**UNIVERSIDAD NACIONAL DE INGENIERIA**
**FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN**
**INGENIERÍA EN COMPUTACIÓN**

# "Sistema Web para el programa de servicio social becario en la Universidad Nacional de Ingeniería para el Recinto Universitario Simón Bolívar"

**TRABAJO MONOGRÁFICO PRESENTADO POR:**

Br. Erick Antonio Navarrete Orozco
Br. Pedro Roberto Morales Ruiz

Para optar a título de
Ingeniero en Computación

**Tutor: MSc. María Lourdes Montes López**

**Octubre, 2016**

**Managua, Nicaragua**

# Dedicatoría

Primeramente a DIOS por habernos permitido llegar hasta este punto y habernos dado salud, ser el manantial de vida y darnos lo necesario para seguir adelante día a día.

A mi hijo Erick Navarrete Jr., por ser mi inspiración para seguir adelante en mis metas.

A Mi Tutora MSc. María Lourdes Montes López por brindarnos su sabiduría y apoyo en la realización de este proyecto.

A nuestros esfuerzos por permitirnos culminar nuestra meta.

# Dedicatoría

Primeramente a DIOS ser la fuerza en la que me sostengo.

A Mis padres Pedro Morales y Lilian del Carmen Ruiz Cabrera, por apoyarme en esta etapa de mi vida.

A Mi Tutora MSc. María Lourdes Montes López por ser el pilar fundamental de sabiduría y apoyo en la realización de este proyecto.

# Agradecimientos

Agradecemos a Dios por ser nuestra guía, nuestra luz y camino, y darnos la fortaleza para seguir adelante cada día.

A Mi Esposa Darlina Sánchez y Mi Hijo Erick Navarrete Jr., por ser Inspiración para culminación de esta meta.

A Mis padres Thelma Orozco y  Norman Navarrete, por apoyarme en esta etapa de mi vida.

A mi Abuela Thelma García, a mis Tías Lesbia Orozco y Darling Orozco por ser pacientes en mi crianza y enseñarme a seguir adelante.

A la Universidad y docentes por la formación y el conocimiento que nos transmitieron en toda la carrera.

A nuestra Tutora Ing. María Lourdes Montes por darnos la orientación, tiempo, dedicación además de su conocimiento y apoyo.

A la Ing. Giselle Calero, Responsable de la Oficina de Acción Social Becario por toda la colaboración y confianza brindada.

# Agradecimientos

Agradecemos a Dios por ser nuestra guía, nuestra luz y camino, y darnos la fortaleza para seguir adelante cada día.

A la Universidad y docentes por la formación y el conocimiento que nos transmitieron en toda la carrera. .

A nuestra Tutora Ing. María Lourdes Montes por darnos la orientación, tiempo, dedicación además de su conocimiento y apoyo.

A la Ing. Giselle Calero, Responsable de la Oficina de Acción Social Becario por toda la colaboración y confianza brindada.

# Resumen

El presente trabajo monográfico titulado **"Sistema Web para el programa de servicio social becario en la Universidad Nacional de Ingeniería para el Recinto Universitario Simón Bolívar (SISBECA)"** tiene como finalidad brindar a la Oficina de Acción Social una herramienta que ayude al control y seguimiento del Servicio Social Becario que realizan los estudiantes de todas las carreras de la "Universidad Nacional de Ingeniería **UNI**".

El Desarrollo del Sistema de Información SISBECA (Sistema del Servicio Becario) en la Universidad Nacional de Ingeniería "**UNI**", fue desarrollado haciendo uso del modelo de programación MVC (Modelo, Vista, Controlador), utilizando el software de desarrollo de aplicaciones Visual Studio Community Edition 2015 y como gestor de base de datos usamos SQLSERVER 2008 R2.

Para el análisis y desarrollo del sistema de información se utilizó la Metodología orientada a objetos.

Con el desarrollo del Sistema de Información en la Universidad Nacional de Ingeniería "**UNI**", se logró una mejor distribución de las asignaciones de las tareas a los becarios, información más ordenada, exacta y al día, para el Programa del Servicio Social.

# Tabla de contenido

# Indice de Tablas

# Índice de Figuras

# 1. Introducción

La Universidad Nacional de Ingeniería (UNI) es una casa de estudios superiores radicada en la ciudad de Managua, Nicaragua, estatal y autónoma. Es la primera universidad nacional que aglutina en una sola casa de estudios las ingenierías existentes hasta finales del siglo XX en Nicaragua[1].

La UNI, al igual que otras universidades del país, otorga becas estudiantiles como un beneficio adicional al estudiante que se destaca por su alto rendimiento académico y/o que posee limitaciones de carácter socioeconómico, actualmente se manejan siete tipos de becas. Adicionalmente, se cuenta con un programa de **SERVICIO SOCIAL DEL BECARIO/A,** dicho programa está bajo la responsabilidad de la **OFICINA DE ACCIÓN SOCIAL** adscrita a la **DIRECCIÓN DE BIENESTAR ESTUDIANTIL**, esta instancia es la responsable de la administración, planificación, control y seguimiento del servicio social becario, tiene como objetivo principal lograr la inserción de los/as estudiantes en actividades administrativas y académicas que complementen su formación integral, creándoles una cultura estudiantil que se fundamente en valores éticos y morales, excelencia académica, responsabilidad, vocación de servicio y solidaridad.

Todo estudiante de la UNI que tiene beca tiene que realizar tiempo (horas) de servicio social dentro y fuera de la institución, como lo estípula el reglamento de beca estudiantil, el cual establece lo siguiente:

1. El estudiante debe cumplir al menos con diez horas semanales en las distintas actividades programadas de servicio social e institucional en las diferentes instancias de la universidad.
2. El incumplimiento y mal desempeño del mismo son causales de pérdida de beca.

[1] http://www.uni.edu.ni/Extension **(Recuperado el 01 de octubre del 2014)**

En la Universidad Nacional de Ingeniería existe la ***"Normativa del servicio social del becario"*** en la que especifican los términos bajo los cuales se debe de regir el servicio social del becario, así como las funciones y responsabilidades tanto del alumno como de quien recibe el servicio.

Los estudiantes brindaran su aporte en actividades, programas y proyectos propios del quehacer institucional en las instalaciones de la universidad en horario de oficina, tales como:

1. Auxiliar docente
2. Servicios administrativos
3. Asistencia en laboratorios
4. Colaborando en proyectos específicos
5. Actividades de la Dirección de Bienestar Estudiantil

Para que los becarios/as cumplan con el tiempo de servicio social deben presentarse ante el responsable de la instancia donde fue asignado y cumplir con el trabajo asignado durante el periodo académico.

Actualmente la Oficina de Acción Social de la Universidad Nacional de Ingeniería, solicito el apoyo especializado a Secretaria de Facultad de la FEC, porque no cuenta con un sistema automatizado para el **PROGRAMA DE SERVICIO SOCIAL DEL BECARIO/A UNI**, el control del tiempo de servicio social becario se realiza de forma manual auxiliado de herramientas como Word y Excel, además la Responsable de la Oficina de Acción Social tiene que movilizarse a través de las diferentes instancias de la UNI para recopilar información del cumplimiento de los becados así como en el RUPAP, motivo por el cual este trabajo de tesis pretende dar respuesta a esta necesidad, mediante la implementación del sistema de Programa de servicio social becario/a UNI.

El Sistema Web del Programa de Servicio Social Becario tiene como propósito agilizar el proceso de control de tiempo y calidad del servicio que brindan los becarios en las áreas de la universidad o proyectos sociales a los cuales son asignados durante el semestre que dura la beca.

El Sistema será diseñado en plataforma Web porque el Servicio Social Becario es en su mayoría realizado a lo interno de la Universidad y en vista que el responsable de las diferentes instancias de la UNI que soliciten el servicio de becarios le deberá dar seguimiento, se hace necesario alimentar la información desde cualquiera de las oficinas de donde el estudiante este brindando su tiempo de servicio social, información que le llegará consolidada a la Responsable de la Oficina de Acción Social. Además, considerando que los becados están distribuidos por carreras en varios recintos, se pretende en un futuro que el uso del sistema se extienda a todos los recintos.

Adicionalmente, el Sistema del Programa del Servicio Social Becario, será diseñado de forma independiente, al Sistema de Becas y Sistema de Registro Académico de la UNI, por el nivel de confidencialidad y mecanismos de seguridad establecidos por la DITI, sin embargo será alojado en los servidores de la FEC para el periodo de prueba, una vez concluido este periodo la DITI definirá una interfaz para integrarlo con los sistemas antes mencionados y alojarlo en los servidores de la DITI.

# 2. Objetivos

## 2.1. Objetivo General

Desarrollar un Sistema web para el programa del servicio social becario, que permita dar seguimiento al trabajo social desarrollado por los becados de la Universidad Nacional de Ingeniería.

## 2.2. Objetivos Específicos

1. Levantar la información necesaria relacionada con el tópico de estudio, mediante entrevistas no estructuradas y observación directa de los procesos que se ejecutan en relación al programa del servicio social becario.
2. Analizar la información recopilada para determinar el alcance y las necesidades del sistema web para el programa de servicio social becario.
3. Diseñar la aplicación web que responda a los requerimientos del proceso y de la responsable de la Oficina de Servicio Social.
4. Realizar las pruebas o correcciones necesarias, conjuntamente con el personal de la Oficina de Servicio Social, con la finalidad de verificar que la aplicación web cumpla con las expectativas deseadas.
5. Implantar la aplicación web para validar su efectividad en el manejo del programa de servicio social becario, realizando la respectiva capacitación de los usuarios finales.
6. Agilizar la distribución y control del tiempo del servicio social que brindan los becarios a la Universidad Nacional de Ingeniería.

# 3. Justificación

Hoy en día las instituciones se desarrollan y crecen, estas tienen más exigencias que antes, como la agilización de los procesos, ya que el éxito se basa en la agilidad y eficiencia con que se manejen los datos, naciendo así la necesidad de la integración de las tecnologías de la información y la comunicación (TIC).

Tomando en consideración que la Oficina de Acción Social de la Universidad Nacional de Ingeniería, trabaja de forma manual sus procesos y no cuenta con una herramienta automatizada que le permita agilizar y administrar la información del tiempo de servicio social becario, se dificulta la recopilación y procesamiento de la información.

Partiendo de esta necesidad, el presente trabajo monográfico, contempla el desarrollo de un Sistema Web que permitirá controlar el tiempo de servicio social becario, agilizar la distribución de asignaciones a los becarios y sus horas trabajadas de forma que permita retroalimentar el proceso de autorización y renovación de becas.

Con la incorporación de las tecnologías de la información y comunicación (TIC´S) se manejará la información de forma más centralizada y ordenada, además de tener acceso desde cualquier punto dentro de la institución, ya que el software se desarrollará bajo ambiente web.

Por otra parte, el sistema será desarrollado en plataforma WEB dado que han demostrado mejores resultados frente a los Sistemas Tradicionales Cliente/Servidor, brindando mejores beneficios, considerando también que no es necesario pagar licencias por cada computadora a la que se le instalará porque el sistema está alojado en un servidor web, será fácil de acceder desde cualquier punto con conexión a internet y los usuarios podrán consultar información en cualquier momento.

La información que gestionará la aplicación Web permitirá tomar decisiones más oportunas y precisas.

# 4. Marco Teórico

A continuación se describe de forma ordenada los elementos teóricos, metodologías y conceptos, que sustentan la formulación y evaluación de este proyecto monográfico.

## 4.1. Universidad Nacional de Ingeniería (UNI)

La Universidad Nacional de Ingeniería es una Institución de la Educación Superior, estatal y autónoma, en búsqueda permanente de la excelencia académica, dedicada a formar profesionales en el campo de la Ciencia, la Ingeniería y la Arquitectura para que generen y difundan conocimientos científicos con conciencia social, ética y humanística, con la finalidad de contribuir a la transformación tecnológica y al desarrollo sustentable de Nicaragua y la región Centroamericana.

"La Universidad Nacional de Ingeniería es una Institución que se consolida como líder nacional en la enseñanza de la Ingeniería y la Arquitectura, y es un referente en la investigación científica y tecnológica, construido mediante la interacción con los diversos actores y sectores sociales, económicos y culturales del país, contribuyendo al crecimiento y desarrollo nacional en función del bienestar de la sociedad nicaragüense" [13].

En la actualidad, la población estudiantil que tiene la UNI es de 10,880 estudiantes de pregrado y postgrado, con 11 carreras que se distribuyen en seis facultades y tres campus, además cuenta con un amplio programa de postgrados y doce maestrías especializadas. La planta docente de la universidad es de 380 catedráticos, de los cuales el 50% tiene estudios de Master o títulos de Doctorados y el 30 % son docentes investigadores.[2]

[2] **http://www.uni.edu.ni/Alma_Mater/Historia** - **Unificación de la enseñanza de la Ingeniería y la Arquitectura (Recuperado el 01 de octubre del 2014)**

Las principales áreas de investigación de la Universidad son la Biotecnología, Alimentos, Procesos químicos, Sistema de Información y Control Industrial, Prevención, Mitigación y Atención de desastres, Medio Ambiente, Asentamientos Humanos y Desarrollo Urbano, entre otros.

En cumplimiento de la misión y visión institucional, la Universidad Nacional de Ingeniería ha implementado procesos de auto evaluación de la universidad a través del Sistema Centroamericano de Evaluación y Acreditación de la Educación Superior (SICEVAES), además es fundadora y miembro activa de la Red Centroamericana de Instituciones de Ingeniería (REDICA) participando en el proyecto de mejoramiento de la enseñanza de la Ingeniería y la Arquitectura en Centroamérica.

La Universidad Nacional de Ingeniería como su lema lo indica “Líder en Ciencia y Tecnología”, fue la primera institución de Nicaragua poseedora de un nodo de Internet y en la actualidad impulsa un amplio proceso de modernización contando con 85 laboratorios; sin embargo la Universidad no solo se destaca por lo señalado, sino que también desde 1991 ha tenido la supremacía en festivales artísticos ínter universitario.

El acceso a la Universidad Nacional de Ingeniería es libre y gratuito para todos los nicaragüenses, siempre que los interesados cumplan con los requisitos y condiciones académicas exigidas, sin discriminación por razones de nacimiento, nacionalidad, credo político, raza, sexo, religión, opinión, origen, posición económica o condición social.

## 4.2. Dirección de Bienestar Estudiantil (DBE) UNI

Actualmente la Dirección de Bienestar Estudiantil es una instancia adscrita funcionalmente a Rectoría, su función es contribuir a la formación integral de los estudiantes, atender problemas de índole académicos, socio-económicos, deportivos y culturales de la comunidad estudiantil, además de organizar la acción social universitaria hacia los distintos sectores de la sociedad.

La Dirección de Bienestar Estudiantil, ha creado la Oficina de Acción Social Universitaria como una instancia coordinadora que permita organizar, ordenar, proyectar y aglutinar acciones de servicio social y humano, que puedan realizar los becarios como parte del cumplimiento de sus horas de servicio social e institucional, de acuerdo al Reglamento de Beca Estudiantil de Grado, la Ingeniera Giselle Calero, es la encargada de la asignación y distribución del servicio social becario.

En la oficina de acción social todos los procesos se desarrollan de forma manual, mediante un formulario que se entrega en las áreas donde el becario realiza su tiempo de servicio social y una vez completado se transcriben a Word/Excel.

## 4.3. Becas

Una beca es la ayuda económica o subvención que generalmente una institución le entrega a alguien para que lleve a cabo estudios o investigaciones.

Tradicionalmente, las becas se entregan a aquellos estudiantes, profesionales que no cuentan con el capital o el nivel económico suficiente para pagarse el estudio o investigación en cuestión, pero su nivel intelectual o profesional es tan alto que se les decide ayudar. En tanto diferentes instituciones estatales, Ministerios, Universidades y escuelas, Organizaciones no gubernamentales, tales como fundaciones o asociaciones y empresas privadas, como bancos y compañías, son quienes normalmente entregan las becas.

Se designa como ***becario*** al individuo que goza y es titular de una beca. El término además se encuentra vinculado a la figura de pasante, que es aquel estudiante o profesional recién recibido o que está a punto de serlo en la Universidad.[3]

[3] **http://estatico.uned.ac.cr/bienestar/vida/becas/documentos.shtml (Recuperado el 8 de Octubre del 2014).**

En la Universidad Nacional de Ingeniería se brindan siete tipos de becas a las cuales los estudiantes pueden optar, las cuales se categorizan de la siguiente manera:

1. Beca residencia: Se destina a estudiantes que provienen de zonas alejadas y/o de difícil acceso, según estudio realizado por la comisión de beca por carrera. La beca contempla alojamiento, alimentación, estipendio monetario, enseres personales, servicios médicos de emergencia y exoneración de matrícula.
2. Beca monetaria: Es la asignación en efectivo que se le entrega al estudiante becario. Se clasifica en: A, B, C. Los montos en efectivo de la beca monetaria se establecen en el presupuesto general de ingresos y egresos de la Universidad.
3. Beca alimenticia: Es la asignación de un almuerzo que se otorga al estudiante de lunes a viernes en cada semestre académico exceptuando los períodos de primera y segunda convocatoria.
4. Beca a la excelencia académica: Es la asignación monetaria que se otorga al estudiante cuyo rendimiento académico semestral sea igual o mayor a ochenta y cinco (85%) en promedio.
5. Beca deportiva y cultural: Es la asignación monetaria que se le entrega al estudiante becario; que se destaca en las actividades deportivas y culturales con proyección institucional.
6. Beca Monográfica: Se asigna al o a los estudiantes egresados cuyo tema monográfico sea de interés institucional y social.
7. Beca Curso de Graduación: Se asigna al o los estudiantes que hayan cumplido los requisitos establecidos en la normativa correspondiente, y la aprueba la Comisión Central de Beca.

