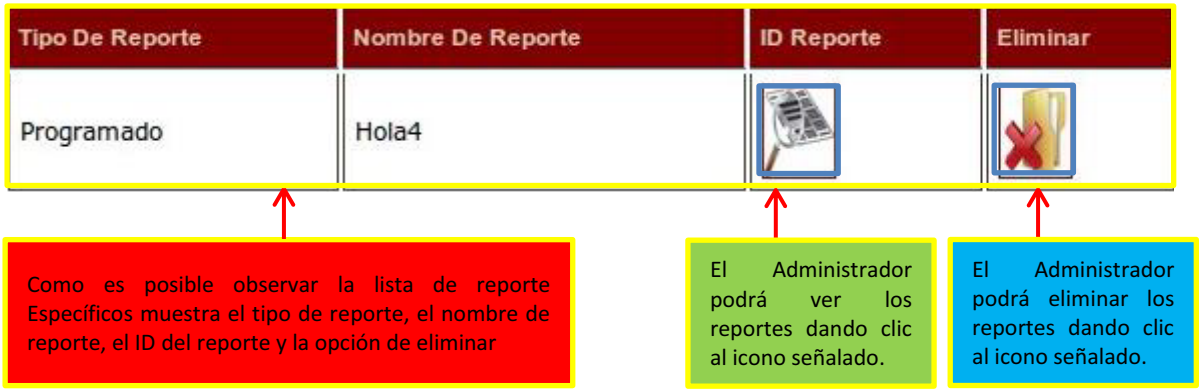


4) Se deberán seleccionar los datos correspondientes a la sección de temáticas.
   4.1 se deberá seleccionar la temática que se incluirá en el reporte.

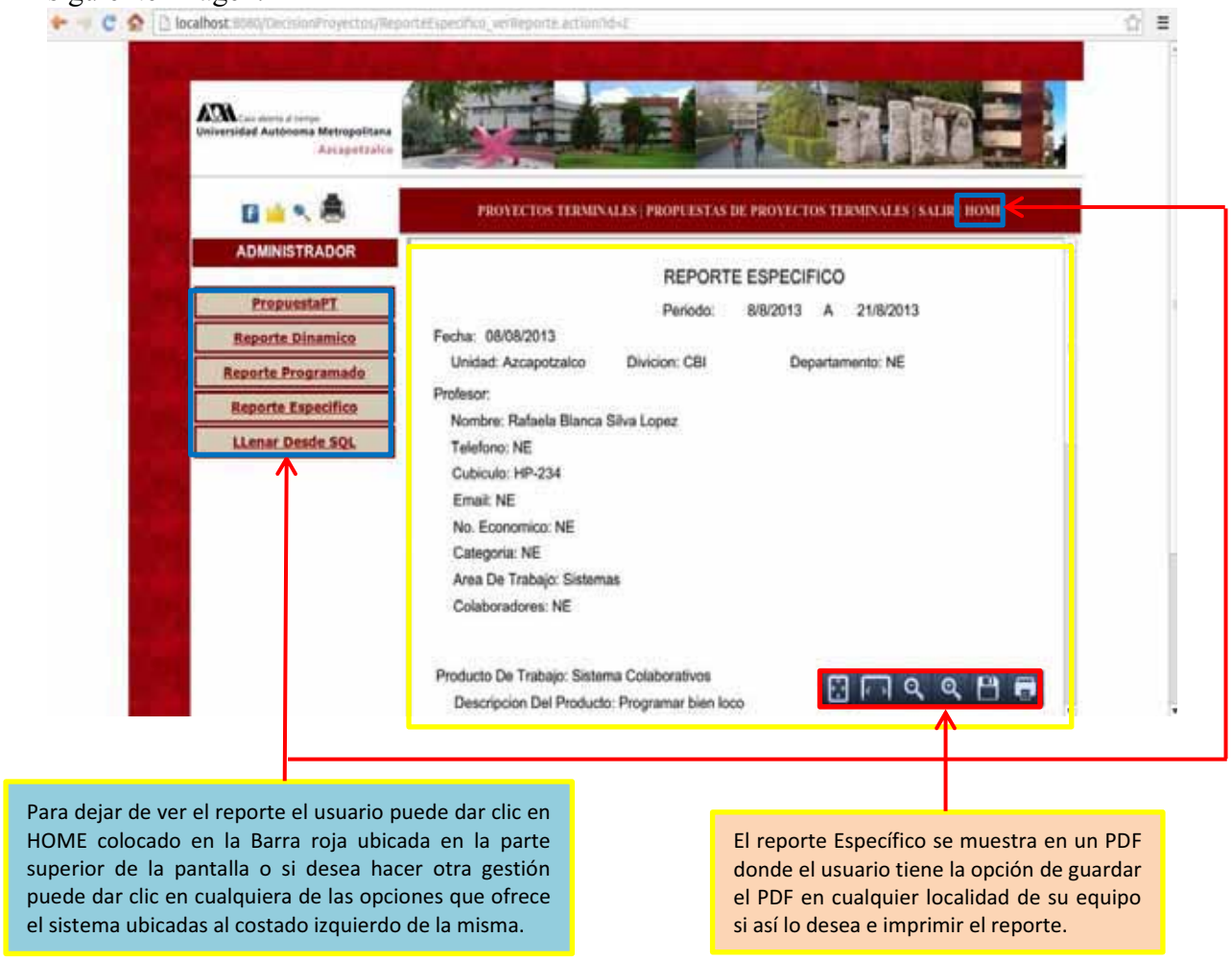
5) Se deberá seleccionar los datos correspondientes a la sección de Productos de Trabajo.

6) Por último el usuario deberá dar clic en Enviar para poder generar el reporte especifico.

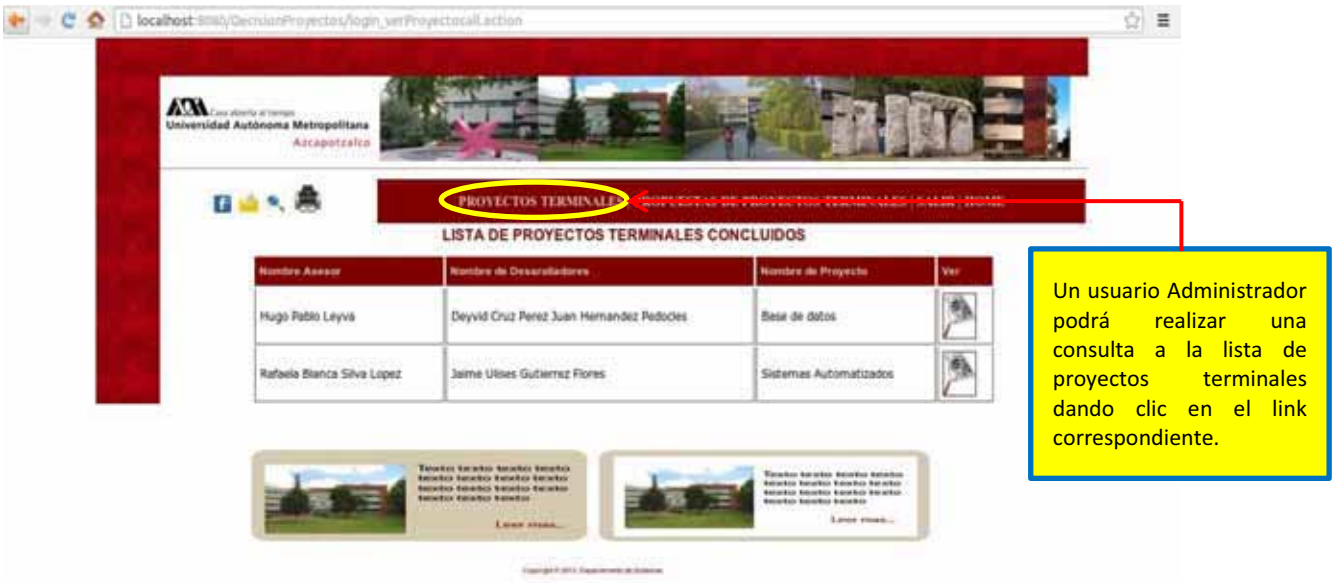
Sí el usuario genera de manera correcta el Reporte Específico, dicho reporte aparecerá en la lista de reportes que se encuentra en la parte superior de la pantalla, tal y como se muestra en la siguiente imagen:



El reporte generado será un archivo con extensión PDF tal y como se muestra en la siguiente imagen:



Es importante mencionar que el usuario administrador dentro de su sesión también podrá visualizar los proyectos terminales de la misma forma que un usuario no registrado, tal y como se muestra en la imagen siguiente:

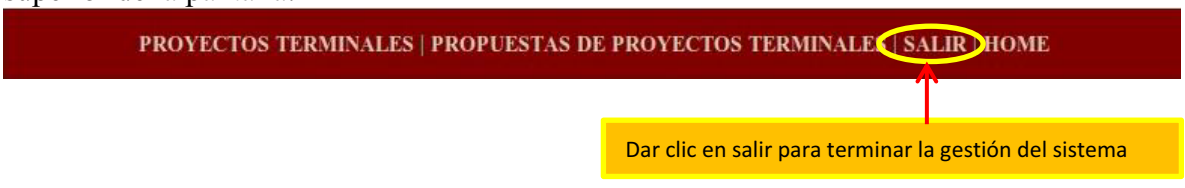


Para poder realizar una consulta a cualquier proyecto terminal se deberá dar clic en el icono representado con una lupa tal y como se muestra en la imagen siguiente:

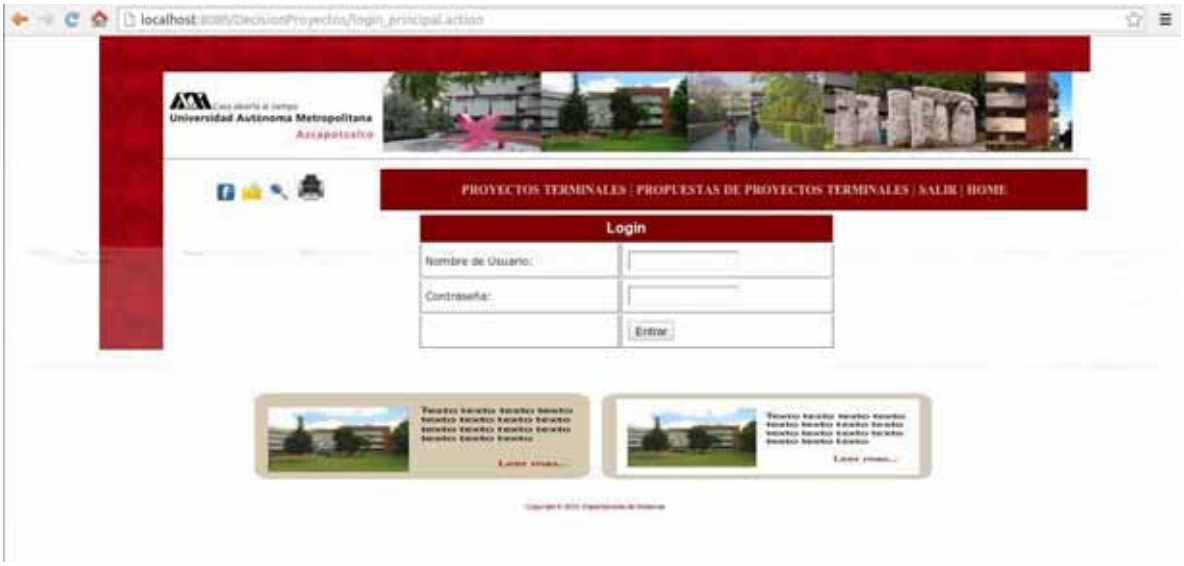
Como se puede observar en la lista de proyectos terminales se indican los datos correspondientes al nombre del asesor, el nombre del proyecto y el nombre de los desarrolladores del proyecto.

Si se da clic en el icono marcado podrá visualizarse el archivo correspondiente al proyecto terminal, este archivo corresponde a un documente con extensión PDF.

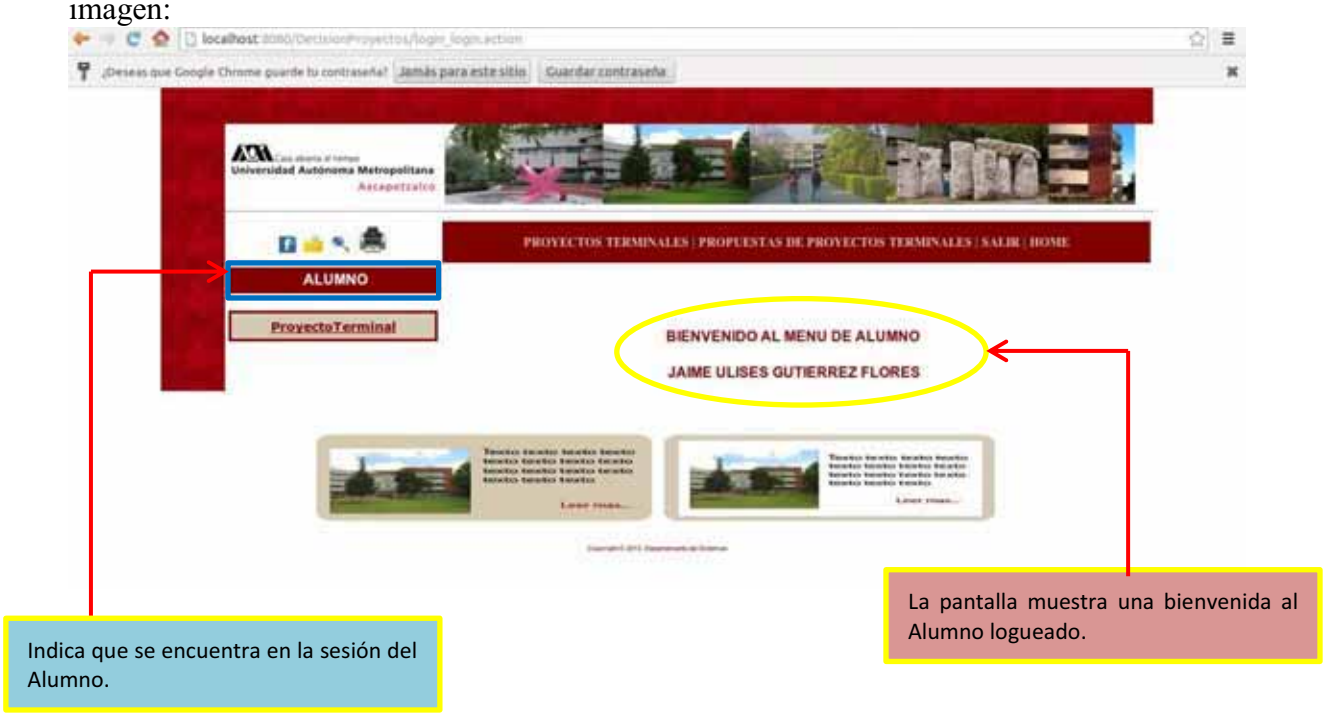
Sí el usuario Administrador ya no desea realizar ninguna operación en el sistema podrá salir dando clic en el link salir colocado en la Barra de color rojo que se encuentra en la parte superior de la pantalla:



Una vez terminada la sesión, el sistema mostrará la pantalla de inicio donde cualquier otro usuario podrá iniciar su sesión para gestionar el sistema.