Por otra parte, la beca en la Universidad Nacional de Ingeniería estimula el rendimiento académico de sus estudiantes.

La Universidad Nacional de Ingeniería está comprometida a crear condiciones que garanticen la permanencia exitosa y la eficiencia de terminar la graduación, promoviendo la solidaridad social ante los problemas individuales y colectivos que se presentan en la Comunidad Universitaria. Esta solidaridad a lo interno se extiende y proyecta hacia la sociedad, mediante el reforzamiento de sus funciones de servicio a la población y su capacidad en la prevención, asistencia y mitigación de desastres. Con un compromiso permanente hacia el desarrollo sostenible por medio de asesorías, asistencia técnica, capacitación, ayuda moral y material.

Es la instancia del DBE (Dirección de Bienestar Estudiantil) mediante el cual la Universidad Nacional de Ingeniería, promueve la formación académica a nivel superior del estudiante con buen rendimiento académico, favoreciendo a estudiantes con limitaciones económicas y originarias de las regiones más alejadas del país. Este programa forma parte del conjunto de acciones que ejecutan las autoridades universitarias con el fin de estimular la excelencia académica y en cumplimiento del principio de equidad establecido en el Estatuto de nuestra Alma Mater.

### 4.3.1. Programa de Becas

El objetivo de este programa es contribuir a la formación integral de los estudiantes, a la formación de una cultura estudiantil que se fundamente en valores éticos y morales, excelencia académica, responsabilidad, vocación de servicio y solidaridad. Promover altos niveles de calidad en la búsqueda de la excelencia académica y el mejoramiento de la vida estudiantil, garantizando mayores niveles de justicia, equidad y eficiencia en los recursos destinados al becario.

Se cuenta con comisiones de becas, cuya funciones es la de seleccionar y aprobar por unanimidad a los becarios de sus respectivas carreras, estas están sujetas a la Comisión Central de Beca (CCB).

La Comisión Central de Beca es una estructura colegiada integrada por el Rector o el que delegue, el Director de la Dirección de Bienestar Estudiantil, el Responsable de Beca y los presidentes estudiantiles de recintos, tiene como función ratificar los casos aprobados por la Comisión de Carrera, así como revisar, analizar, dictaminar y resolver aquellas situaciones que se presenten en materia de beca, que no estén contempladas en el reglamento.

Para optar a una beca, se debe presentar la documentación de solicitud de beca completa en la Dirección de Bienestar Estudiantil y una vez que se obtenga la beca deberá cumplir con los deberes que estipula el reglamento de becas de la Universidad Nacional de Ingeniería.

## 4.4. Las tecnologías de información y comunicación (TIC)

### 4.4.1. ¿Que son las TIC?

Podemos abordar el concepto de las TIC desde tres enfoques, que van a unificarlo en uno solo:

- Desde el enfoque tecnológico:
  1. Aplicación de los conocimientos científicos en las actividades humanas.
  2. Creación de productos, instrumentos, lenguajes, procesos y métodos al servicio de las personas.
- Desde el enfoque de información:
  1. Datos que tienen significado para determinados colectivos.
- Desde el enfoque de comunicaciones:
  1. Comunicación entre seres sociales
  2. Expresión de ideas, sentimientos y necesidades

Por lo anterior, podemos decir que las TIC son:

- Un conjunto de avances tecnológicos que proporciona la informática, las telecomunicaciones y las tecnologías audiovisuales.

- Comprenden los desarrollos relacionados con las computadoras, Internet, la telefonía, las aplicaciones multimedia y la realidad virtual.
- Estás tecnologías proporcionan información, herramientas para su proceso y canales de comunicación.

### 4.4.2. Convergencia de las TIC


Figura Nº 1: Convergencia de TIC, herramientas de la Informática para gestión empresarial

### 4.4.3. Características de las TIC

1. Interactividad:

    a. Permiten la interacción de sus usuarios.

    b. Posibilitan que dejemos de ser espectadores pasivos, para actuar como participantes.

2. Instantaneidad:

    Se refiere a la posibilidad de recibir información en buenas condiciones técnicas en un espacio de tiempo muy reducido, casi de manera instantánea

3. Interconexión:

    La interconexión de equipos, procesos, personas y diferentes tecnologías.

4. Digitalización:

   Hace referencia a la transformación de la información analógica en códigos numéricos, lo que favorece la transmisión de diversos tipos de información por un mismo canal.

5. Diversidad:

   La diversidad de tecnologías permite desempeñar diversas funciones. Un videodisco transmite informaciones por medio de imágenes y textos y la videoconferencia puede dar espacio para la interacción entre los usuarios.

6. Colaboración:

   Las TIC como tecnologías colaborativas posibilitan el trabajo en equipo, es decir, varias personas en distintos roles pueden trabajar para lograr la consecución de una determinada meta común.

7. Penetración en todos los sectores:

   Por todas esas características las TIC penetran en todos los sectores sociales; sean culturales, económicos o industriales. Afectan al modo de producción, distribución y consumo de los bienes materiales, culturales y sociales.

### 4.4.4. Actividades de las TIC

El sistema de información es una colección organizada de personas, información y procesos de negocio; diseñados para transformar entradas en salidas y lograr un objetivo.

Los sistemas de información llevan a cabo tres actividades básicas: entrada, procesamiento y salida.

### 4.4.5. Infraestructura de las TIC

Consiste en:

1. Un conjunto de dispositivos físicos y aplicaciones de software que se requieren para operar toda la empresa.

2. Conjunto de servicios que una empresa es capaz de proveer a sus clientes, proveedores y empleados.
3. Esta infraestructura debería apoyar la estrategia de sistemas de información y sobre todo de negocio de la empresa.

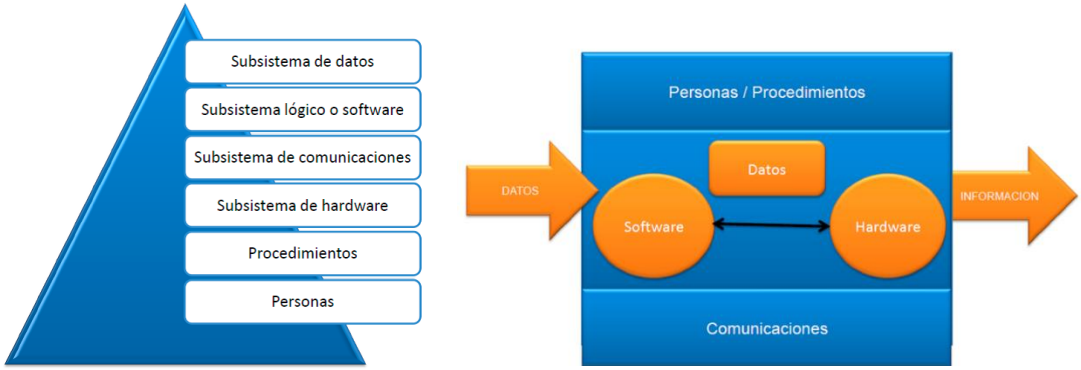



Figura Nº 2: Infraestructura de las TICS

### 4.4.6. Hardware

Se refiere a todas las partes tangibles de un sistema informático; sus componentes son: eléctricos, electrónicos, electromecánicos y mecánicos.

### 4.4.7. Software

Instrucciones en lenguaje especial y organizado en programas, que dicta al sistema físico las tareas a realizar, permitiendo la relación usuario-computador.

Para el desarrollo del sistema web para el programa de servicio social becario de la Universidad Nacional de Ingeniería nos apoyamos en el siguiente software:

- Software base: tareas genéricas e interface de usuario, compiladores de visual net y C++.
- Sistema operativo: ambientes Windows 7 y superiores.
- Software de ayuda a la programación: traducen el lenguaje simbólico a código de máquina (ensambladores, compiladores e intérpretes) (C#, HTML).

Adicionalmente utilizamos el método de programación ASP.net MVC Express y el diseño de la base de datos en SQL SERVER EXPRESS por ser herramientas de uso libre.

### 4.4.8. MVC (Modelo Vista Controlador)

MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, Model, Views & Controllers.

#### *4.4.8.1. Arquitectura de aplicaciones MVC*

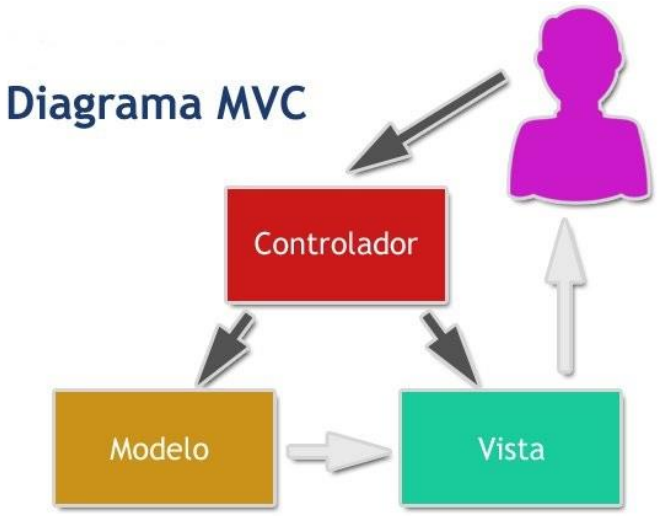



Figura Nº 3: Arquitectura de aplicaciones MVC

#### 4.4.8.2. Flujo de trabajo característico en un esquema MVC

1. El usuario realiza una solicitud a nuestro sitio web. Generalmente está desencadenada por acceder a una página de nuestro sitio. Esa solicitud le llega al controlador.
2. El controlador comunica tanto con modelos como con vistas. A los modelos les solicita datos o les manda realizar actualizaciones de los datos. A las vistas les solicita la salida correspondiente, una vez se hayan realizado las operaciones pertinentes según la lógica del negocio.
3. Para producir la salida, en ocasiones las vistas pueden solicitar más información a los modelos. En ocasiones, el controlador será el responsable de solicitar todos los datos a los modelos y de enviarlos a las vistas, haciendo de puente entre unos y otros. Sería corriente tanto una cosa como la otra, todo depende de nuestra implementación; por eso esa flecha tiene otro color.
4. Las vistas envían al usuario la salida. Aunque en ocasiones esa salida puede ir de regreso al controlador y es quien hace el envío al cliente, por eso la flecha de otro color.

MVC es un "invento" que ya tiene varias décadas y fue presentado incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos frameworks de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.[4]

#### 4.4.8.3. ASP.NET MVC

El patrón de arquitectura Model-View-Controller (MVC) separa una aplicación en tres componentes principales: el modelo, la vista y el controlador.

El ASP.NET MVC framework proporciona una alternativa al patrón de ASP.NET Web Forms para crear aplicaciones Web basadas en MVC. El ASP.NET MVC

[4] **http://www.desarrolloweb.com/articulos/que-es-mvc.html,** (Recuperado el 18 de octubre del 2014)

framework. Es un framework de presentación ligero, altamente comprobable que (como con aplicaciones basadas en formularios Web Forms) está integrada con las características ASP.NET existentes, como la autenticación basada en membresía y páginas maestras. El framework MVC se define en el espacio de nombres System.Web.Mvc y es una parte fundamental, con el apoyo del espacio de nombres System.Web.

MVC es un patrón de diseño estándar con el que muchos desarrolladores están familiarizados. Algunos tipos de aplicaciones Web se beneficiarán del framework MVC. Otros continuarán utilizando el patrón de aplicación ASP.NET tradicional basada en formularios Web y las devoluciones de datos. Otros tipos de aplicaciones Web combinan ambos enfoques.

El modelo de objetos es la parte de la aplicación que implementa la lógica para el dominio de datos de las aplicaciones. A menudo, objetos del modelo recuperan y almacenan el estado del modelo en una base de datos. Por ejemplo, un objeto producto podría recuperar la información de una base de datos, operarlo y luego escribir información actualizada a una tabla de productos de SQL Server.

En pequeñas aplicaciones, **el modelo** es a menudo una separación conceptual en lugar de una física. Por ejemplo, si la aplicación sólo lee un conjunto de datos y envía a la vista, la aplicación no tiene una capa de modelo físico y clases asociadas. En ese caso, el conjunto de datos asume el papel de un objeto del modelo.

**Vistas:** son los componentes que muestran la interfaz de usuario de las aplicaciones (IU). Normalmente, esta interfaz de usuario es creado a partir de los datos del modelo. Un ejemplo sería una vista de edición de una tabla de productos que muestra cuadros de texto, listas desplegables, casillas de verificación basadas en el estado actual de un objeto de productos.

**Los controladores:** Los controladores son los componentes que manejan interacción con el usuario, trabajan con el modelo y en última instancia,

selecciona una vista para representar que la interfaz que muestra al usuario. En una aplicación MVC, la vista sólo muestra información; el controlador maneja y responde a la entrada del usuario y la interacción. Por ejemplo, el controlador trata los valores de cadena de consulta y pasa estos valores al modelo, que a su vez consulta la base de datos mediante el uso de los valores.

#### 4.4.8.4. Ventajas de una aplicación Web basada en MVC

MVC ofrece las siguientes ventajas:

- Resulta más fácil de administrar la complejidad dividiendo una aplicación en el modelo, la vista y el controlador.
- No utiliza el estado de vista o formas basadas en servidor. Esto hace que el framework MVC sea ideal para desarrolladores que desean el control total sobre el comportamiento de una aplicación.
- Utiliza un patrón Front Controller que procesa las solicitudes de aplicación Web a través de un único controlador. Esto le permite diseñar una aplicación que soporta una rica infraestructura de enrutamiento.
- Proporciona mayor apoyo para el desarrollo guiado por pruebas (TDD).
- Funciona bien para aplicaciones Web que son apoyadas por grandes equipos de desarrolladores y diseñadores Web que necesitan un alto grado de control sobre el comportamiento de la aplicación.

#### 4.4.8.5. Características de ASP.NET MVC Framework

El ASP.NET MVC framework proporciona las siguientes características:

- Separación de tareas de la aplicación (entrada lógica, lógica de negocio y la lógica de interfaz de usuario), capacidad de prueba y desarrollo guiado por pruebas (TDD) por defecto. Todos los contratos del núcleo en el framework MVC están basados en la interfaz y pueden ser probados usando objetos mock, que son objetos simulados que imitan el comportamiento de objetos reales en la aplicación. Usted puede hacer pruebas unitarias a la aplicación sin tener que correr los controladores en

- un proceso ASP.NET, que lo hace rápido y flexible de pruebas unitarias. Puede usar cualquier marco de pruebas unitarias que sea compatible con .NET Framework.
- Un marco extensible y enchufable. Los componentes del marco ASP.NET MVC están diseñados para que puedan ser fácilmente reemplazados o modificados para requisitos particulares. Puede enchufar su propio motor de vista, política de enrutamiento de URL, serialización de parámetro de método de acción y otros componentes. El marco de ASP.NET MVC también soporta el uso de la inyección de dependencia (DI) y la inversión del Control (IOC) modelos de envase. DI permite inyectar objetos de una clase, en lugar de depender de la clase para crear el objeto en sí. IOC especifica que si un objeto requiere otro objeto, los primeros objetos deberían obtener el segundo objeto desde una fuente externa como un archivo de configuración. Esto hace más fácil la prueba.
- Un componente URL-cartografía potente que te permite crear aplicaciones que tienen URLs comprensibles e investigable. URLs no tiene que incluir las extensiones de nombre de archivo y están diseñadas para soportar patrones de URL nombres que funcionan bien para la búsqueda de optimización (SEO) y transferencia de estado representacional (REST) direccionamiento del motor.
- Soporte para usar el marcado en la página ASP.NET existente (archivos .aspx), control de usuario (archivos .ascx) y página maestra (Master files) marcado archivos como plantillas de vista. Puede usar las características existentes de ASP.NET con el marco ASP.NET MVC, tales como páginas maestras anidadas, expresiones en línea (< % = % >), controles de servidor declarativos, plantillas, enlace de datos, localización y así sucesivamente.
- Soporte para las características existentes de ASP.NET. ASP.NET MVC le permite utilizar funciones como la autenticación de formularios y autenticación de Windows, URL autorización, membresía y papeles, salida y almacenamiento en caché de datos, administración de Estados,

de sesión y perfil, vigilancia de la salud, el sistema de configuración y la arquitectura de proveedor.

El sistema web de control y seguimiento del tiempo del servicio social becario de la Universidad Nacional de Ingeniería está desarrollado en **ASP.NET MVC EXPRESS es una versión gratuita de la plataforma .NET,** este se preocupa por crear procesos que aseguren calidad en los programas que se realizan y esa calidad atiende a diversos parámetros que son deseables para todo desarrollo, como la estructuración de los programas o reutilización del código, lo que debe influir positivamente en la facilidad de desarrollo y el mantenimiento.

Los ingenieros del software que se dedican a estudiar de qué manera se pueden mejorar los procesos de creación de software y una de las soluciones a las que han llegado es la arquitectura basada en capas que separan el código en función de sus responsabilidades o conceptos. Por tanto, cuando ocupamos ASP.net MVC lo primero que tenemos que saber es que está ahí para ayudarnos a crear aplicaciones con mayor calidad. [5]

#### *4.4.8.6. Ventajas de usar ASP.NET MVC FrameWork*

1. Permite dividir la lógica de negocio del diseño, haciendo los proyectos más escalables.
2. Facilita el uso de URL amigables, importantes para el SEO (posicionamiento web), la mayoría de frameworks MVC lo controlan.
3. Muchos frameworks MVC ya incluyen librerías de JavaScript como Jquery, lo que permite validar formularios (Ej. Jquery.Validate) en el cliente y en el servidor.
4. Permite utilizar abstracción de datos, como lo hace Ruby on Rails o con frameworks como Hibernate para Java o NHibernate para ASP .NET MVC, facilitando la realización de consultas a la base de datos.

[5] **Ingeniería del Software, Roger S. Pressman, Un enfoque Práctico. Sexta Edición.**

5. La mayoría de frameworks controlan el uso de la memoria Caché, hoy en día muy importante para el posicionamiento web, ya que buscadores como google dan prioridad a las web que tengan menor tiempo de descarga.
6. En el caso de proyectos donde hay varios desarrolladores, el seguir métodos comunes de programación, hace que el código sea más entendible entre estos, pudiendo uno continuar el trabajo de otro. En estos casos es conveniente utilizar herramientas de control de versiones como subversiones.
7. Los frameworks están creados para facilitar el trabajo de los desarrolladores, contiene clases para controlar fechas, URL's, Webservices, etc., lo que tiene una gran ventaja en cuanto a productividad.
8. Poco a poco el desarrollo web se orienta a lo que se denomina "Agile Web Development" (Desarrollo ágil de aplicaciones web), con frameworks como Ruby on Rails que ayudan a crear proyectos de calidad y en corto tiempo. Existen varios frameworks en C# e incluso ASP .NET que en su nueva versión ya contemplan el MVC con Visual C#.
9. Utilizar herramientas con tecnología escalable hace más atractivo un proyecto de software en caso de buscar inversión externa.
10. Un Framework MVC ayuda a controlar los recursos del servidor, evitando Bugs que puedan repercutir en el rendimiento, por ejemplo, muchas veces olvidamos cerrar conexiones a la base de datos, sobrecargando el servidor.

#### *4.4.8.7. Lógica del negocio/lógica de la aplicación*

Es un conjunto de reglas que se siguen en el software para reaccionar ante distintas situaciones. En una aplicación el usuario se comunica con el sistema por medio de una interfaz, pero cuando acciona esa interfaz para realizar acciones con el programa, se ejecutan una serie de procesos que se conocen como la lógica del negocio. Este es un concepto de desarrollo de software en general.

La lógica del negocio, aparte de marcar un comportamiento cuando ocurren cosas dentro de un software, también tiene normas sobre lo que se puede y no se puede hacer. Eso también se conoce como reglas del negocio. En el MVC la lógica del negocio queda del lado de los modelos. Ellos son los que deben saber cómo operar en diversas situaciones y las cosas que pueden permitir que ocurran en el proceso de ejecución de una aplicación.

## 4.5. Ingeniería del Software

La ingeniería de software es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de los programas informáticos (software).[6]

Esta disciplina trasciende la actividad de programación, que es el pilar fundamental a la hora de crear una aplicación. El ingeniero de software se encarga de toda la gestión del proyecto para que éste se pueda desarrollar en un plazo determinado y con el presupuesto previsto.

La ingeniería de software, por lo tanto, incluye el análisis previo de la situación, el diseño del proyecto, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema.

Cabe destacar que el proceso de desarrollo de software (según Roger Pressman. 2002), Figura Nº 4, implica lo que se conoce como ciclo de vida del software, que está formado por siete etapas: análisis, diseño, codificación, pruebas, validación, mantenimiento y evolución, definición de necesidades.

El proceso de Ingeniería del software se aplica también en el “Sistema web del programa del servicio social becario de la Universidad Nacional de Ingeniería”, con la finalidad de proponer mejoras de funcionalidad del sistema y documentar todo el proceso de desarrollo del sistema web.

[6] **Roger S. Pressman. 2002. Ingeniería del software, Un enfoque práctico”; McGraw-Hill**

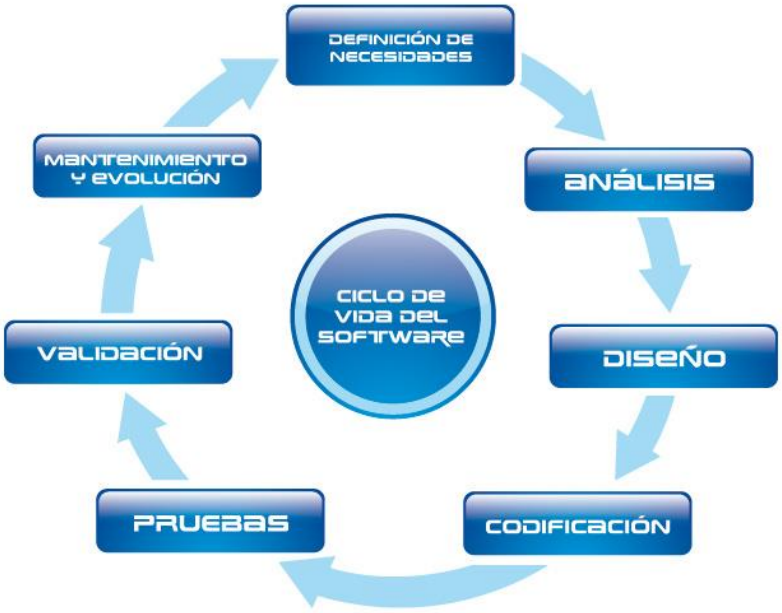



Figura Nº 4: Ciclo de vida del software, Ingeniería de Software, Pressman, 5ta Edición

### 4.5.1. Capas del Software

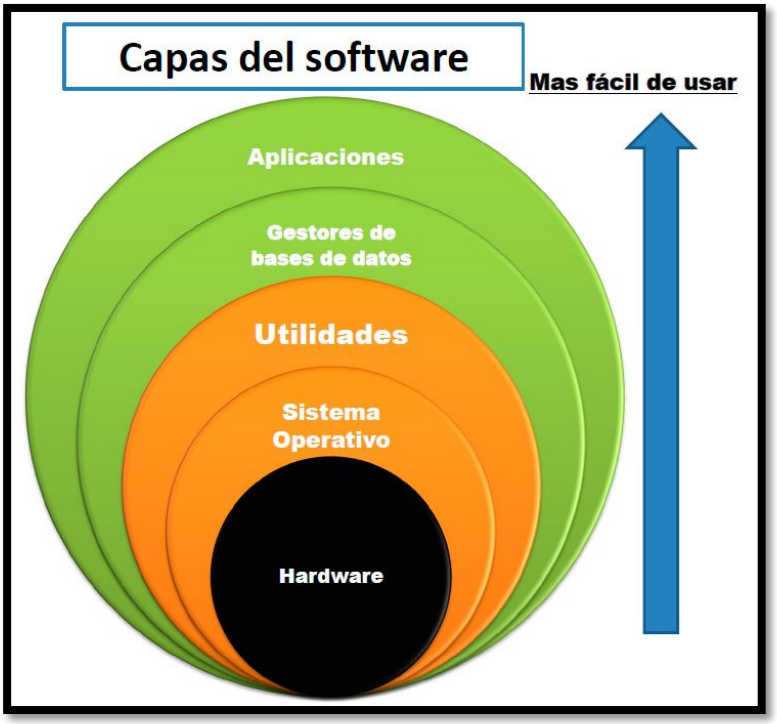



Figura Nº 5: Capas del Software, TIC y la gestión comercial

### 4.5.2. Bases de Datos

Son datos ordenados según cierta estructura (registros y campos). Independientes de las aplicaciones que los manejan. Se clasifican en dinámicas o transaccionales y en estáticas o históricas.

#### 4.5.2.1. Beneficios de las bases de datos (integradas)

- Recursos compartidos (calidad y gestión más eficiente)
- Seguridad
- Control de la redundancia de datos
- Consistencia de los datos
- Independencia de los datos y los programas que los tratan
- Mínimo coste de actualización (personal, administrativos, de equipos, etc.)

#### 4.5.2.2. Desventajas

Son más vulnerables que los ficheros, requiere medidas de seguridad y control de acceso (usuarios autorizados).

### 4.5.3. Sistemas gestores de base de datos (SGBD)

Es el software que permite a una organización centralizar los datos, administrarlos eficientemente y proporcionar mediante programas de aplicación el acceso a los datos almacenados.

Separa las vistas lógica y física de los datos, quedando el usuario final liberado de la tarea de comprender dónde y cómo se almacenan realmente los datos. La vista lógica presenta los datos cómo deberían percibirlos los usuarios finales y la vista física muestra cómo están organizados y estructurados realmente los datos en un medio de almacenamiento físico.

Utiliza tres lenguajes: Lenguaje de definición de datos, Lenguaje de manipulación de datos, Lenguaje de consulta (el más usual es el SQL, Structured Query

Language). Actúan como una interfaz entre la base de datos, el usuario y las aplicaciones que las utilizan.

#### 4.5.3.1. Funcionalidades que debe cumplir un SGBD:

***Consistencia:*** actualizar automáticamente los datos repetidos, cuando no se ha podido eliminar la redundancia de bases de datos.

***Seguridad:*** frente a usuarios y terceros malintencionados o errores humanos. Asignará permisos a usuarios y grupos de usuarios para controlar el nivel de acceso a los datos.

***Integridad y respaldo:*** copias de respaldo y restauración de los datos almacenados.

***Control de la concurrencia:*** controlar el acceso simultáneo a una misma base de datos para evitar inconsistencias.

### 4.5.4. Microsoft SQL Server

Como las aplicaciones de red y la Web son cada vez más importante, la importancia de los sistemas de gestión de bases de datos relacionales también aumenta. Al seleccionar el más adecuado para sus necesidades es esencial para garantizar la calidad y el rendimiento adecuado de su aplicación.

**Microsoft SQL Server** es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional cuya principal función es la de almacenar y consultar datos solicitados por otras aplicaciones, sin importar si están en la misma computadora, si están conectadas a una red local o si están conectadas a través de internet (plataforma “Cloud-Ready”).

Para el desarrollo del Sistema web del Programa de Servicio Social se decidió **Microsoft SQL Server Express** por que incluye software de gestión profesional, a nivel de empresa de base de datos, es fácil de usar y tiene muchas funciones. Soporte completo para desencadenantes. Por otro lado, el software ofrecido por

Microsoft también ofrece una estrecha integración con el marco NET. Además de ser un **sistema de administración de datos gratuito, eficaz y confiable** que ofrece un almacén de datos completo y confiable para sitios web ligeros y aplicaciones de escritorio.

Microsoft SQL Server está diseñado para ejecutarse en servidores basados en Windows, posee características que promueven la restauración y recuperación de datos. Aunque las tablas individuales no se pueden copiar o restaurar, existen opciones completas de restauración de bases de datos. A través del uso de archivos de registro, almacenamiento en caché, y copias de seguridad, SQL SERVER permite seguridad ya que las opciones de recuperación de desastres son abundantes.

## 4.6. Metodología del desarrollo del Software

El Proceso Unificado Racional o comúnmente llamado RUP, es una metodología cuyo fin es entregar un producto de software. Se estructuran todos los procesos y se mide la eficiencia de la organización.

Es un proceso de desarrollo de software que utiliza el lenguaje unificado de modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

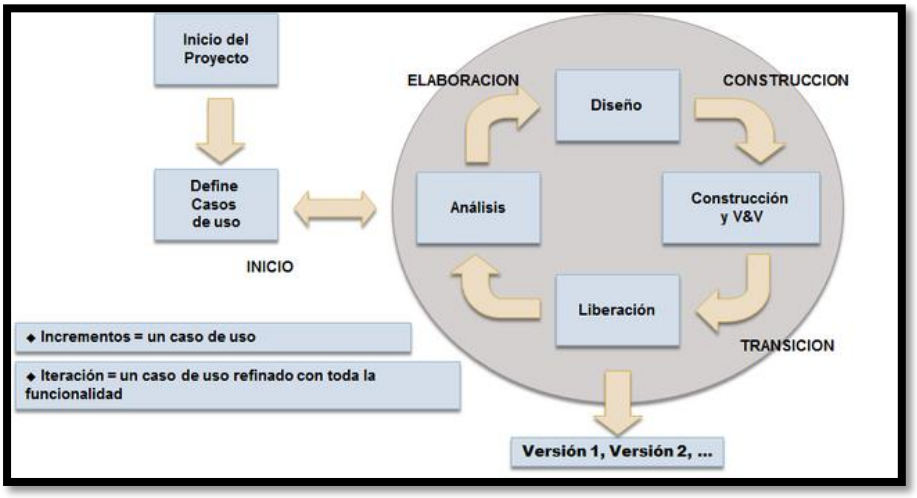



Figura Nº 6: Forma del Trabajo del RUP

El proceso metodológico está basado en el ciclo de vida del RUP, Figura Nº 6, el cual consta de cuatro fases con sus entregables: Inicio, Elaboración, Construcción y Transición.

### 4.6.1. Fases del modelo RUP:

1. Inicio

Esta fase tiene como propósito definir y acordar el alcance del proyecto con los interesados, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores.

2. Elaboración

En la fase de elaboración se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

3. Construcción

El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requisitos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.

4. Transición

El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

### 4.6.2. Características del RUP

- Dirigido por Casos de Uso: Los casos de uso son los artefactos primarios para establecer el comportamiento deseado del sistema.
- Centrado en la Arquitectura: La arquitectura es utilizada para conceptualizar, construir, administrar y evolucionar el sistema en desarrollo.
- Iterativo e Incremental:
- Maneja una serie de entregas ejecutables.
- Integra continuamente la arquitectura para producir nuevas versiones mejoradas.
- Conceptualmente amplio y diverso.
- Enfoque orientado a objetos.
- En evolución continua.
- Adaptable.
- Repetible.
- Permite mediciones: Estimación de costos y tiempo, nivel de avance, etc.

### 4.6.3. Ventajas y desventajas de la metodología propuesta:

| Ventajas | Desventajas |
|---|---|
| ▪ Requiere conocimientos del proceso y de UML.<br>▪ Progreso visible en las etapas tempranas.<br>▪ El uso de Iteraciones (actividades).<br>▪ Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema | ▪ Pretende prever y tener todo el control de antemano.<br>▪ Modelo genera trabajo adicional por abarcar mucha documentación.<br>▪ Genera muchos costos. |

| Ventajas | Desventajas |
|---|---|
| ▪ Facilita la reutilización del código teniendo en cuenta que se realizan revisiones en las primeras iteraciones lo cual además permite que se aprecien oportunidades de mejoras en el diseño. | |

Tabla 1: Ventajas y Desventajas del RUP

Se decidió usar RUP por proveer un entorno de proceso configurable, basado en estándares, dejando en claro el acceso al proceso de desarrollo que se sigue, permitiendo ser adaptable a las necesidades de la institución y del proyecto.

RUP como metodología para ser aplicada en los sistemas basados en la WEB, permite no solo definir una serie de etapas y entregables sino que además garantiza la calidad del producto orientado a la WEB. Esta metodología aumenta la eficiencia en el desarrollo de productos WEB.

Los sistemas y aplicaciones basadas en WEB hacen posible que una población extensa de usuarios finales dispongan de una gran variedad de contenido y funcionalidad.

# 5. Diseño Metodológico

## 5.1. Tipo de Investigación

| | |
|---|---|
| Tipo de investigación: | Exploratoria |
| Enfoque de investigación: | Cualitativa |
| Nivel de investigación: | Descriptivo y correlacional |
| Sujetos que intervienen: | Oficina de acción social, Becados, Solicitantes de servicio social |
| Técnicas de recolección de datos: | Cuestionarios escritos y entrevistas. |
| Instrumentos: | Entrevistas, información documental y observación. |
| Procesamiento de datos: | Herramientas automatizadas como Word y Excel. |

Tabla 2: Tipo de Investigación

El tipo de investigación fue exploratoria, porque se aplica cuando el propósito de la investigación es familiarizarse con un fenómeno o adquirir nuevos conocimientos, con el fin de formular un problema más preciso, los estudios exploratorios (también conocidos como la investigación formulativa) son útiles cuando los problemas se encuentran en una etapa preliminar o cuando el tema o asunto es nuevo.