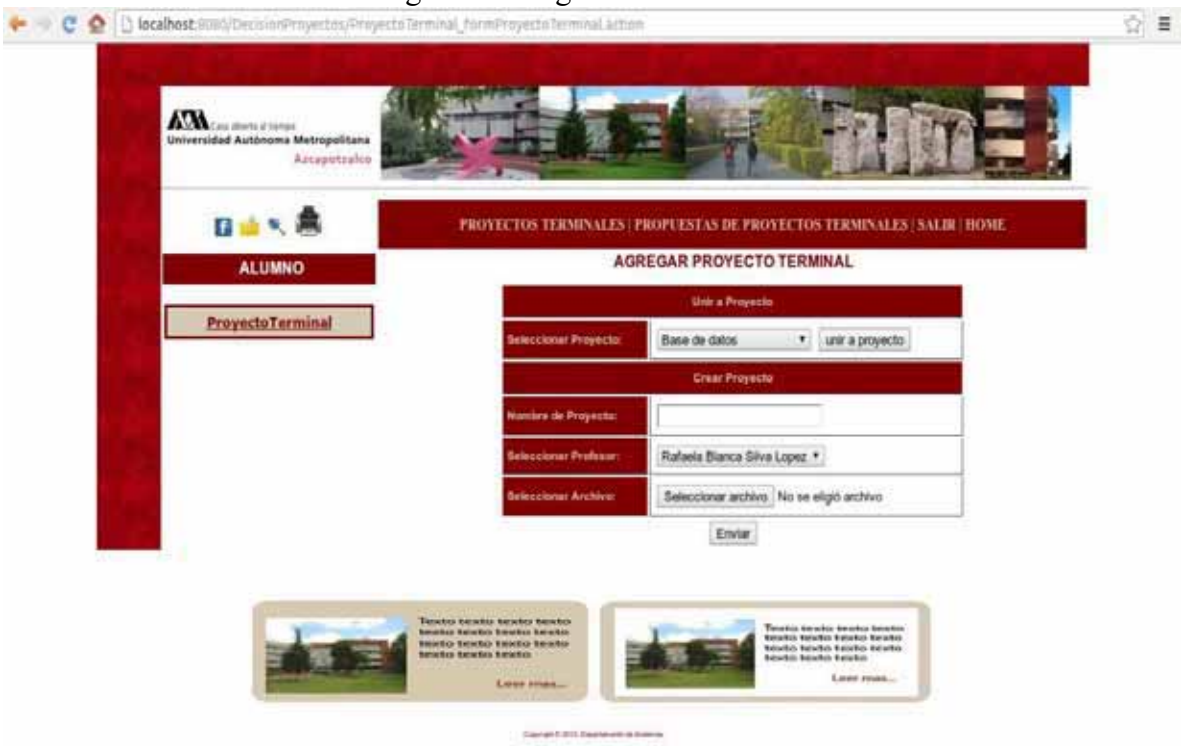


Sí el usuario es un Alumno; Una vez iniciada su sesión el sistema mostrará la siguiente imagen:



Como se muestra en la siguiente imagen el Alumno solo tiene la opción de ProyectoTerminal por lo cual el solo podrá gestionar los Proyectos Terminales de las que dispone para ofrecer a los usuarios interesados.

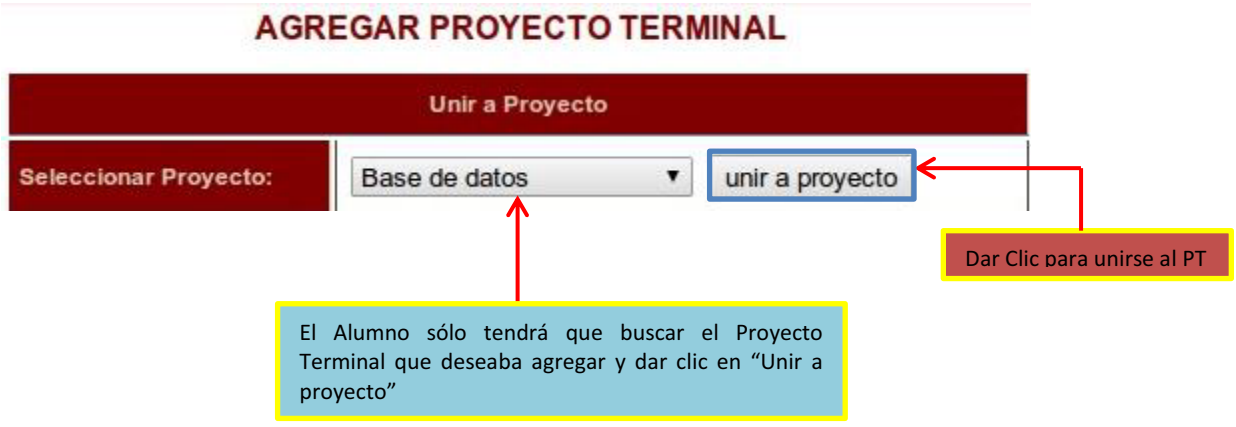
Sí el alumno desea dar de alta un Proyecto terminal deberá dar clic en ProyectoTerminal a lo cual el sistema mostrará la siguiente imagen:



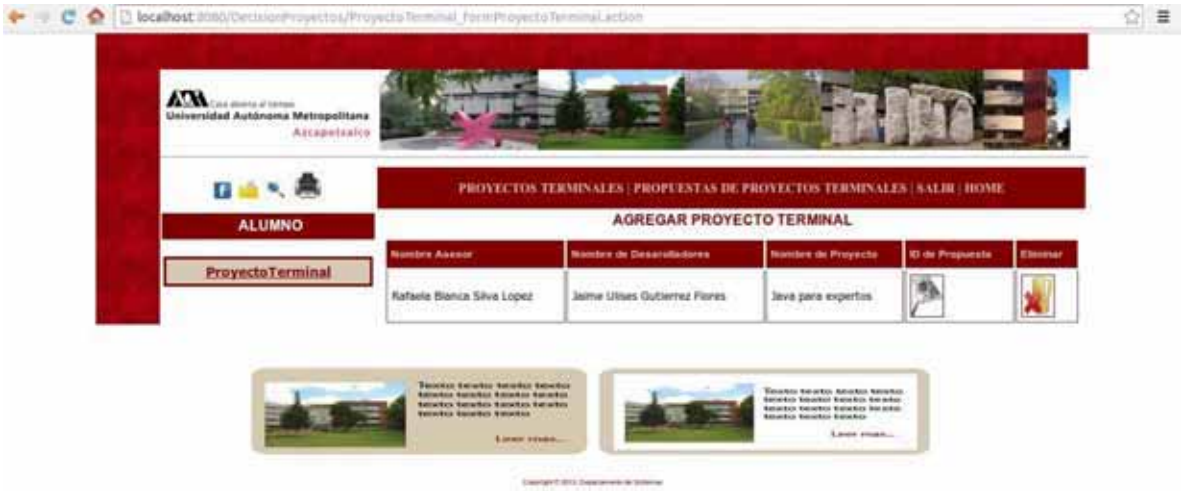
Para poder dar de alta un proyecto terminal existen 3 casos por lo cual el Alumno deberá hacer lo siguiente:

CASO 1: Este caso se presenta cuando más de un alumno está trabajando en un mismo proyecto, por lo cual cualquiera de los alumnos responsables del proyecto pudo haber dado de alta el proyecto con anterioridad, a lo que el actual Alumno que quiere dar de alta ese proyecto, este solo tendrá que unirse al proyecto pues ya fue agregado con anteriormente por su o sus compañeros de proyecto.

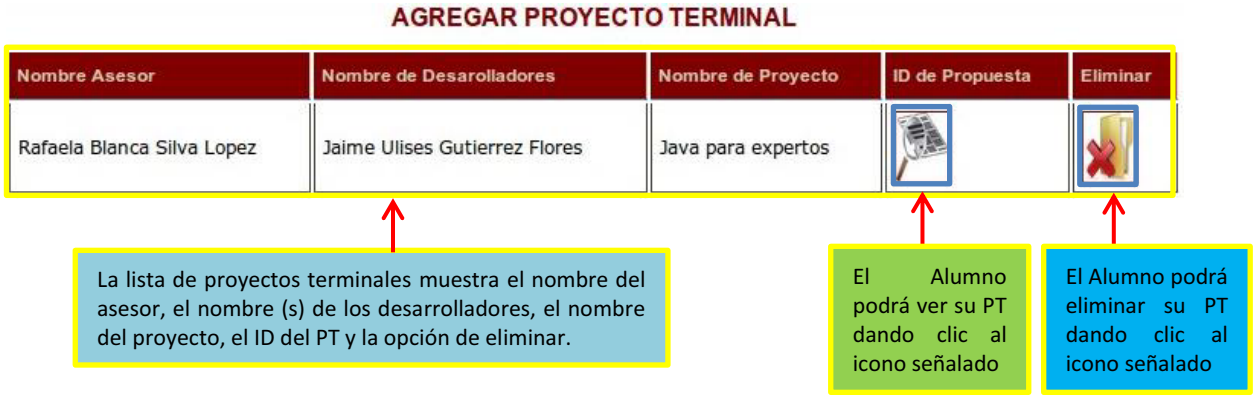
1) En caso de que el Proyecto Terminal que el alumno desea agregar ya exista en la Lista de proyectos terminales el alumno solo deberá unirse a un Proyecto ya existente, tal y como se muestra en la siguiente imagen:



2) Una vez que el Alumno de clic en unir a proyecto , dicho proyecto se agregará a la lista de proyectos terminales del alumno. Es importante mencionar que esta lista regularmente solo estará conformada por un solo proyecto terminal, pues un alumno solo puede trabajar en un proyecto terminal durante su carrera.



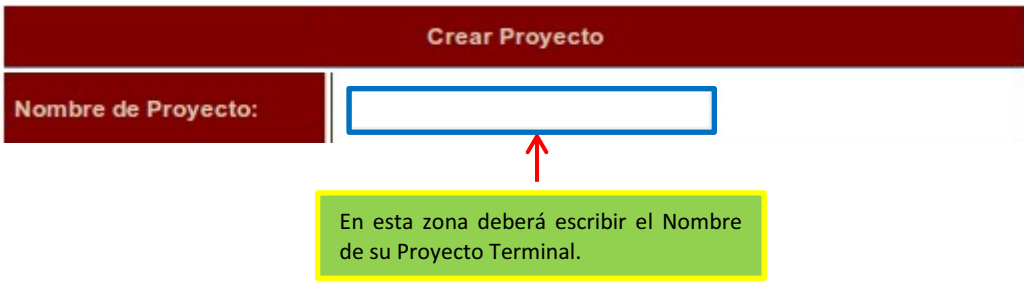
Como se muestra en la imagen anterior el alumno podrá ver el proyecto al cual se unió en la lista de proyectos, así como también podrá eliminarlo de su lista, cabe mencionar que esta eliminación solo es para el alumno que acaba de unirse al proyecto y no es una eliminación del proyecto en sí del sistema.



CASO 2: Este caso consiste en que un proyecto aún no ha sido de alta por ninguno de los alumnos responsables del proyecto, por lo cual no se encuentra en la lista de proyectos a unirse, a lo que el actual usuario del sistema deberá agregar el Proyecto Terminal por primera vez al sistema. NOTA: La pantalla que muestra el sistema en este caso es la misma que en el caso anterior:

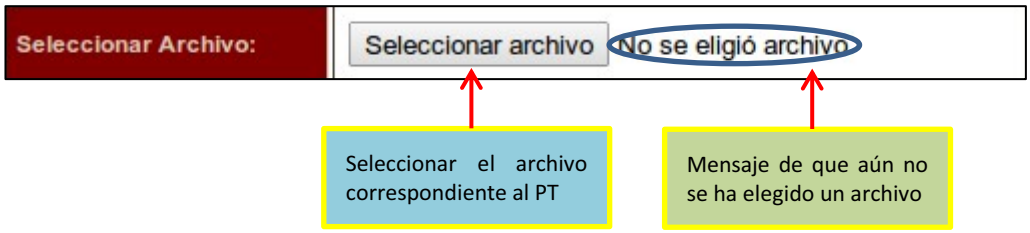
Para poder agregar un proyecto se debe proseguir de la siguiente manera:

1) El Alumno deberá agregar el nombre del proyecto que desea agregar, tal y como se muestra en la siguiente imagen:



2) El alumno deberá seleccionar de la Lista de Profesores al Profesor asesor del Proyecto Terminal, tal y como se muestra en la imagen siguiente:

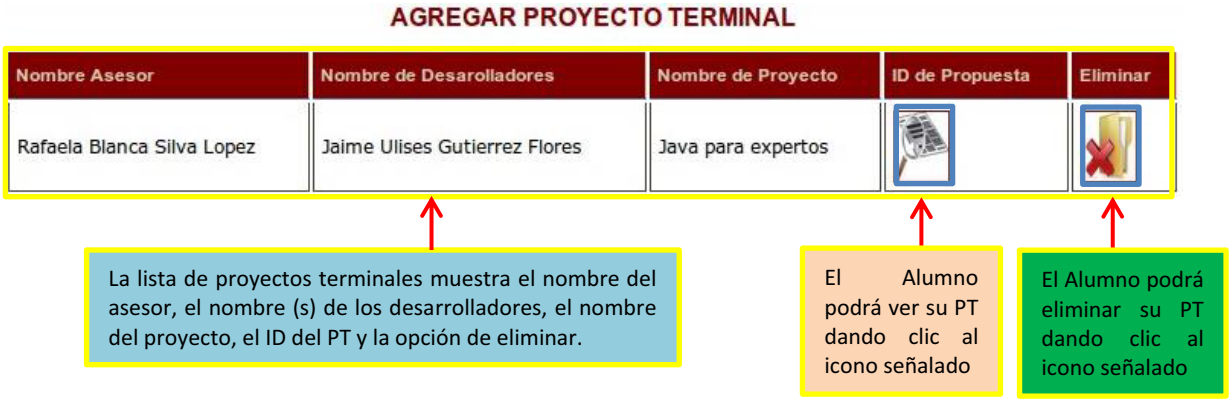
3) El siguiente paso consiste en seleccionar el archivo correspondiente al Proyecto Terminal que se desea cargar al sistema, la carga de este archivo se realiza desde cualquier localidad de su computadora tal y como se muestra a continuación:



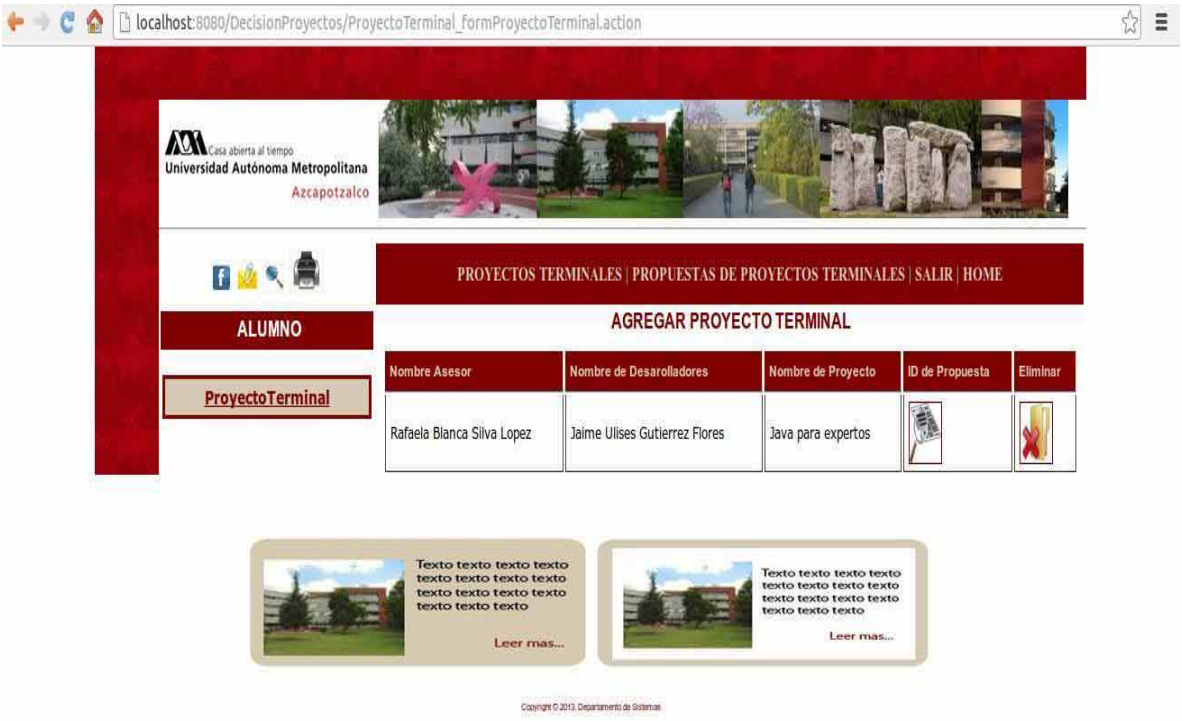
4) Por último el Alumno deberá dar clic en 

para con ello dar de alta su PT.

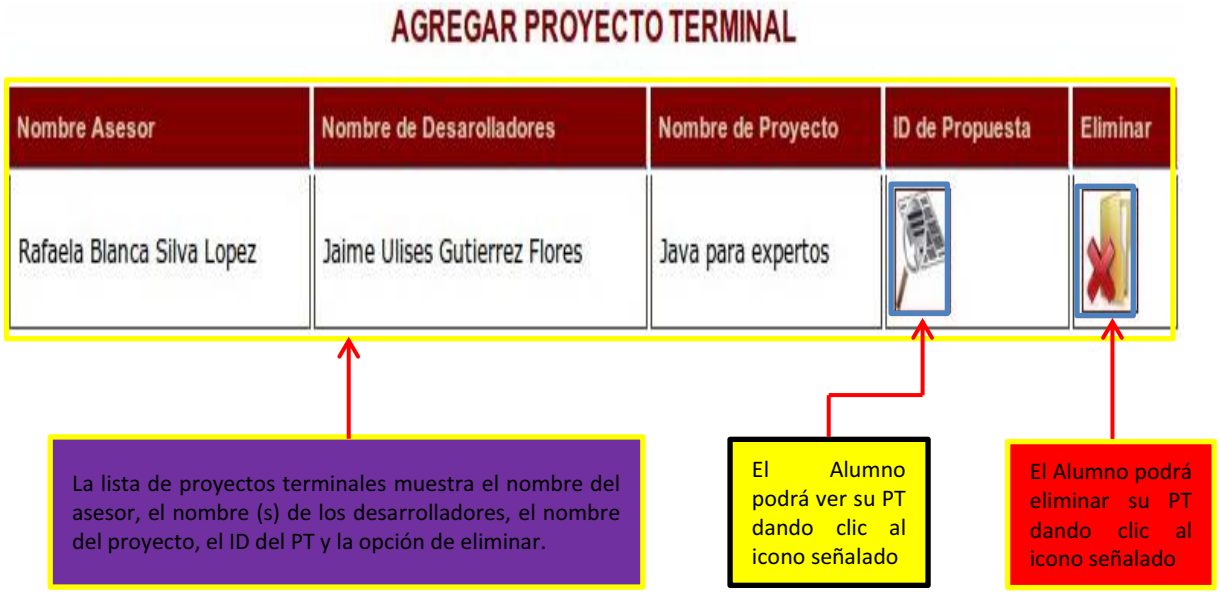
Una vez que el usuario agregue de manera exitosa el Proyecto Terminal deseado este podrá visualizar dicho proyecto tal y como se mostró en el CASO 1.



CASO 3: Este caso se refiera a cuando un alumno quiere dar de alta un proyecto terminal, sin embargo ya lo había agregado anteriormente, por lo cual ya no es posible agregar un nuevo proyecto, a lo que el sistema muestra la siguiente imagen:



Como muestra la imagen anterior una vez que un alumno ha agregado un proyecto lo único que puede hacer es ver o eliminar el proyecto de la lista de PT.