El enfoque de la presente investigación fue de tipo cualitativa porque es un método de investigación empleado en muchas disciplinas académicas, considerando que principalmente se analizará el tiempo de servicio asignado a los becarios y el análisis de su desempeño en el cumplimiento de las horas becarias para su control en la acumulación de horas demandadas en su programa de formación.

El desarrollo de este trabajo monográfico fue de naturaleza descriptiva y correlacional. Descriptivo porque se describirán las variables relacionadas al programa del servicio social becario, distribución de las asignaciones a los becarios, así como, el suministro de información para la renovación y autorización de beca al estudiante solicitante.

Fue de tipo correlacional porque se analizó el grado y nivel de relación entre las variables control del tiempo de servicio social que realiza el becario y el tiempo de cumplimiento realizado por el estudiante, lo que facilitará la autorización y renovación de beca al estudiante solicitante.

La aplicación web fue desarrollada en ASP.net MVC Express y la base de datos en SQL SERVER EXPRESS. Haciendo uso de MVC cuyo fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores.

En cuanto a la metodología utilizada se optó por aquella que se adaptó más a nuestro medio, conocida como RUP, los requerimientos funcionales son expresados en la forma de Casos de Uso, que guían la realización de una arquitectura ejecutable de la aplicación. Además el proceso focaliza el esfuerzo del equipo en construir los elementos críticos estructuralmente y del comportamiento (llamados Elementos Arquitecturales) antes de construir elementos menos importantes. La mitigación de los riesgos más importantes guía la definición / confirmación del alcance en las primeras etapas del ciclo de vida. Finalmente RUP, particiona el ciclo de vida en iteraciones que producen versiones increméntales de los ejecutables de la aplicación.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Cabe mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

1. ***Disciplina de Desarrollo***

   Basada en la Ingeniería de Negocios entendiendo sus necesidades:

   - Requerimiento: trasladando las necesidades del negocio a un sistema automatizado, mediante el levantado de información a través de entrevistas estructuradas para recopilar la información genérica del sistema y entrevistas a profundidad para recopilar la información particular, las cuales constituyen el marco para la recopilación de las particularidades del proceso del programa de servicio social becario. Adicionalmente, se hará uso del método de observación directa de los procesos que se ejecutan en relación al programa del servicio social becario.
   - Análisis y diseño: trasladando los requerimientos dentro de la arquitectura de software, mediante la selección, organización, análisis y diseño detallado de la información recopilada, para determinar el alcance y las necesidades del sistema web para el programa de servicio social becario.
   - Pruebas: asegurándose que el comportamiento requerido es el correcto y se realizaran conjuntamente con el personal de la Oficina de Servicio Social, con la finalidad de verificar que la aplicación web cumpla con las expectativas deseadas y que todo lo solicitado está presente.
   - Implementación: creando un software que se ajuste a la arquitectura y que tenga el comportamiento deseado, para validar su efectividad en el manejo del programa de servicio social becario, realizando la respectiva capacitación de los usuarios finales.

2. ***Disciplina de Soporte***

Se fundamenta en la configuración y administración del cambio, guardando todas las versiones del proyecto, administrando los horarios y recursos del proyecto, así como también, el ambiente de desarrollo y realizando la distribución, es decir hacer todo lo necesario para la salida del proyecto.

El tipo de estudio fue de corte-transversal ya que los datos de prueba corresponden al segundo semestre del año 2014, para conocer el comportamiento del sistema web con la información del tiempo de servicio social que se hace en la oficina de Acción Social.

## 6. Análisis y presentación de resultados

A continuación se presentan los datos obtenidos de las encuestas aplicadas al personal que trabaja y atiende a los becarios de la UNI.

Se aplicaron un total de 10 encuestas a colaboradores de la Universidad Nacional de Ingeniería.

1. ¿Cuántos becarios realizan tiempo de servicio social becario?

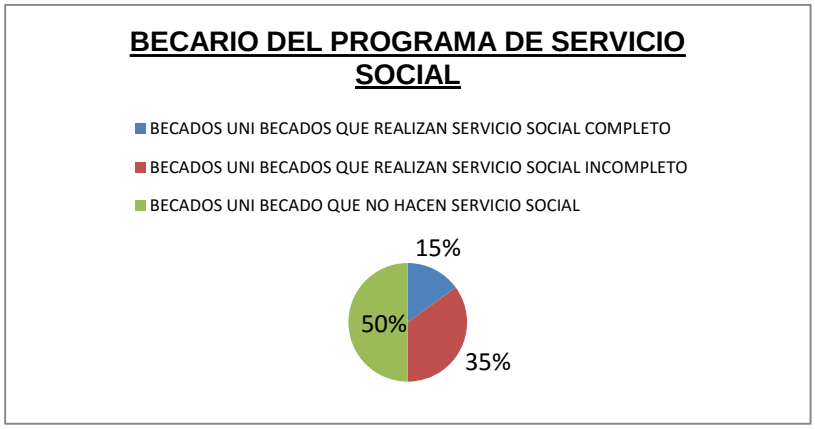



Figura Nº 7: Gráfico de Becarios del Programa Servicio Social

El gráfico anterior refleja que la oficina de Acción social atiende a más de 2000 estudiantes becados por Periodo Académico, de los cuales: solo el 15% realiza servicio social completo; un 35% realiza el servicio social incompleto, es decir, comienzan con su asignación pero no la terminan; y un 50% no realiza servicio social.

2. ¿Por qué los becarios no realizan el servicio social o no lo terminan?

En relación a las causas de incumplimiento del Servicio Social un 59% del incumplimiento del servicio social es causado porque los becarios hacen caso omiso al llamado de la Oficina de Acción Social, el 18% del incumplimiento es causado por falta de tiempo del Becario, el 17% por falta de interés del Becario y el 6% aducen que es por desconocimiento del *Programa del Servicio Social* y el *Reglamento de Becas*.

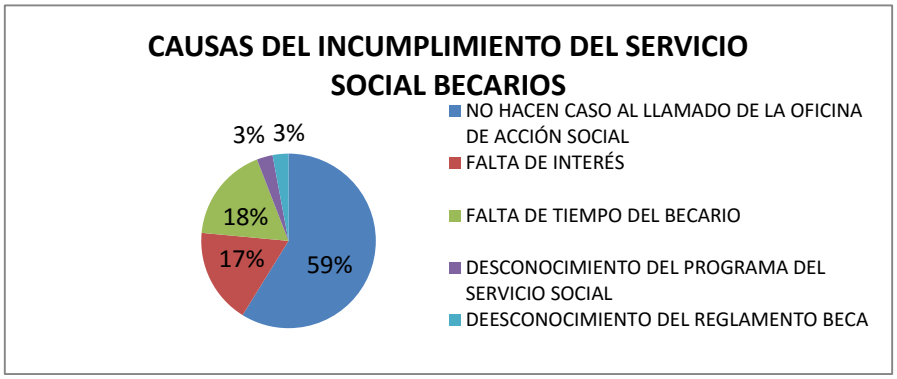



Figura Nº 8: Gráfico de Causas del Incumplimiento del Programa

3. ¿Cuántos Becarios asisten a la Convocatoria que ustedes realizan?

   La oficina de Acción Social realiza una convocatoria a los becarios donde obtiene una captación de la mitad de becados, es aquí donde le dan a conocer el Programa de Servicio Social y el reglamento de becas.

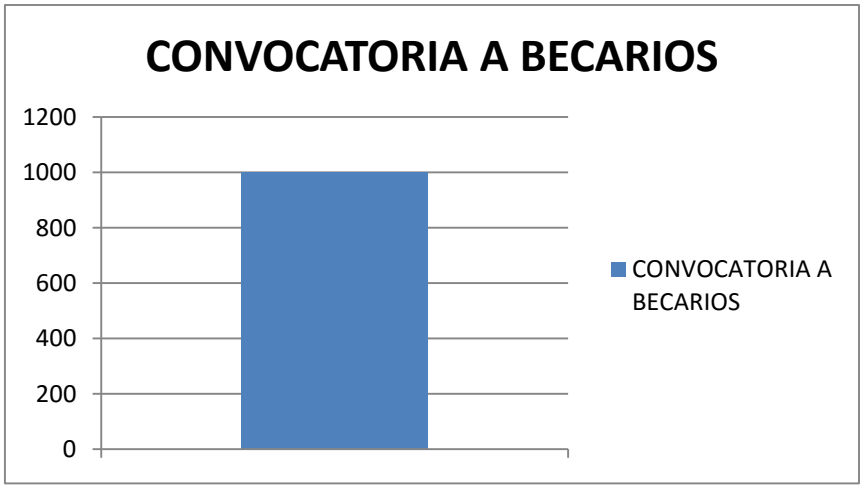



Figura Nº 9: Gráfico sobre Convocatoria a Becados del Programa

4. ¿Qué dificultades presenta la Oficina de Acción Social para poder retener a los becarios y hacerlos cumplir el Servicio Social?

Un 60% de las dificultad que tiene la Oficina de Acción Social es no contar con un Sistema Automatizado que facilite el control y seguimiento del tiempo de Servicio Social efectuado, el 20% equivale a la mala distribución de las tareas, el resto de las dificultades se da por la dificultad de recopilación de la información y el bajo seguimiento que se le brinda al becario.

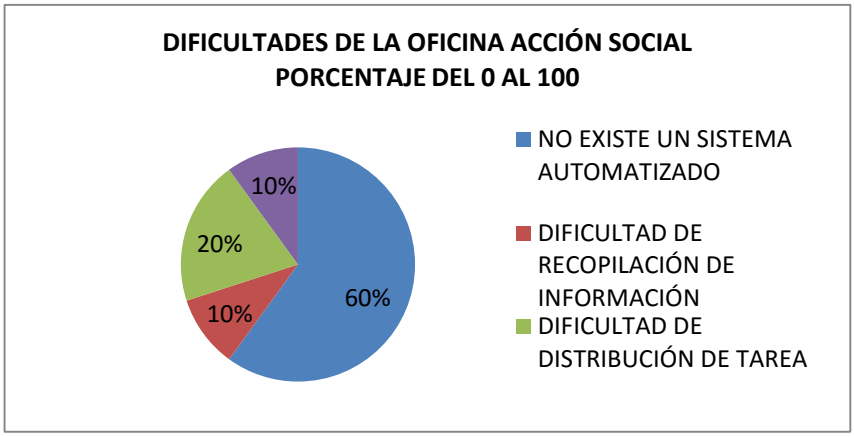



Figura Nº 10: Gráfico Dificultades del Cumplimiento del Programa

5. ¿De qué herramientas se Auxilia la Oficina de Acción Social para llevar el control del Tiempo que realiza el becario y quien apoya en esta labor?

Actualmente la Oficina de Acción Social cuenta con el Formato **"Registro de asistencia becados para el cumplimiento de su servicio social del becario"** para llevar el control de las asistencias que realiza el becario, así mismo, cuenta con el **"Formato de evaluación y desempeño del servicio de becario"** utilizado para evaluar el desempeño de la tarea que se le ha asignado al becario, esto son entregados impresos a los responsables de cada Área, una vez llenos son procesados en los programas MS Word/Excel para su debido análisis. Esta labor es bastante tediosa y tardía por que el proceso se realiza de forma manual.

6. ¿Qué información necesitaría controlar si se implementará un Sistema de Información para el programa de Servicio Social?

En respuesta a esta pregunta se documentaron los requerimientos para la implementación del Sistema de Información, donde la Responsable de la Oficina de Acción Social detallo que es lo que necesita;

A. Un sistema donde estén registrados todos los estudiantes becados.
B. Que se pueda registrar las asistencias de los becarios.
C. Se pueda asignar tareas a los estudiantes y llevar el control de los becarios que no tengan tareas asignadas.
D. Que se pueda evaluar el cumplimiento de las tareas de los becarios.
E. Reportes de la cantidad de estudiantes con tareas.
F. Reportes de las horas por carrera.
G. Reportes de los estudiantes que no realizaron servicio social.
H. Manejar información relevante del estudiante como formas de contacto.

***Observación***

Se llevaron a cabo observaciones del control de asistencia de los becarios en las áreas, se observó que el formato de control de asistencia es ubicado en un lugar visible para los becarios, también se observó que los becarios son los que anotan su hora de entrada y salida.

También se llevó a cabo la observación del procesamiento de la información del Formato de Control de Asistencia en la Oficina de Acción Social donde se observó que el procesamiento se dificulta por no existir una herramienta automatizada para el Control de Asistencia, ya que actualmente el procesamiento se realiza con los programas MS Word/Excel.

***Análisis de los Resultados***

En base a la encuesta y a las observaciones realizadas se encontró que la Oficina de Acción de la Universidad Nacional de Ingeniería necesita con urgencia un Sistema de Información para el Programa de Servicio Social Becario, ya que existe mucho incumplimiento del Programa de Servicio Social por parte de los becarios.

Dada esta situación se plantea el desarrollo del Sistema de Información dividido en tres Etapas de desarrollo del Software como son:

1. Análisis del Sistema de Información.
2. Diseño del Sistema de Información.
3. Implementación del Sistema.

## 6.1. Factibilidad Económica

El costo de desarrollo del sistema se estructura en fases definidas por la metodología utilizada, con el propósito de mostrar, no solo el costo total del proyecto, sino también, detallar el costo de cada una de las fases.

### 6.1.1. Costo del Proyecto

A continuación se detallan los costos necesarios para el desarrollo e implementación del Sistema.

#### 6.1.1.1. Costos de Recursos Humanos

El costo de recurso humano para el desarrollo del proyecto que se tomó como referencia, es el estándar que perciben en la actualidad los ***Analistas Programadores Junior*** en las empresas del país por día, el cual equivale U$24.00 (veinticuatro dólares netos), considerando que realizan la función de análisis, diseño, programación y control de calidad de las aplicaciones.

Cabe señalar que los costos de Recursos Humanos mostrados a continuación son asumidos por el equipo de desarrollo del presente proyecto monográfico:

| Procesos | Duración en días | Costo Por hora ($24) |
|---|---|---|
| **Análisis** | **23** | **$552.00** |
| Estudio Previo | 5 | $120.00 |
| Periodo de entrevistas | 7 | $168.00 |
| Recopilación de los requerimientos | 7 | $168.00 |
| Análisis de la documentación obtenida | 4 | $96.00 |
| | | |
| **Diseño del Sistema** | **25** | **$600.00** |
| Elaboración de Documentación | 5 | $120.00 |
| Diseño de diagramas | 10 | $240.00 |
| Elaboración de base de datos | 5 | $120.00 |
| Corrección y pruebas de la base de datos | 5 | $600.00 |
| | | |
| **Desarrollo del Sistema** | **55** | **$1,320.00** |
| Elaboración de las Interfaces | 10 | $240.00 |
| Programación del sistema | 30 | $720.00 |
| Pruebas del Sistema | 15 | $360.00 |
| | | |
| **Implementación** | **39** | **$936.00** |
| Elaboración del manual de usuario | 12 | $288.00 |
| Revisión y actualización de documento técnico | 20 | $480.00 |
| Corrección de observaciones | 7 | $168.00 |
| **Total** | **142** | **$3,408.00** |

Tabla 3: Costo del Desarrollo del Proyecto

#### 6.1.1.2. Costos de Recursos Hardware y Software

El software no incurrirá en inversión de equipos para su implementación, ya que será instalado en equipos de la Universidad Nacional de ingeniería y se hará uso del software disponible. Por lo tanto, en el siguiente cuadro únicamente se reflejan los costos de equipos y software en que incurrió el equipo de desarrollo del software.

| Cantidad | Descripción | Costo Unitario | Depreciación anual US$ | Costo Total US$ | Descripción |
|---|---|---|---|---|---|
| Hardware | | | | | |
| 1 | Laptop | 1,600.00 | 320.00 | 49.58 | Según el INE, Resolución Nº 16-2001, el costo de depreciación de una computadora es del 20% a 5 años. |
| Software | | | | | |
| 1 | Visual Studio 2015 Comunity | 0 | | 0 | Licencia libre de costos |
| 1 | SQL Server 2008 R2 Express | 0 | | 0 | Licencia libre de costos |
| **Total** | | | | 49.58 | |

Tabla 4: Costos de Recursos Hardware y Software

### 6.1.1.3. Costos de Materiales y Servicios

En la siguiente tabla se presentan los costos incurridos durante el tiempo invertido en el desarrollo del proyecto, se incluyen los costos de uso de equipo considerando el costo de depreciación, los costos de uso de internet utilizado para consulta de información, comunicación (con el tutor/usuario Final/equipo de desarrollo) y validación del sistema y los costos del material de oficina.

| Rubros | Tiempo (Días) | Total de Horas | Total Meses de 22 días | Costo Unitario (US$) | Costo Total (US$) | Descripción |
|---|---|---|---|---|---|---|
| Uso de Equipos | 142 | 1,136 | | 2.4238 | $2,753.44 | Equivale a 8 horas diarias por costo promedio de 2.4238 (C$/kWh) estipulado por el Ente Regulador |
| Uso de Internet | 142 | | 6.454545 | $23.99 | $154.84 | Costo mensual según estudio realizado por la Unión Internacional de Comunicaciones |
| Material de Oficina | | | | | $40.00 | |
| **Total** | | | | | $2,948.28 | |

Tabla 5: Costos de Servicios y Materiales

| *Materiales* | *Cantidad* | *Costo Total US$* |
|---|---|---|
| *Papel Bond* | 1 | $5.00 |
| *Lápices* | 2 | $1.00 |
| *Cuadernos* | 2 | $4.00 |
| *USB* | 2 | $30.00 |
| | ***Total*** | ***$40.00*** |

Tabla 6: Materiales de Oficina

### 6.1.1.4. Costo total del Proyecto

El costo total de desarrollo del proyecto es de $ 6,405.86 (seis mil cuatrocientos cinco dólares con ochenta y seis centavos netos), incluyendo los costos de Recurso Humano. El detalle se muestra en el siguiente cuadro:

| Descripción | Tiempo total (Días) | Costo Total US$ |
|---|---|---|
| Recurso Humano | 142 | 3,408.00 |
| Hardware y Software | | 49.58 |
| Servicios y Materiales | | $2,948.28 |
| **Total** | **142** | **$ 6,405.86** |

Tabla 7: Costo Total de Proyecto Incluyendo RRHH

El costo real de desarrollo del proyecto sin incluir los costos de Recurso Humano, que serán asumidos por los monografistas, es de $ 2,997.86 (dos mil novecientos noventa y siete dólares con ochenta y seis centavos. El detalle se muestra en el siguiente cuadro:

| Descripción | Tiempo total (Días) | Costo Total US$ |
|---|---|---|
| Recurso Humano | 142 | 0.00 |
| Hardware y Software | | 49.58 |
| Servicios y Materiales | | $2,948.28 |
| **Total** | **142** | **$2,997.86** |

Tabla 8: Costo Total de Proyecto sin Incluir RRHH

## 6.2. Análisis del Sistema de Información

Para modelar el “Sistema web del programa del servicio social becario de la Universidad Nacional de Ingeniería”, se analizaron los procesos manuales de la Oficina de Acción Social para la definición de los requerimientos de usuario, así como también, se clasificaron las entradas y salidas.