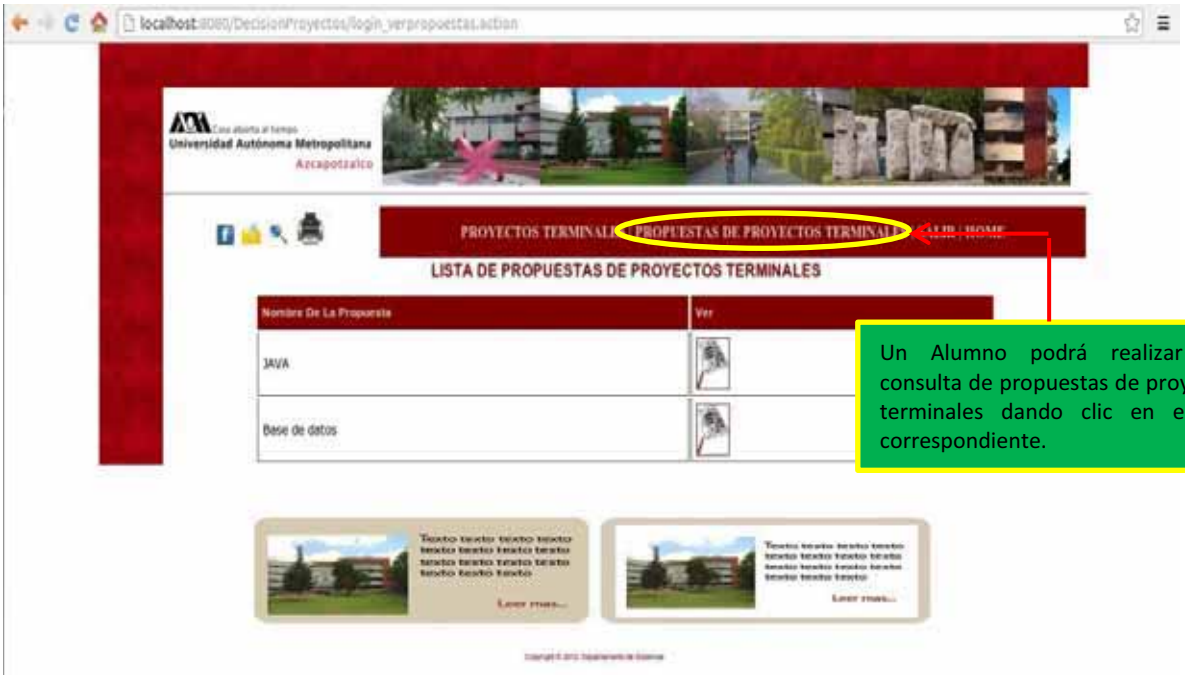


El Proyecto terminal cargado es un archivo PDF que se visualiza de la siguiente manera:

Para dejar de ver el Proyecto Terminal puede dar clic en HOME colocado en la Barra roja ubicada en la parte superior de la pantalla o si desea hacer otra gestión puede dar clic en la opción de ProyectoTerminal que ofrece el sistema ubicada al costado izquierdo de la misma.

El Proyecto Terminal se muestra en un PDF donde el usuario tiene la opción de guardar el PDF en cualquier localidad de su equipo si así lo desea e imprimir el PT.

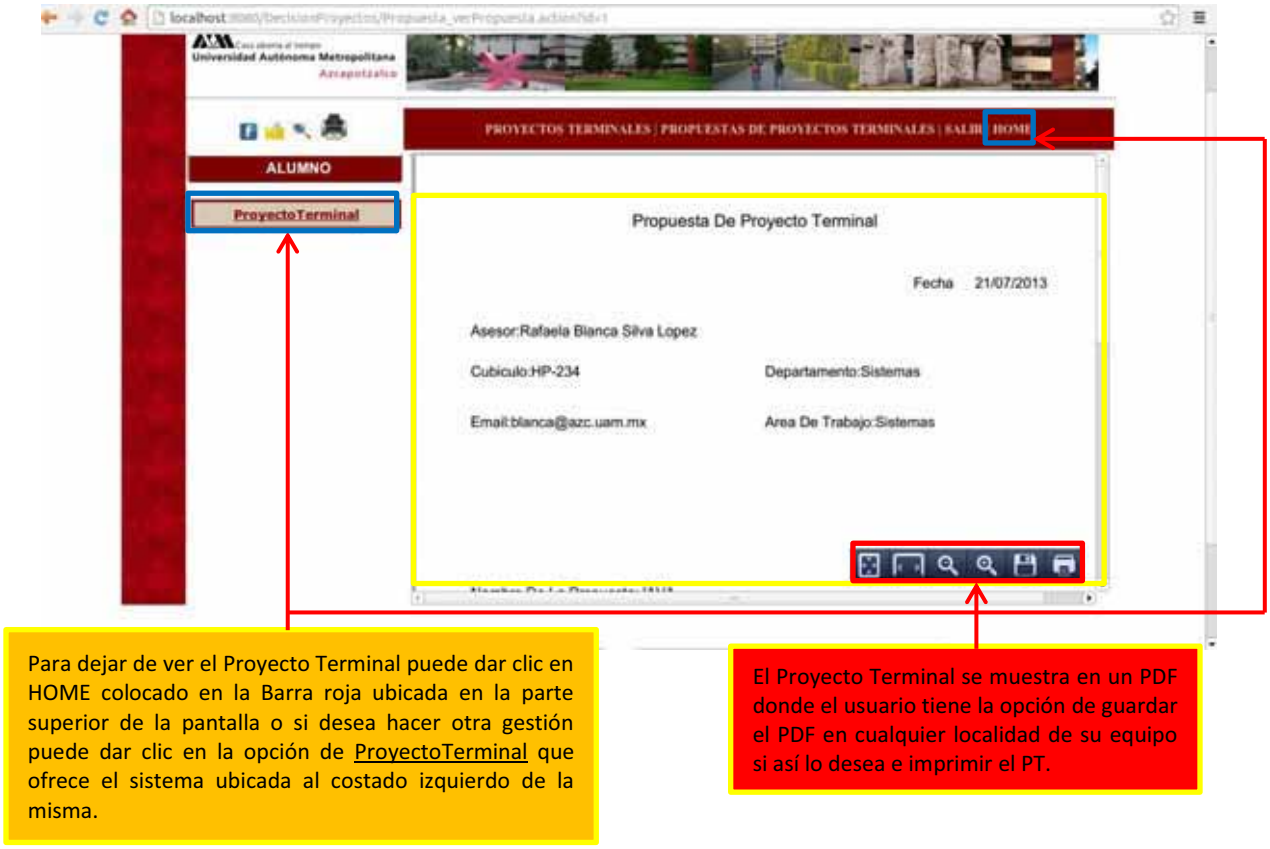
Es importante mencionar que el usuario Alumno dentro de su sesión también podrá visualizar las propuestas de proyectos terminales de la misma forma que un usuario no registrado, tal y como se muestra en la imagen siguiente:



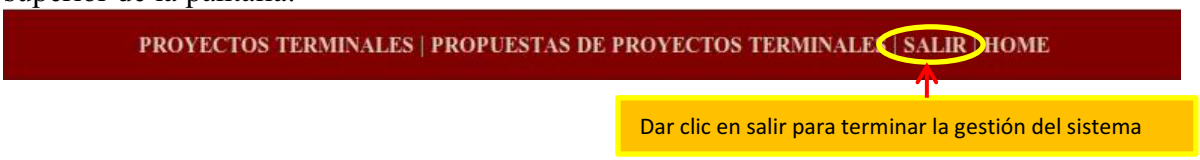
Un Alumno podrá realizar una consulta de propuestas de proyectos terminales dando clic en el link correspondiente.