### 6.2.1. Descripción de los procesos del programa de servicio social becario

La oficina de Acción Social realiza cinco grandes procesos para llevar a cabo su labor y son los siguientes:

1. Recepción de Solicitudes de Requerimientos de Servicio Social.
2. Solicitud de actividades por parte de los Becarios.
3. Registro de Asistencia.
4. Evaluación y desempeño del Programa de Servicio Social Becario.
5. Elaboración de reportes de resultados del programa de servicio social becario.

#### 6.2.1.1. Recepción de solicitudes de requerimiento de servicio social

El proceso de Recepción de Solicitudes de Requerimiento de Servicio Social es la parte medular de la Oficina de Acción Social.

El proceso inicia con la solicitud de Becarios, de las diferentes áreas de la Universidad para que estos realicen actividades que asignarán los responsables de cada área, para que los becarios cumplan con la "**Normativa del Servicio Social Becario".**

Las solicitudes de Becarios son recepcionadas en la Oficina de Acción Social a cargo de la Ing. Giselle Calero, quién después de analizar todas las solicitudes, asigna al Becario a la respectiva instancia donde tendrá que realizar las actividades que se le encomendarán en un horario consensuado entre el responsable del área y el becario.

Cada instancia a la que se le asigna el becario, es responsable de orientar y supervisar, que las actividades y/o tareas sean realizadas de manera Excelente o Eficiente.

#### 6.2.1.2. Solicitud de servicio social por parte de los becarios

El proceso de Solicitud de Servicio Social por parte de los becarios, es una alternativa que brinda la Oficina de Acción Social a todos aquellos becarios, cuyo interés es realizar un proyecto o integrarse a uno ya existente en pro de la Universidad Nacional de Ingeniería.

En el caso que el becario se integre a un proyecto existente se le asignará un tutor para que este supervise las actividades o tareas que se le asignen y garantice que sean desarrolladas de forma eficiente.

Cuando el becario realiza un proyecto de índole personal este deberá entregar avances en la oficina de acción social para garantizar que el proyecto sea realizado.

Así mismo, la Oficina de Acción Social les hace saber a los becarios que existen actividades extracurriculares fuera de la Institución en donde pueden realizar tiempo de servicio social becario, por ejemplo:

1. Ferias Tecnológicas
2. Pipitos
3. Teletón.

#### 6.2.1.3. Registro de asistencia de becarios

El Proceso de Registro de Asistencia de los becarios, se lleva a cabo en la instancia donde fue asignado el becario. Cada vez que el becario llega a realizar una actividad o tarea en un tiempo determinado, este debe de registrar su asistencia en el Formato **"REGISTRO DE ASISTENCIA BECADOS PARA EL CUMPLIMIENTO DE SU SERVICIO SOCIAL DEL BECARIO",** siempre bajo la responsabilidad y supervisión del responsable o tutor del becario.

Los datos que el becario registra en el formato son los siguientes:

1. Carrera
2. Nombres y Apellidos
3. Carne
4. Hora de entrada y Salida
5. Firma

Una vez que el becario completo el formato este es enviado a la Oficina de Acción Social donde la información es procesada a Word/Excel.

#### 6.2.1.4. Evaluación y desempeño del programa de servicio social becario

El Proceso de Evaluación y Desempeño del Servicio Social que realiza el becario, es importante para la oficina de Acción Social, ya que con esto se puede medir la actitud del estudiante y su desempeño en el trabajo realizado. La objetividad con la que se efectúa la evaluación determina mejoras en el Servicio Social Becario.

Este proceso se efectúa gracias a la supervisión de los responsables y tutores de los becarios, que garantizan que se cumplan los acuerdos de las actividades o tareas encomendadas a los becarios.

Los Beneficiados del Servicio Social reportan de manera objetiva y transparente mediante el **"Formato de evaluación y desempeño del servicio de becario"** registrando la siguiente información:

1. Nombre del Estudiante
2. Carne
3. Carrera
4. Área
5. Beneficiario del Servicio
6. Período Académico
7. Fecha
8. Tareas
9. Desempeño
10. Aspectos Generales (Desempeño Laboral, Factor Humano, Habilidades)
11. Horas Trabajadas

Una vez completado el formato es enviado a la Oficina de Acción Social, para su análisis y toma de decisiones.

#### 6.2.1.5. Presentación de resultados del programa de servicio social becario

Para la Dirección de Bienestar Estudiantil, el servicio social becario promueve en el estudiante el sentido de la responsabilidad y el compromiso que poco a poco este espacio va propiciando en él; adicionalmente el estudiante, descubre sus potencialidades y limitaciones, desarrolla la habilidad de comunicarse y de interactuar con el otro. La conciencia social va surgiendo en el momento en que el Estudiante descubre cómo su comportamiento y sus actitudes le afectan no sólo a él mismo, sino también a su grupo de compañeros.

Dada la importancia del servicio social becario para la Dirección de Bienestar Estudiantil, la presentación de resultados es vital, es por eso que cada Periodo Académico, la Oficina de Acción Social elabora reportes de los resultados del Servicio Social Becario.

Estos reportes son los resultados del servicio social de los becarios, por carrera y por semestre, los cuales son remitidos al Director de Bienestar Estudiantil, para ser publicados en todos los murales de la comunidad universitaria.

### 6.2.2. Requerimientos de Usuarios

Después del análisis, recopilación de información y procesamiento de la misma, elaboramos la lista de requerimientos para el sistema web de información, se estratificaron en Catálogos, Asignaciones y Reportes para su mejor análisis y ordenamiento de prioridades, los cuales se detallan a continuación:

#### 6.2.2.1. Catálogos

- Generales
  - Usuarios
  - Roles
  - Recinto
  - Facultad
  - Carrera
  - Beca (Tipo)
  - Periodo académico
  - Aspectos generales a Evaluar
  - Tipo de Tarea
  - Área
  - Beneficiario
  - Evaluación

- Programa de Servicio Social
  - Tarea
  - Estudiante (Becario)

#### 6.2.2.2. Asignaciones

- Programa de Servicio Social de Becario
  - Asignación de Tareas al Becario
  - Asignación de Periodo-Carrera
  - Registrar Asistencia del Becario
  - Evaluación de Tareas

#### 6.2.2.3. Reportes

- Programa de Servicio Social Becario
  - Catálogo de Estudiante
  - Asignaciones de Tareas
  - Asistencias de los becarios por periodo especifico
  - Asistencia de los becarios por periodo académico
  - Evaluación de tareas por periodo académico
  - Total de horas de servicio social por carrera
  - Total de horas de servicio social por becario
  - Horas de servicio social del becario

#### 6.2.2.4. Requerimientos de restricción (seguridad)

- Clave de seguridad por rol de usuario del sistema
- Roles de usuarios según su participación en el sistema
- Encriptación de las claves de los usuarios
- Almacenamiento de usuario, fecha y hora en las tablas de la base de datos.

#### 6.2.2.5. Definición de roles

El rol que el usuario desempeña dentro del Sistema SISBECA (Sistema del Servicio Becario) es de suma importancia, ya que el sistema fue construido para satisfacer las necesidades de los usuarios finales, en función de los objetivos estratégicos de la oficina de Acción Social. A continuación se identifican los roles de usuario definidos en el Sistema SISBECA:

- ☆ **Administrador**: Los usuarios a los que se les asigna el rol de administrador, tienen acceso a todas las opciones que comprende el sistema. (Responsable de Oficina de Acción Social)
- ☆ **Primario**: Este rol se le asigna a todos aquellos usuarios encargados de ingresar, modificar, eliminar y consultar reportes contables del sistema (Usuario delegado por el Responsable de la Oficina de Acción Social).

- ☆ **Secundario**: Este rol se le asigna a todos aquellos usuarios encargados de ingresar, modificar y eliminar ciertos datos del sistema (Usuario delegado por el Responsable de la Oficina de Acción Social).
- ☆ **Restringido:** Este rol se le asigna a todos aquellos usuarios que consultan información del sistema (Estudiante o Becario).

### 6.2.3. Clasificación de las entradas (automáticas, no automáticas)

#### 6.2.3.1. Automáticas

Las entradas automáticas son aquellas que provienen de otros sistemas. El Sistema SISBECA (Sistema del Servicio Becario) no contará con entradas automáticas.

#### 6.2.3.2. No automáticas

Las entradas no automáticas son aquellas que se proporcionan de forma directa por el Usuario, las definidas en el sistema son las siguientes:

- ☆ Catalogo Usuarios
- ☆ Catalogo Roles
- ☆ Catalogo Recinto
- ☆ Catalogo Facultad
- ☆ Catalogo Carrera
- ☆ Catalogo Periodo académico
- ☆ Catalogo Beca
- ☆ Catalogo Estudiante (Becario)
- ☆ Catalogo Teléfono
- ☆ Catalogo Tarea
- ☆ Catalogo Tipo Tarea
- ☆ Catalogo Área
- ☆ Catalogo Beneficiario
- ☆ Catalogo Evaluación
- ☆ Catalogo Aspectos Generales a Evaluar

- ☆ Asignación de Tareas al Becario
- ☆ Asignación de Periodo-Carrera
- ☆ Registrar Asistencia del Becario
- ☆ Evaluación de Tareas

### 6.2.4. Clasificación de las Salidas (Automáticas y No Automáticas)

#### 6.2.4.1. Automáticas

Las Salidas automáticas son aquellas que alimentan y provienen de otros sistemas. El Sistema SISBECA (Sistema del Servicio Becario) no contará con salidas automáticas.

#### 6.2.4.2. No automáticas

Las salidas no automáticas son aquellas que no alimentaran a ningún otro sistema dentro de la Institución, estas salidas son:

- ☆ Catálogo de Estudiante
- ☆ Listado de Estudiantes becados
- ☆ Listado de tareas ya asignadas a becarios
- ☆ Listado de los becarios por periodo especifico
- ☆ Listado de los becarios por periodo académico
- ☆ Evaluación de tareas por periodo académico
- ☆ Listado de las tareas evaluadas
- ☆ Listado de la evaluación de los becarios
- ☆ Total de horas de servicio social por carrera
- ☆ Total de horas de servicio social por becario
- ☆ Horas de servicio social del becario

### 6.2.5. Requerimientos funcionales

- ☆ Control de Usuarios
- ☆ Gestionar Catalogo Recinto
- ☆ Gestionar Catalogo Facultad
- ☆ Gestionar Catalogo Carrera
- ☆ Gestionar Catalogo Periodo académico
- ☆ Gestionar Catalogo Beca
- ☆ Gestionar Catalogo Estudiante (Becario)
- ☆ Gestionar Catalogo Teléfono
- ☆ Gestionar Catalogo Tarea
- ☆ Gestionar Catalogo Tipo Tarea
- ☆ Gestionar Catalogo Área
- ☆ Gestionar Catalogo Beneficiario
- ☆ Gestionar Catalogo Evaluación
- ☆ Gestionar Asignación Periodo-Carrera
- ☆ Gestionar Catalogo de Aspectos Generales a Evaluar
- ☆ Registrar, editar y cancelar las asignaciones de tareas al Becario
- ☆ Registrar, editar y cancelar la asistencia del Becario
- ☆ Gestionar evaluación de tareas
- ☆ Emitir reportes del programa de servicio social

### 6.2.6. Requerimientos no funcionales

- ☆ Análisis de Asistencia y cumplimiento del programa de servicio social becario
- ☆ Proyecciones del cumplimiento del programa de servicio social becario
- ☆ Asignaciones de beca al estudiante
- ☆ Aprobaciones o retiros de becas
- ☆ Análisis de la evaluación del programa de servicio social becario
- ☆ El Sistema no garantiza que el becario realice el servicio social.
- ☆ El Sistema no garantiza el seguimiento o control de niveles de cumplimiento del servicio social becario.

### 6.2.7. Descripción del sistema de información

El sistema a desarrollar para la **Universidad Nacional de Ingeniería "UNI"** tiene su grado de complejidad debido a que debe de ser parametrizado para que pueda acoplarse fácilmente a la forma de trabajo de la Institución y satisfacer las necesidades de los usuarios finales.

La Dirección de Bienestar Estudiantil de la Universidad Nacional de Ingeniería es una instancia que contribuye a la formación integral de los estudiantes, esta a su vez tiene a cargo la oficina de Acción Social, responsable de la atención, asignación y seguimiento del programa de servicio social becario que debe realizar el estudiante. El sistema SISBECA (Sistema del Servicio Becario) está dirigido a la Oficina de Acción Social y no se enlaza con ningún otro sistema.

El sistema SISBECA realizará las siguientes acciones:

Registrar y editar los datos de los Alumnos, facultades, recintos, carreras, áreas, periodos académicos, etc.

Contar con el módulo de control de Usuarios el cual está disponible para el administrador del sistema, permite agregar nuevos usuarios y roles.

El segundo Modulo es la base del sistema donde se debe crear el registro de los estudiantes, periodos académicos, carreras, facultades, recintos, áreas, becas y tareas; de igual manera el Sistema permite editar, cambiar de estado y dar de baja a dichos datos. En la opción de Asignaciones permite crear, editar, anular las asistencias, asignaciones y evaluaciones de los becarios.

Una de las herramientas que el usuario utilizará con mucha frecuencia es la pantalla de impresión de reportes, la que permite imprimir todo lo que se ha grabado en el sistema, para ver de manera detallada y especifica los datos que considere pertinente.

## 6.3. Arquitectura del Sistema de Información

### 6.3.1. Arquitectura de la solución

En esta sección el diseño a alto nivel y los paradigmas arquitectónicos, evaluados para posteriormente presentar la arquitectura final. Para mayores referencias, revisar el Anexo I: Diagramas de Secuencia y Anexo A: Documento de Manual de Usuario.

#### 6.3.1.1. Representación de la arquitectura

La arquitectura está orientada a entornos Web. Bajo este diseño las tareas se ejecutan por el lado del servidor, evitando delegar tales responsabilidades hacia las máquinas clientes desde sus navegadores. Asimismo, asegura la disponibilidad a tiempo completa con conexión a Internet. Es así como el diseño debe garantizar un óptimo aprovechamiento de las capacidades propias de los sistemas Web satisfaciendo adecuadamente los requisitos no funcionales del sistema. Entre las fortalezas exigidas a la arquitectura se encuentran (Ver Figura N°5).