Para poder realizar una consulta a cualquier propuesta de proyecto terminal se deberá dar clic en el icono representado con una lupa tal y como se muestra en la imagen siguiente:

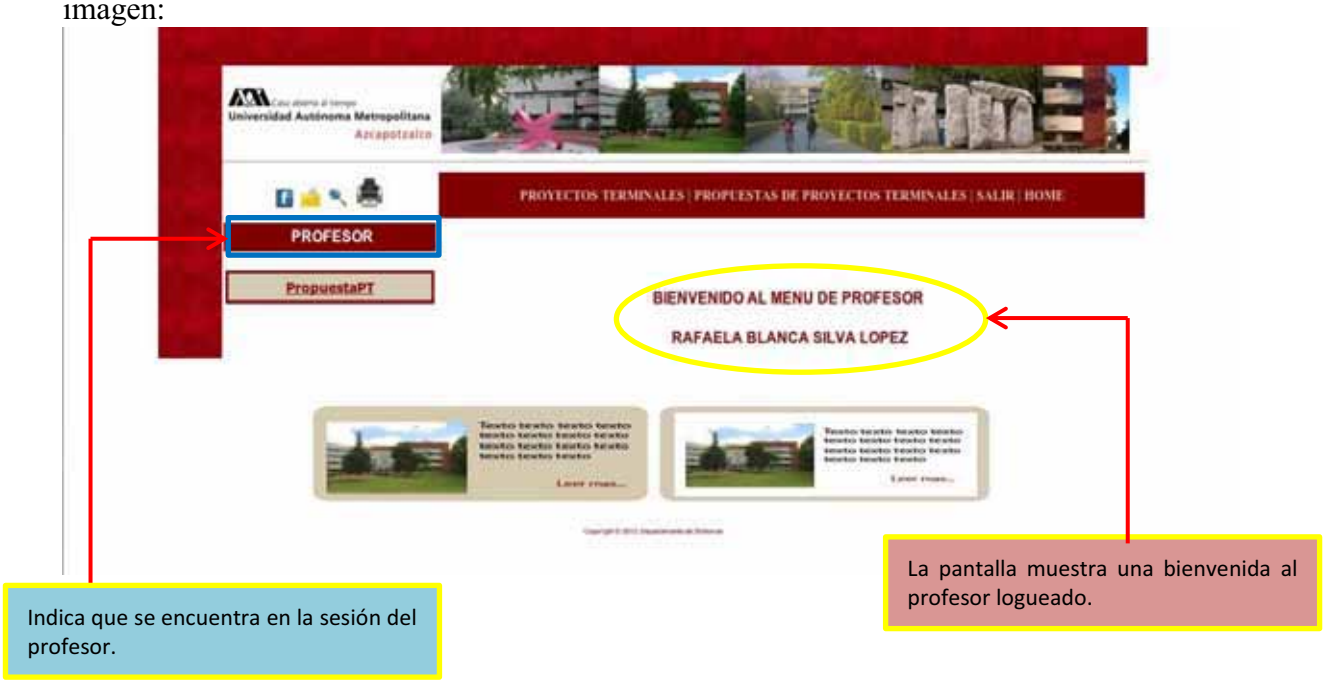
Como se puede observar en la lista de propuestas de proyectos terminales se muestran los datos correspondientes al nombre de la propuesta y se muestra la imagen correspondiente al link de visualización del archivo, este archivo corresponde a un PDF.



Sí el usuario Alumno ya no desea realizar ninguna operación en el sistema podrá salir dando clic en el link salir colocado en la Barra de color rojo que se encuentra en la parte superior de la pantalla:

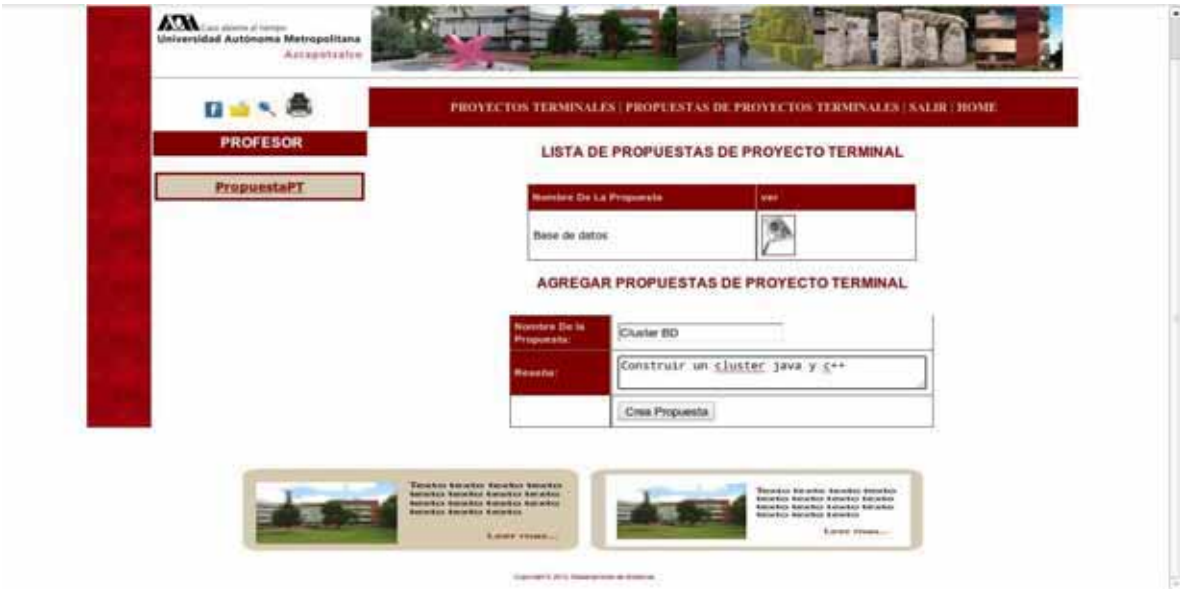


Sí el usuario es un Profesor; una vez iniciada su sesión el sistema mostrará la siguiente imagen:



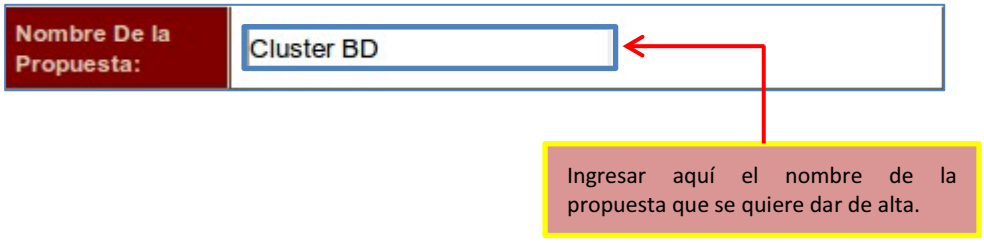
Como se muestra en la siguiente imagen el profesor solo tiene la opción de PorpuestaPT por lo cual el solo podrá gestionar las propuestas de Proyectos Terminales de las que dispone para ofrecer a los alumno.

Sí el usuario desea dar de alta una propuesta de proyecto terminal deberá dar cli en PropuestaPT a lo que el sistema mostrará la siguiente pantalla:

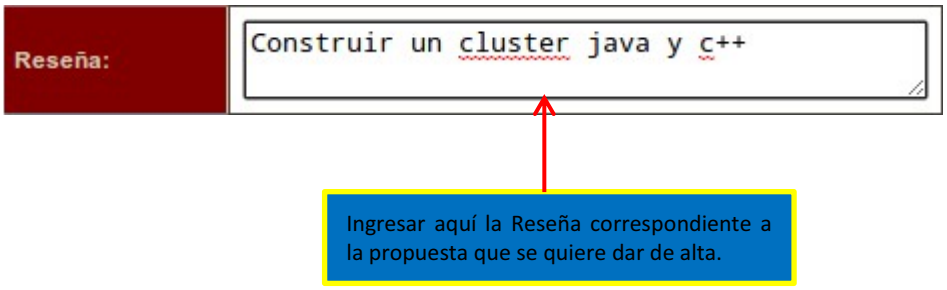


Para poder dar de alta una Propuesta de Proyecto de Terminal el Profesor deberá seguir los siguientes pasos:

1) Deberá ingresar el nombre correspondiente a la Propuesta del Proyecto Terminal que desea dar de alta, tal y como se muestra en la siguiente imagen:



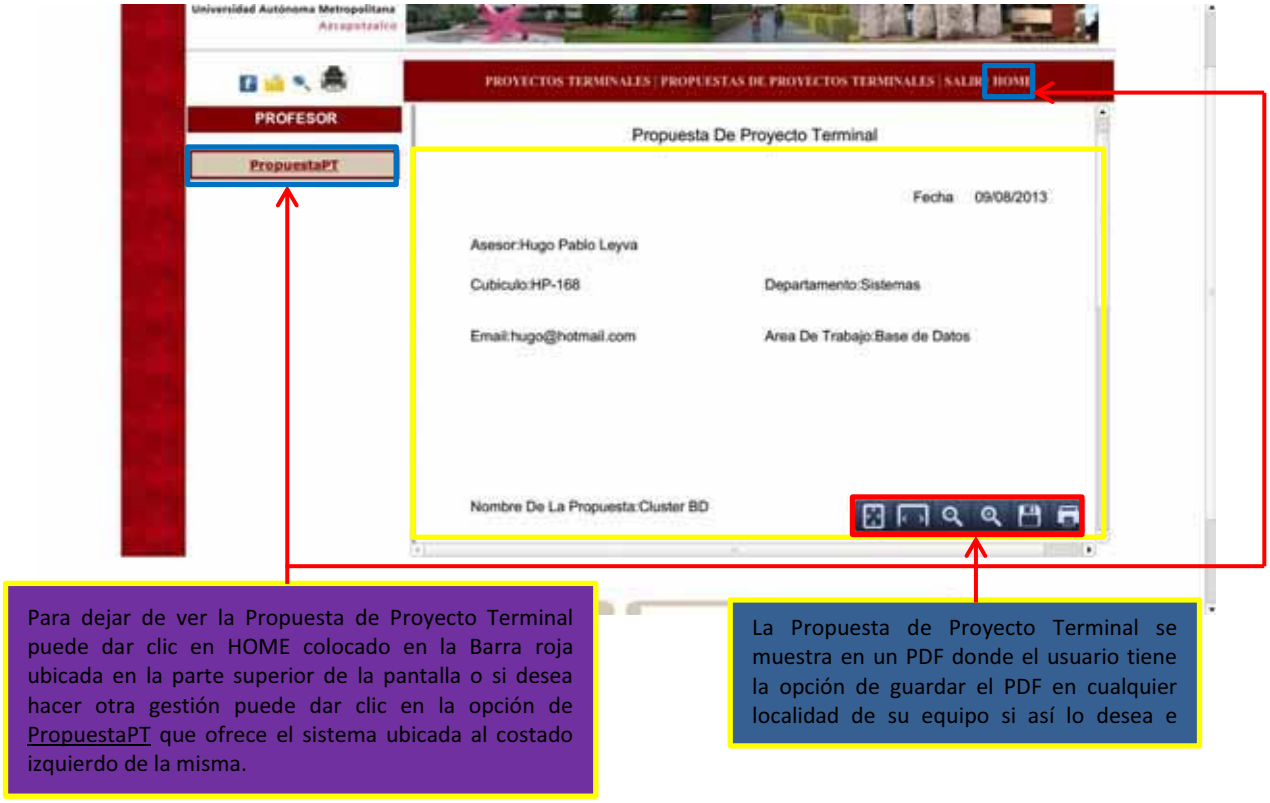
2) Ingresar una reseña correspondiente a la propuesta de proyecto terminal con la finalidad de dar una breve explicación a los alumnos interesados en dicha propuesta. Observar el ejemplo próximo:



3) Por último dar cli en el botón Crea Propuesta, observe la siguiente imagen:

Para poder observar la Propuesta que ha sido dada de alta esta se puede ver en la lista que se encuentra en la parte superior de la pantalla, tal y como se muestra en la siguiente imagen:

El archivo correspondiente a la propuesta será un archivo con extensión PDF y se visualizará de la siguiente manera:



Sí el usuario Profesor ya no desea realizar ninguna operación en el sistema podrá salir dando clic en el link salir colocado en la Barra de color rojo que se encuentra en la parte superior de la pantalla:

Una vez que el usuario termine su sesión el sistema mostrará la pantalla de inicio para disposición de cualquier usuario.

Bibliografía

[1] C. A. Flores, “Implementación de un gestor de documentos”, Propuesta de Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Unidad Azcapotzalco, México D.F. 2008.

[2] J. L. Cantor, “Sistema generador de reportes del puerto Ethernet”, Propuesta de Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Unidad Azcapotzalco, México D.F., 2009.

[3] R. C. Carrillo, “Gestor de contenidos de sitios web”, Propuesta de Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Unidad Azcapotzalco, México D.F., 2010.

[4] A. Esteves, "Sistema de Gestión de Congresos", Propuesta de Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2010.

[5] P. Comonfort, “Prototipo Web para Gestionar Proyectos de Software”, Propuesta de Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Unidad Azcapotzalco, México D.F., 2010.

[6] J. B. Anaya y O. A. Gutiérrez, “Espacios virtuales para el trabajo colaborativo del Observatorio Tecno – Educativo: Proyectos de investigación”, Propuesta de Proyecto Terminal de Ingeniería en Computación, Universidad Autónoma Metropolitana Unidad Azcapotzalco, México D.F., 2011.

[7] *RBT* [En línea], Disponible en: http://www.freedownloadmanager.org/es/downloads/Herramienta_de_Consultas_RBT_57306_p/, [Accesada:Feb. 13, 2012].

[8] Stimulsoft, *Stimulsoft Reports Designer Web* [En línea], Disponible en: http://www.stimulsoft.com/ReportsDesignerWeb.aspx, [Accesada: Feb. 13, 2012].

[9] MyDBR, *myDBR* [en línea], Disponible: http://mydbr.com/, [Accesada: Feb. 13, 2012].

[10] MySQL, *MySQL Report* [En línea], Disponible: http://mysqlreports.com/. [Accesada: Feb. 13, 2012].

[11] *LogAleph* [En línea], Disponible en: http://suba.uach.mx/archivos/capacitacion/Generador%20de%20Reportes2010.pdf. [Accesada: Feb. 13, 2012].