#### 6.3.1.2. Diseño de la arquitectura de la solución

Para la implementación de esta solución se aplicará la arquitectura en N-Capas, Debido a su diseño altamente escalable ante la incorporación de nuevos módulos y funcionalidades a futuro. Además posibilita la distribución de componentes (capas) entre varios niveles de hardware, obteniendo mayor seguridad y rendimiento ante numerosas peticiones al servidor Web. Esta arquitectura orientada a objetos no presenta obstáculos para adaptar tanto el patrón de modelo de dominio en la capa de lógica de negocio como el patrón de repositorio en la capa de acceso a datos, cumpliendo así con los lineamientos base de diseño indicados a comienzos del capítulo. La arquitectura queda dividida en cuatro capas descritas a continuación (ver figura N°11):

### 6.3.1.3. Vista Lógica

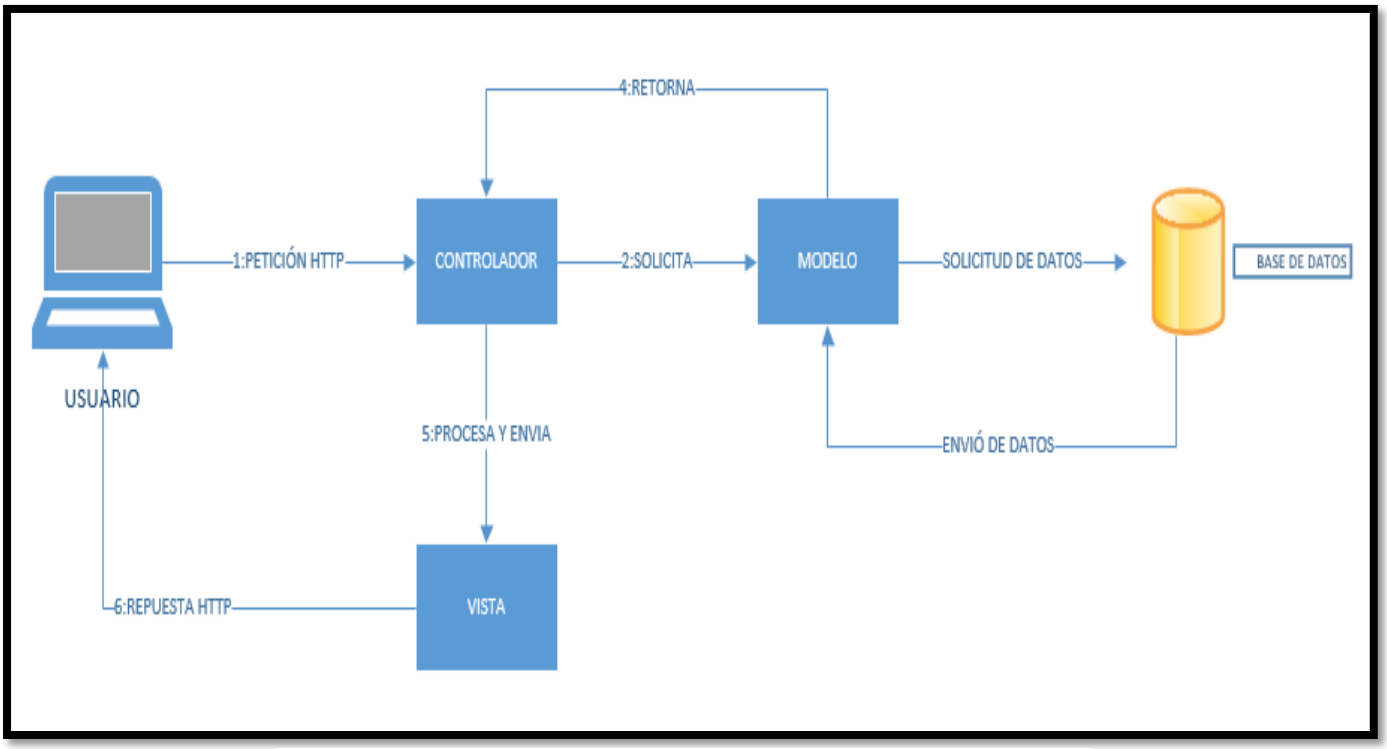



Figura Nº 11: Vista Lógica del Sistema

### 6.3.1.4. Vista de Despliegue

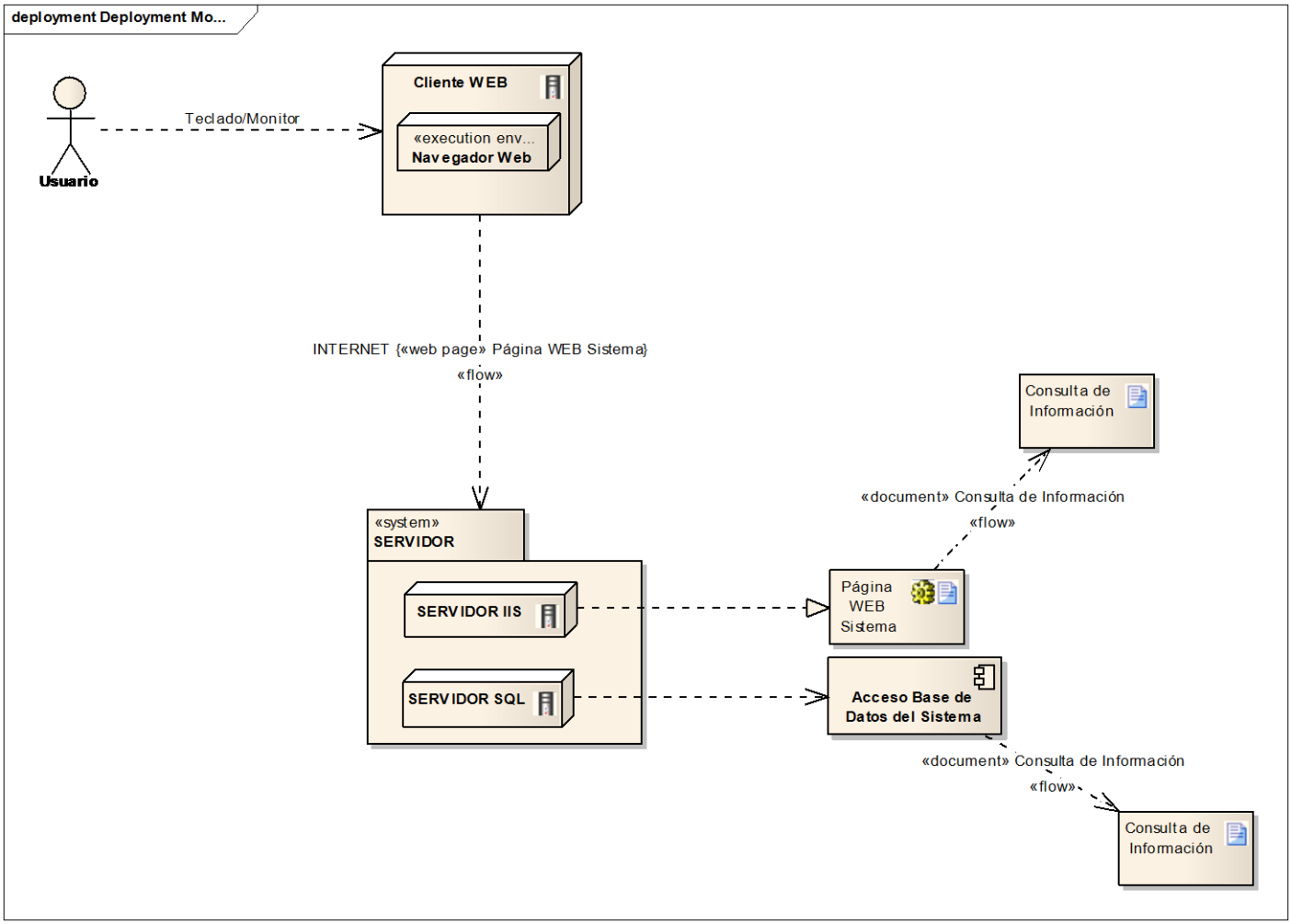



Figura Nº 12: Vista de Despliegue

### 6.3.1.5. Diagrama de Clases

A continuación se presenta el diagrama de clases del Sistema del Programa de Servicio Social Becario. En primer lugar las clases de diseño representan a las entidades de negocio identificadas en la etapa de análisis, con sus atributos y tipos de datos utilizados. En segundo lugar se representan a las clases cuyos métodos más importantes tienen a cargo la implementación de la lógica de negocio:

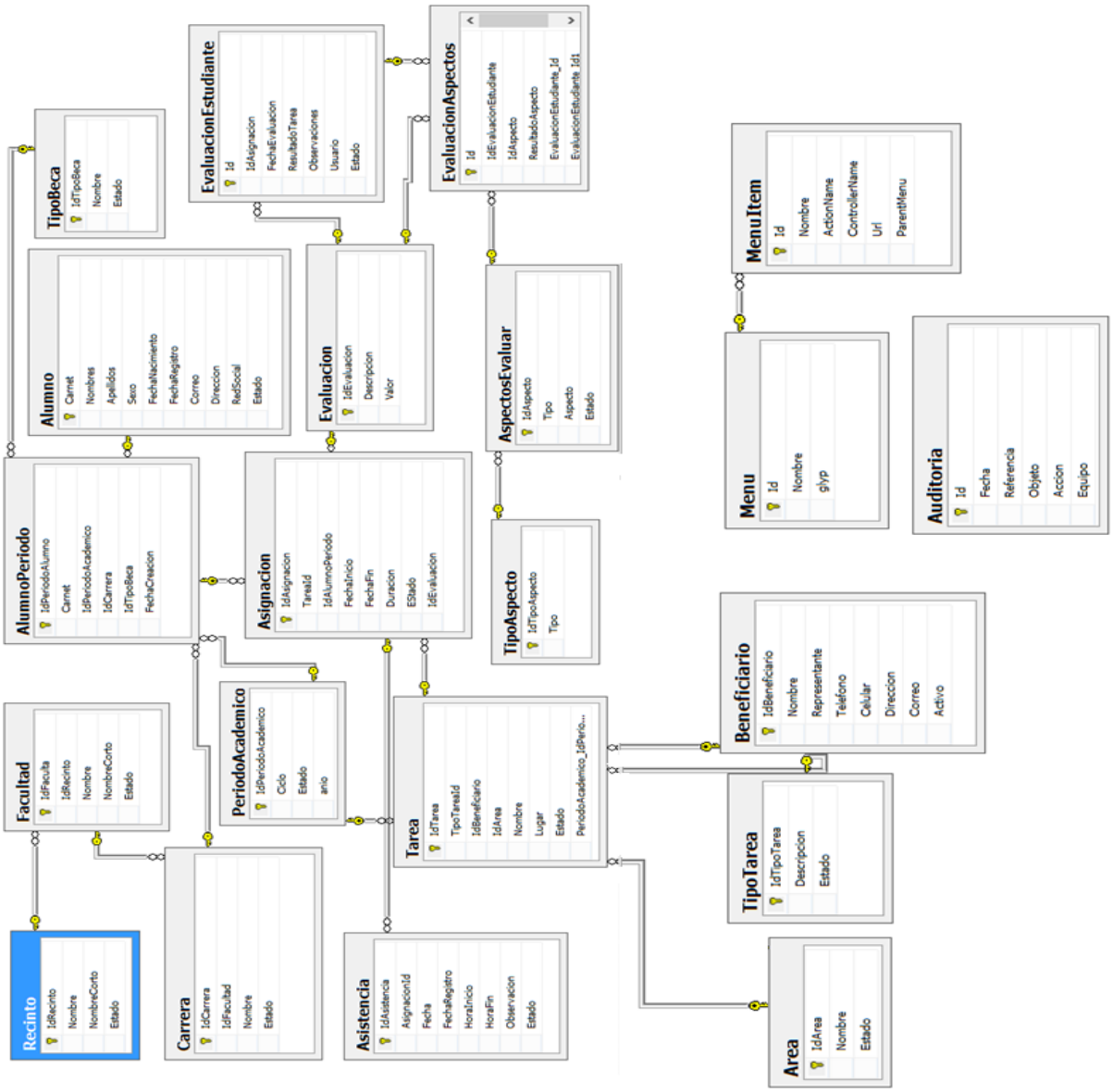



Figura Nº 13: Diagrama de clases del Sistema del Programa del Servicio Social Becario

### 6.3.1.6. Modelo Entidad – Relación

Se presenta a continuación en la figura N° 14, las principales tablas del diagrama de base de datos para las operaciones del sistema. El diccionario de datos se encuentra en el Anexo G: Diccionario de Datos.

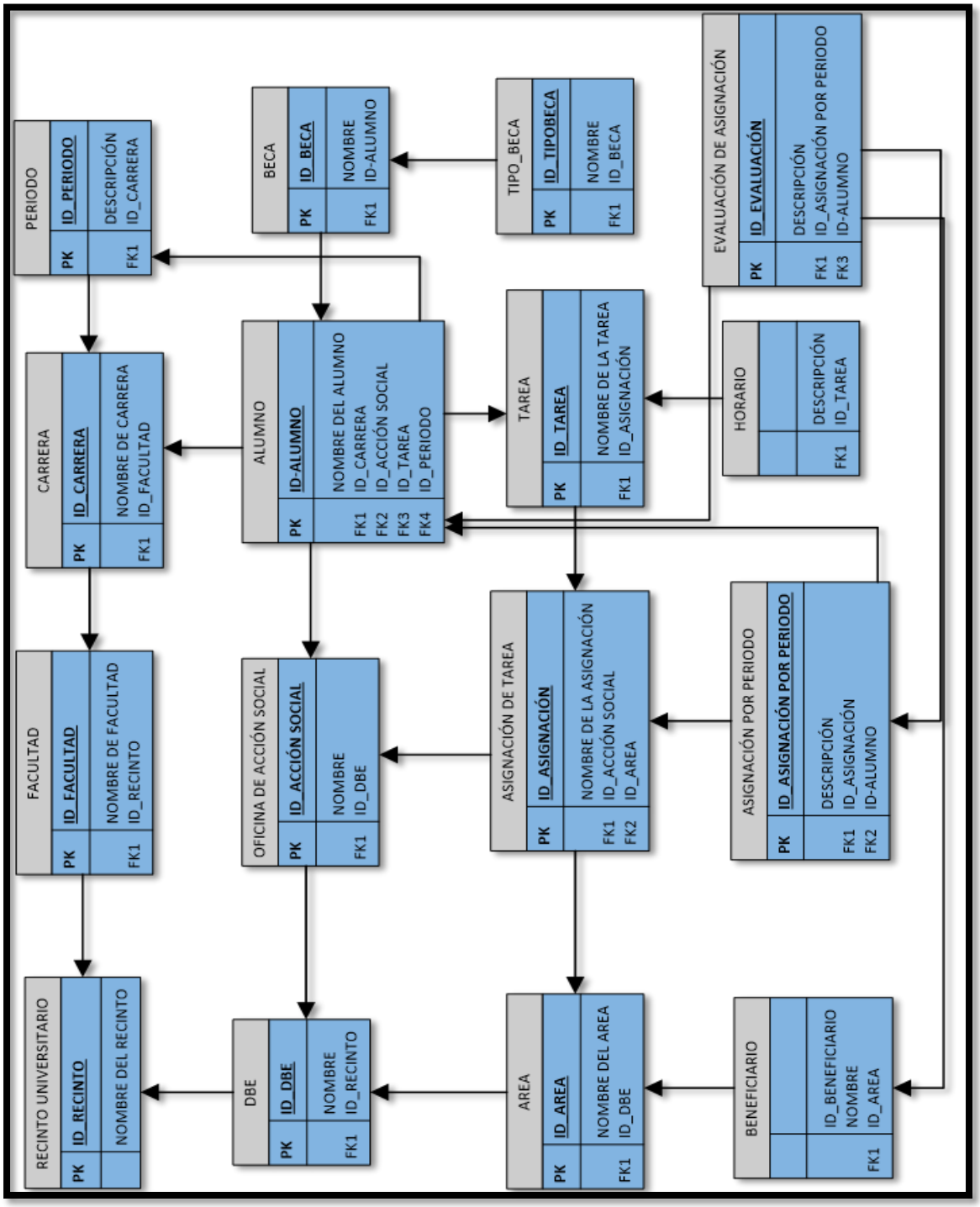



Figura Nº 14: Diagrama Entidad Relación Sistema del Programa del Servicio Social Becario

### 6.3.1.7. Caso de uso general del sistema

A continuación se presenta el diagrama de casos de uso general del Sistema del Programa de Servicio Social Becario:

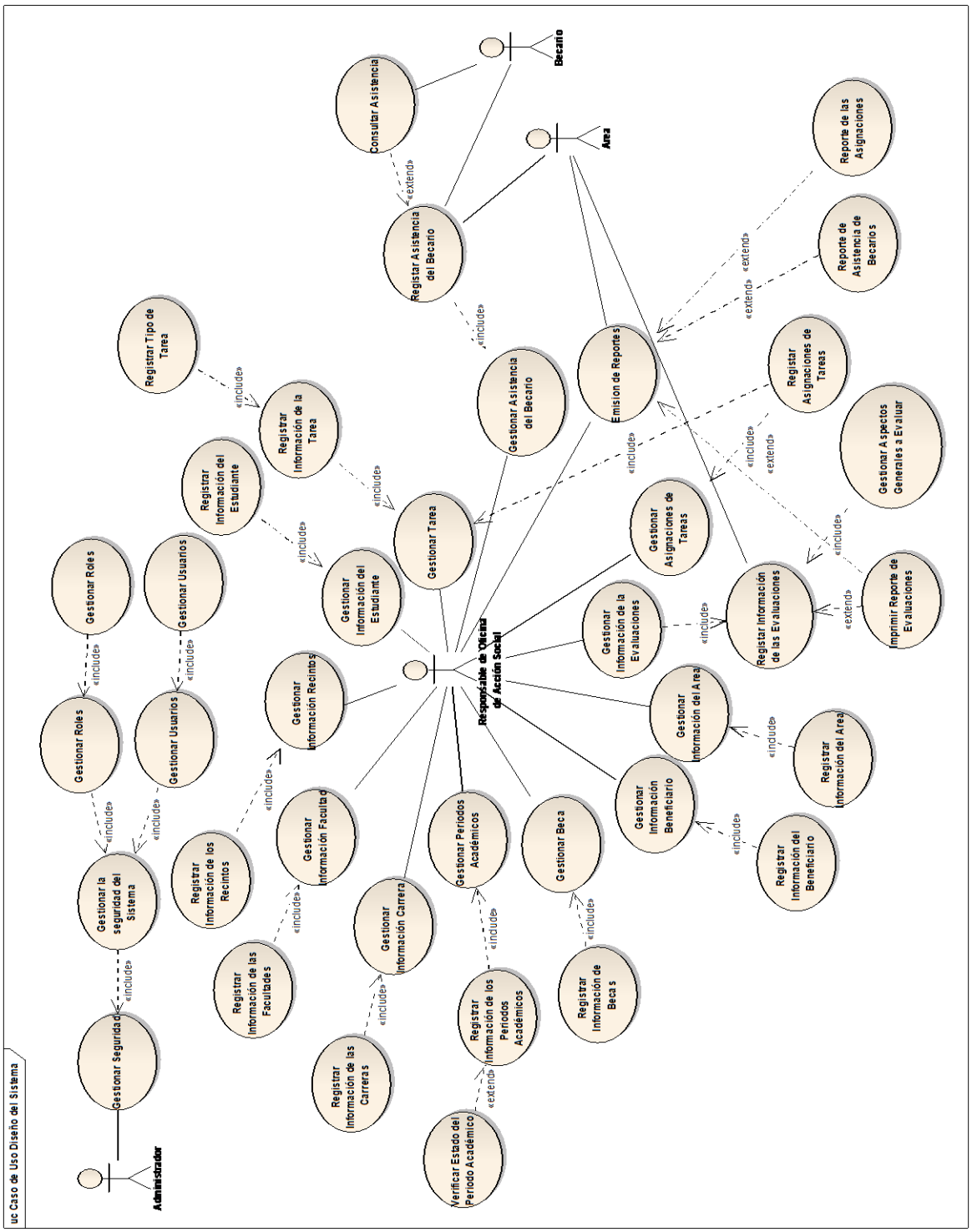



Figura Nº 15: Caso de Uso General del Sistema del Programa del Servicio Social Becario

### 6.3.1.8. Descripción de casos de uso del sistema

Se presentan a continuación la descripción de tres diagramas de casos de uso, correspondiente a los procesos de gestionar asistencia de becario, gestionar evaluaciones de tareas y gestionar evaluaciones de aspectos generales del estudiante. La relación completa de casos de uso del sistema se ubican en el Anexo H: Descripción de Casos de Usos.

#### 6.3.1.8.1. Gestionar Asistencia del Becario

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU1)** | : | Gestionar Asistencia del Becario | | |
| **Definición** | : | Registra la Asistencia del Becario | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| ***ACTORES*** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social y Área | | Encargado de registrar los datos de las Asistencia del Becario | | |
| ***ESCENARIOS*** | | | | |
| **Nombre** | : | Registro de Asistencia del Becario | | |
| **Pre-Condiciones** | : | Oficina de Acción Social debe tener registrado al Becario, Área, Beneficiario y haberle asignado una tarea al becario. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social y Área | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social y Área | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social y/o Área, pueden registrar exitosamente los datos de la Asistencia al Becario | | |
| **Operaciones** | : | 1. Selecciona menu Servicio Social<br>2. Dar Click en el submenu Registro de Asistencia<br>3. Dar Click en Registrar Asistencia<br>4. LLenar el formulario correspondiente a los datos de Asistencia<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | Guardar : Datos incorrectos. La base de datos no ingresa los datos de la Asistencia del Becario | | |

Tabla 9: Caso de uso Gestionar Asistencia del Becario

#### 6.3.1.8.2. Gestionar Evaluaciones de Tarea

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU2)** | : | Gestionar Evaluaciones de Tarea | | |
| **Definición** | : | Registra las Evaluaciones de Tareas | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| ***ACTORES*** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social y Área | | Encargado de registrar los datos de las Evaluaciones de la tareas que realizan los Becarios | | |
| ***ESCENARIOS*** | | | | |
| **Nombre** | : | Registro de Evaluación de Tarea | | |
| **Pre-Condiciones** | : | Oficina de Acción Social debe tener registrado al Becario, Área, Beneficiario, haberle asignado una Tarea | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social y Área | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social y Área | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social y/o Área, pueden registrar exitosamente los datos de la Evaluación de la Tarea | | |
| **Operaciones** | : | 1. Selecciona menu Servicio Social<br>2. Dar Click en el submenu Evaluación de Tarea<br>3. Dar Click en agregar Evaluación de Tarea<br>4. LLenar el formulario correspondiente a los datos la Evaluación<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | Guardar : Datos incorrectos. La base de datos no ingresa los datos de la Evaluación de la Tarea que realiza el Becario | | |

Tabla 10: Caso de uso Gestionar Evaluación de Tarea

### 6.3.1.8.3. Gestionar Evaluaciones de Aspectos Generales

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU3)** | : | Gestionar Evaluaciones de Aspectos Generales del Estudiante | | |
| **Definición** | : | Registra las Evaluaciones de los Aspectos Generales del Desempeño del Servicio Social del Estudiante. | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| ***ACTORES*** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social y Área | | Encargado de registrar los datos de las Evaluaciones de los Aspectos Generales del Desempeño del Servicio Social del Estudiante. | | |
| ***ESCENARIOS*** | | | | |
| **Nombre** | : | Registro de Evaluación Aspectos Generales | | |
| **Pre-Condiciones** | : | Oficina de Acción Social debe tener registrado al Becario, Área, Beneficiario, haberle asignado una Tarea | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social y Área | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social y Área | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social y/o Área, pueden registrar exitosamente los datos de la Evaluación de la Tarea | | |
| **Operaciones** | : | 1. Selecciona menu Servicio Social<br>2. Dar Click en el submenu Evaluación de Tarea<br>3. Dar Click en agregar Evaluación de Tarea<br>4. LLenar el formulario correspondiente a los datos la Evaluación<br>5. Dar Click en Guardar<br>6. Se Activará el Formulario Correspondiente a los Aspectos generales a Evaluar.<br>7. Dar Click Guardar | | |
| **Excepciones** | : | Guardar : Datos incorrectos. La base de datos no ingresa los datos de la Evaluación de la Tarea que realiza el Becario | | |

Tabla 11: Caso de uso Gestionar Evaluación Aspectos Generales

### 6.3.1.9. Diagramas de Secuencia

Se presentan a continuación tres diagramas de secuencia correspondiente a los procesos de gestionar asistencia de becario, gestionar evaluaciones de tareas y gestionar evaluaciones de aspectos generales del estudiante. El propósito es representar gráficamente la interacción entre las capas del software conforme con las acciones del usuario. La relación completa de diagramas se ubica en el Anexo I: Diagrama de Secuencia del sistema.

#### 6.3.1.9.1. Diagrama de secuencia de proceso para gestionar asistencia de becario.

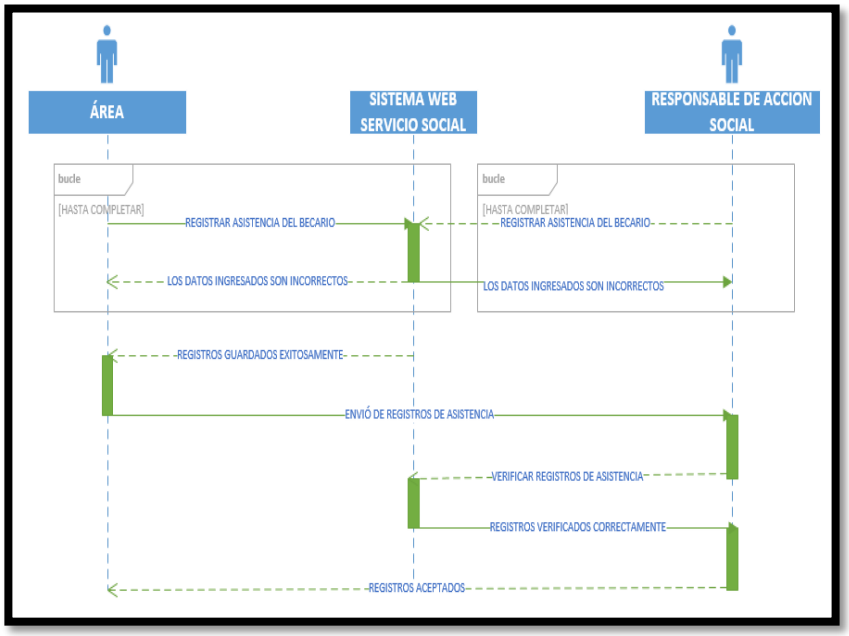



Figura Nº 16: Diagrama de secuencia de proceso para gestionar asistencia de becario

### 6.3.1.9.2. Diagrama de secuencia de proceso para gestionar evaluaciones de tareas.

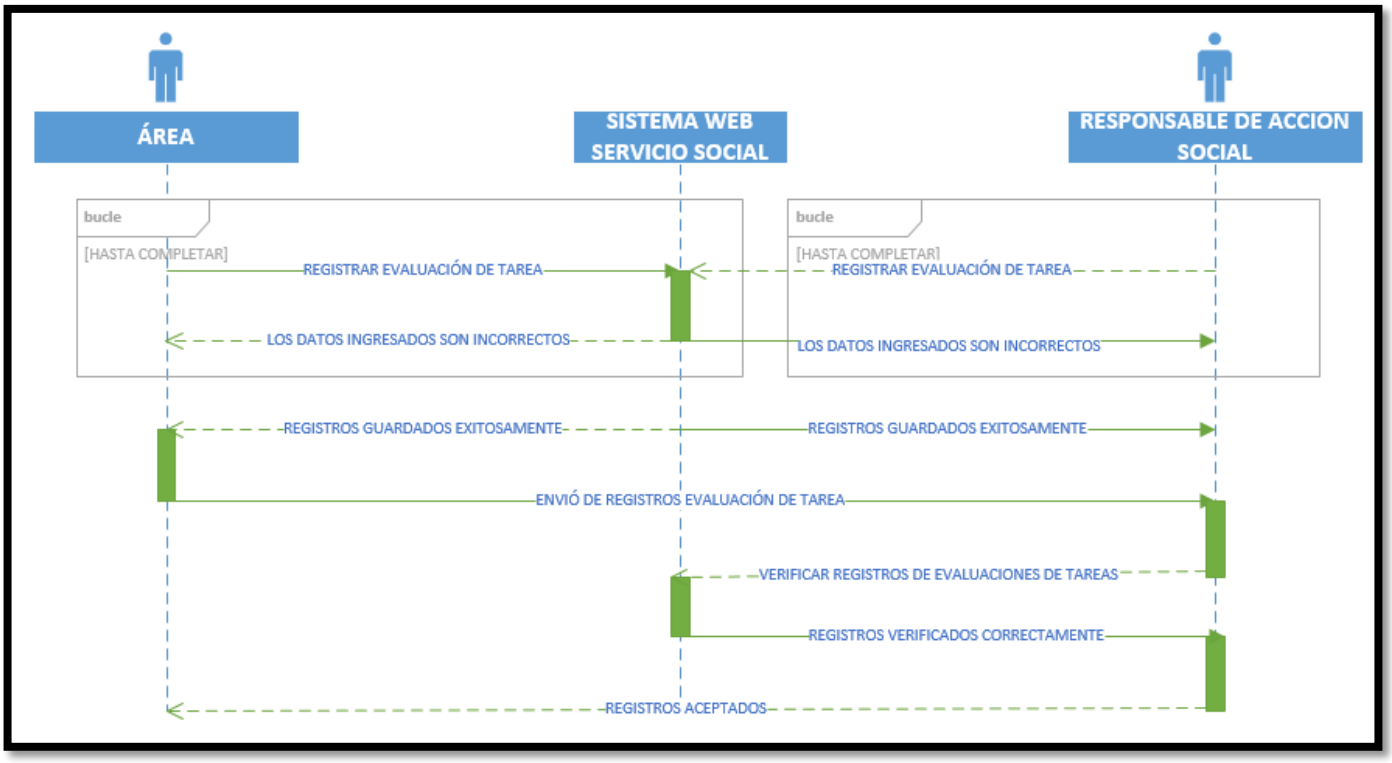



Figura Nº 17: Diagrama de secuencia de proceso para gestionar evaluaciones de tareas.

### 6.3.1.9.3. Diagrama de secuencia de proceso para gestionar evaluaciones de aspectos generales del Becario.

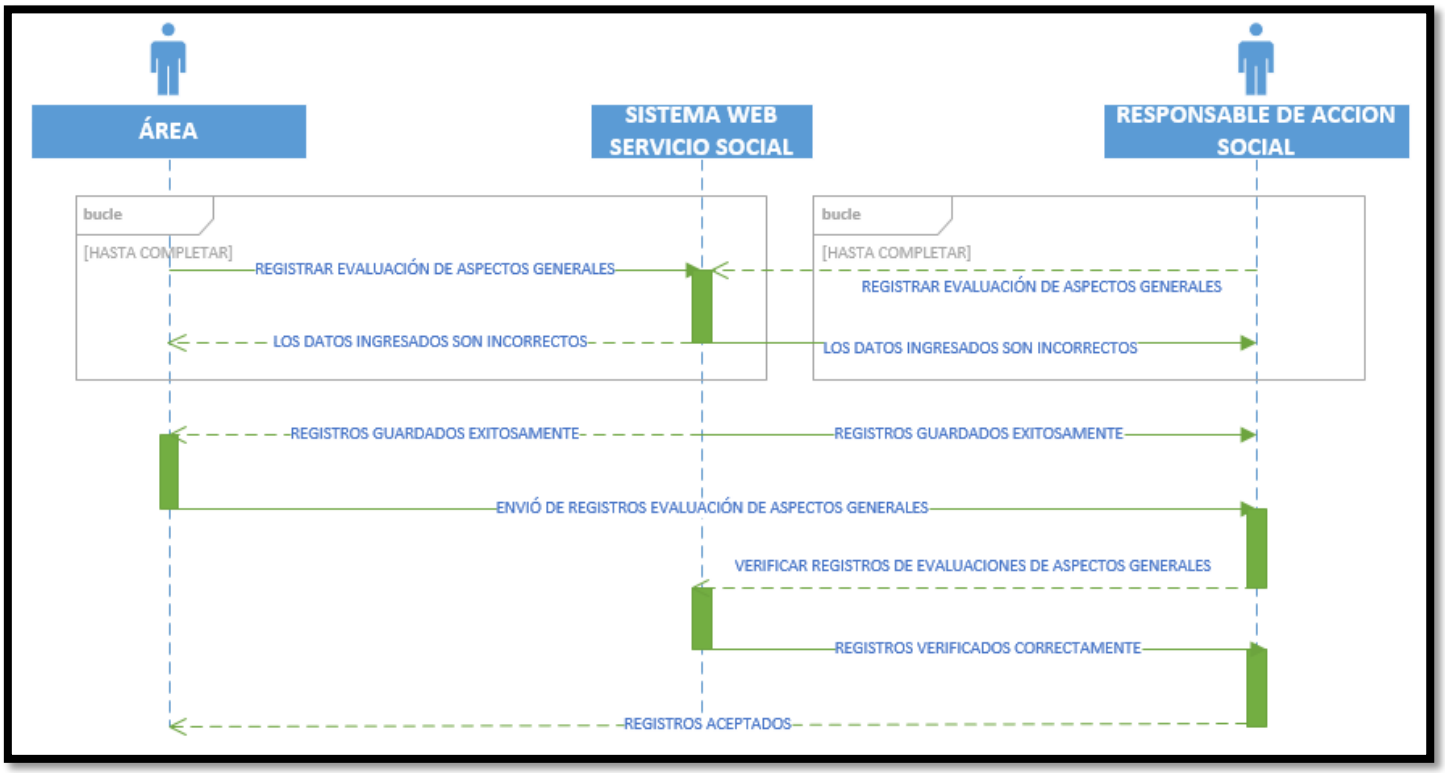



Figura Nº 18: Diagrama de secuencia de proceso para gestionar evaluaciones de aspectos generales del Becario

### 6.3.2. Diseño de Interfaz Gráfica

En esta sección se exponen los criterios para el diseño de la interfaz gráfica para la implementación de la Capa de Presentación. Posteriormente se describen las restricciones asumidas en el diseño gráfico Web.

#### 6.3.2.1. Estándar de Interfaz Gráfica

Todas las páginas del sistema (con excepción de la interfaz de inicio de sesión) seguirán el patrón gráfico mostrado en la figura N° 19.

**Boceto Principal:** El menú está ubicado al lado izquierdo de la aplicación, su fondo es de color negro degradado a un tono gris oscuro. Los accesos del menú están agrupados según el tipo de información y datos que procesa, estos grupos son denominados elementos padres, contienen letras y un glyphicon a la izquierda que serán de color gris en estado normal. Cuando se enfocan en ella el color será color blanco. Los elementos hijos tendrán las mismas propiedades de los elementos padres con la excepción que no tendrán glyphicon.

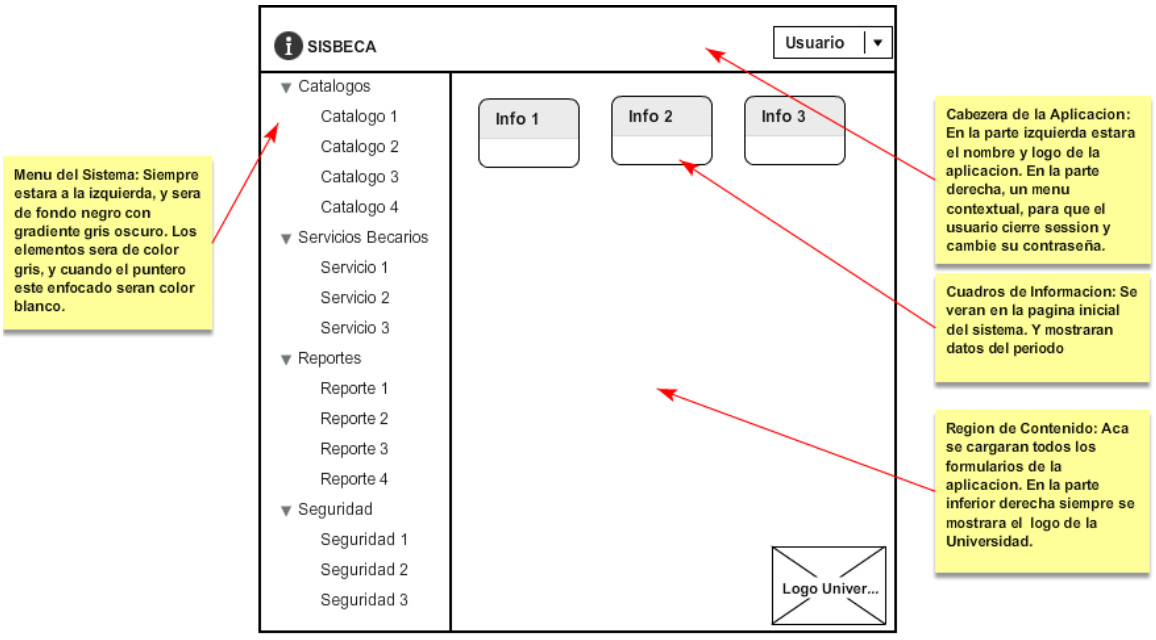



Figura Nº 19: Boceto Pantalla Principal

**Bocetos Formulario de Registros:** Está compuesto por una grilla los cuales contienen datos de cada entidad o tabla de base de datos, se muestran al dar clic al elemento hijo del menú de la aplicación. Esto solo para los casos de los grupos Catálogos, Servicios de Becarios y Seguridad.

En el formulario también se muestra un botón azul con un glyphicon del signo más (+), y se utiliza para cargar un formulario donde se agrega un nuevo registro.

El botón está ubicado arriba de la grilla en el lado izquierdo. Arriba del botón se presenta un texto el cual indica el elemento de base de datos que se está presentando.

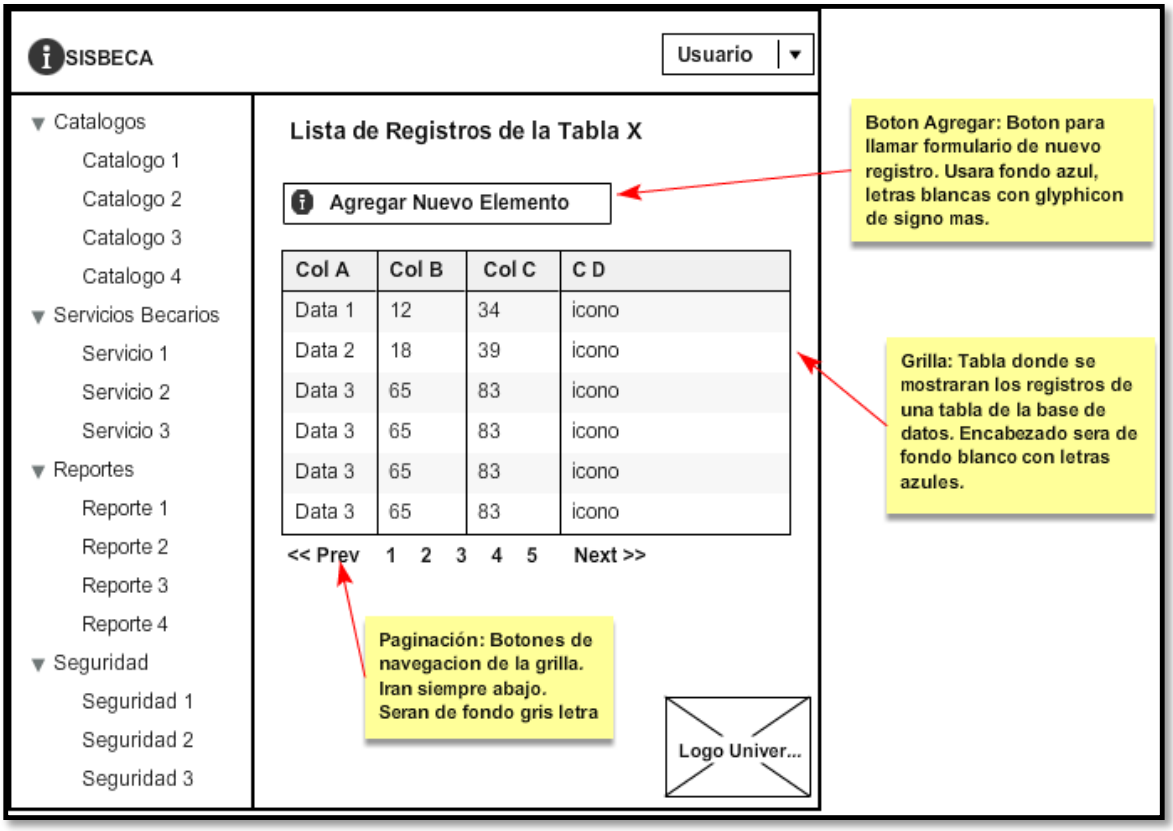



Figura Nº 20: Bocetos Formularios de Registros

**Grilla:** Los registros se mostraran en una grilla (tabla), está contiene un encabezado con columnas de letras color azul y fondo blanco, a la derecha de la columna tiene un glyphicon la cual se utiliza para realizar filtros de los datos. Las filas de los registros tienen fondo de color intercalado blanco y gris. En el pie de la grilla se muestra los botones grises, esta representa la paginación de la grilla.

**Edición de Registro:** la edición de registro se invoca desde la grilla donde se muestran los registros de cada tabla de la base de datos. Este carga un formulario similar al formulario que se utiliza para agregar un nuevo registro. El botón de edición es un glyphicon color azul que representa un lápiz con un cuadro de puntas redondeadas.

**Nuevo Registro:** Tiene la misma apariencia del formulario de edición, es un botón de color azul ubicado arriba de la grilla.

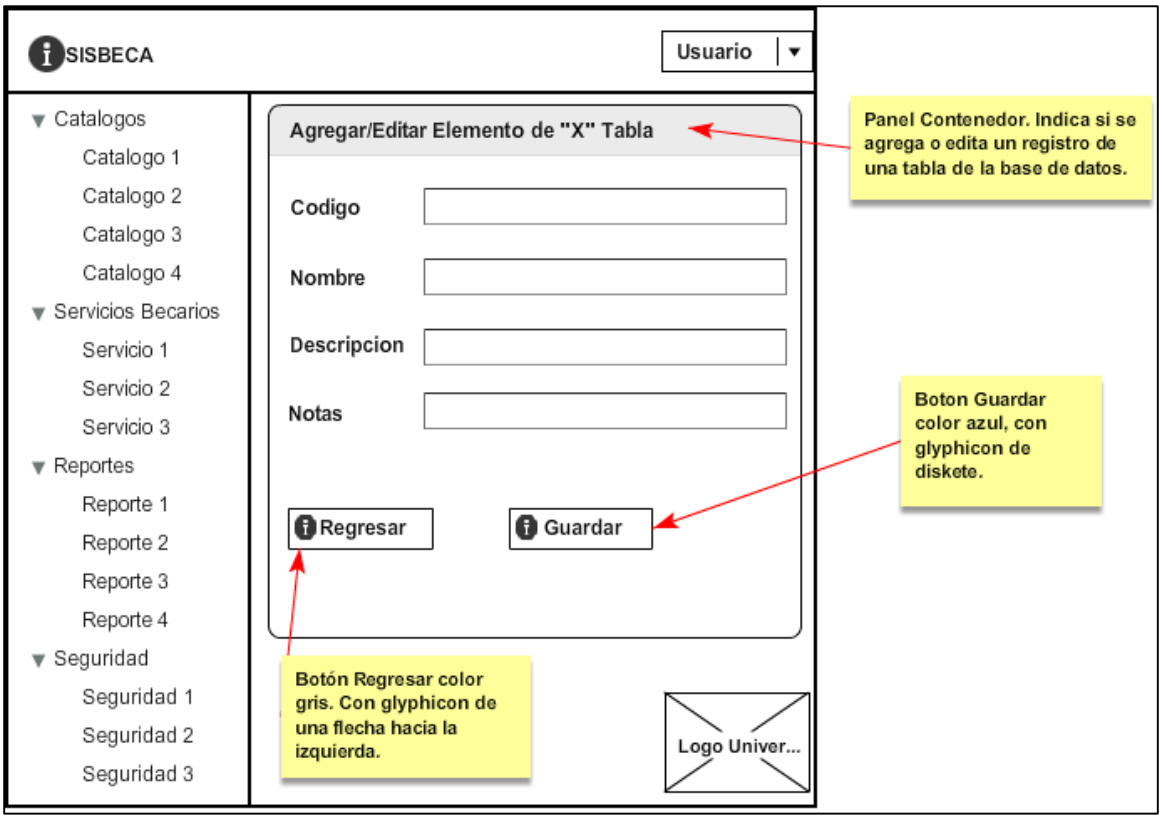



Figura Nº 21: Boceto Formulario Editar

**Botón Guardar:** es de color azul, y contiene un glyphicon de disquete. Está ubicado en la parte inferior, lado izquierdo del formulario a continuación de botón regresar.

**Botón Regresar:** se utiliza para ir a la página anterior, es de color gris y contiene un glyphicon de una flecha dirigida hacia la izquierda. Está ubicado en la parte inferior, lado izquierdo. Es el primer botón de izquierda a derecha.

**Boceto Impresión:** el formulario de impresión muestra un panel de color verde, el botón tiene como descripción "Generar" y es de color verde. Una vez llenado los parámetros que serán seleccionados desde una lista o llenados desde una caja de texto, se mostrará el reporte que se desea ver.

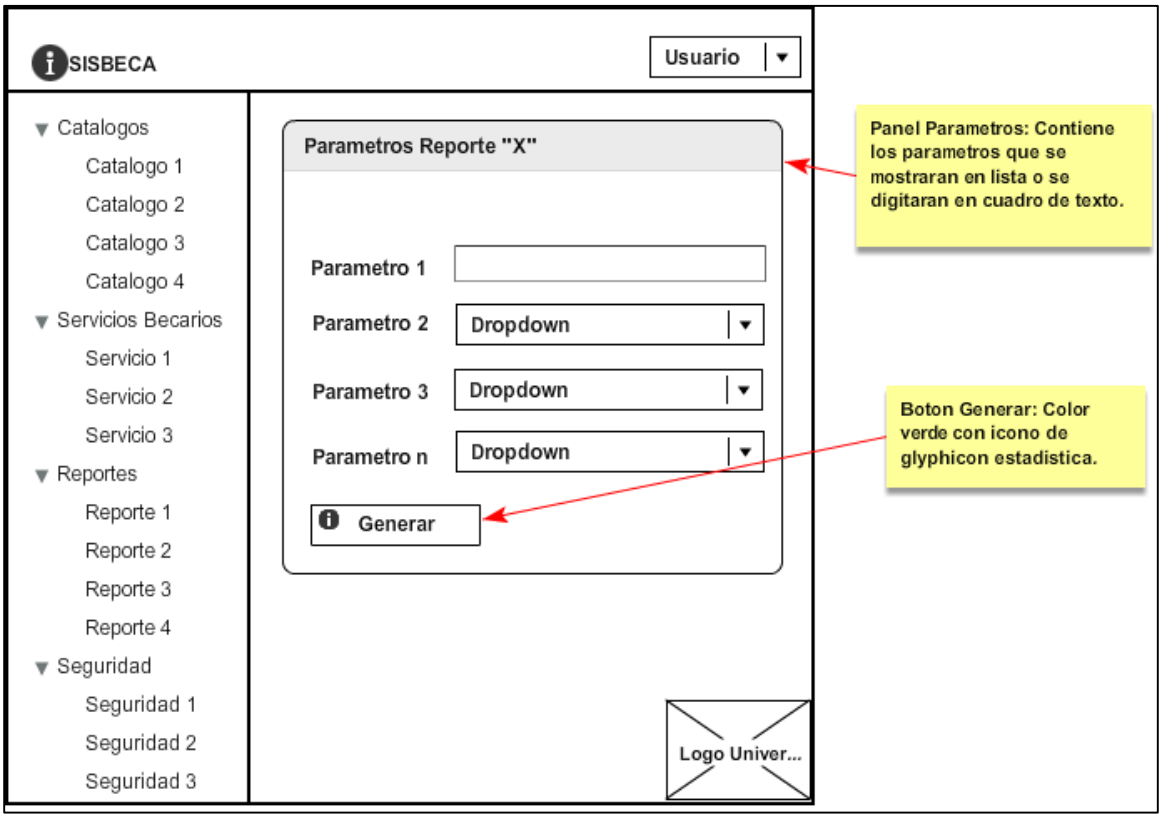



Figura Nº 22: Boceto Formulario de Reportes

#### 6.3.2.2. Consideraciones sobre la Interfaz Gráfica

Las observaciones señaladas a continuación favorecen la implementación de una interfaz sencilla, intuitiva y de fácil interacción para el usuario.

1. Las páginas no albergan elementos dinámicos como contenidos en Flash, archivos de imágenes GIF animados entre otros dado el alto consumo de recursos demandados en la aplicación; para escenarios con múltiples conexiones y transacciones la incorporación de estos componentes afectaría el rendimiento y tiempos de respuesta del servidor.
2. La implementación es trabajó con tablas HTML y páginas maestras para contribuir así con la estandarización del diseño y distribución uniforme de elementos gráficos en pantalla.
3. El tamaño de caracteres se limitó por línea de acuerdo a las dimensiones de la pantalla, evitando de esta forma el truncamiento automático de textos.
4. La interfaz gráfica ofrece opciones para minimizar la escritura a partir de controles como dropdownlists, radiobuttons, checkboxes, entre otros. Así como el establecimiento de valores predeterminados en los campos de las pantallas.
5. La aplicación web deberá funcionar igual, independientemente de la versión del sistema operativo y del navegador web que esté utilizando, siempre y cuando estos sean modernos.
6. Se usaron patrones que son populares para ayudar a que los usuarios se adecuen a la interfaz de la aplicación. Por ejemplo, colores (rojo para identificar los errores), iconos por pictografía (un disquete para indicar el símbolo de guardar) y la ubicación de los controles (el botón “guardar” siempre está en el lado derecho).
7. Mantiene consistencia en la interfaz, haciendo uso de convenciones para que los mismos tipos de elementos se muestren en el mismo lugar, los elementos tengan las mismas apariencias según las acciones que realizan, ejemplo: el botón de guardado debe ser siempre azul, el botón de impresión debe ser verde.

8. Con el fin de evitar ralentizar la carga de la aplicación web y reducir la cantidad de datos transmitidos, se usan pictografías (“glyphicon”) que son fuentes tipográficas que contienen iconos y se comportan como letras (fuentes); se pueden redimensionar y no pierden la resolución en ningún momento, se puede cambiar de color y añadirles efectos como si se tratase de una fuente de letra.

## 6.4. Implementación del Sistema de Información

El presente acápite tiene como propósito presentar las tecnologías seleccionadas para la implementación del software. Adicionalmente, se define la estrategia de pruebas y los tipos de pruebas seleccionados en esta etapa.

### 6.4.1. Construcción

En esta sección se hace un resumen de las características de las principales tecnologías, motores y frameworks empleados en la implementación como el lenguaje de programación, librerías, motor de base de datos entre otros.

El sistema se desarrolló en entorno de Visual studio 2015 Comunity, el lenguaje en el que fue programado es C SHARP por ser considerado uno de los más idóneos a la hora de crear sistemas web.

Las herramientas utilizadas fueron:

- .NET Framework 4.5: permite que el sistema pueda ser instalado en sistemas operativos Microsoft y alcanzar la máxima compatibilidad.
- Entity Framework: facilita la creación de métodos para grabar, modificar o eliminar datos a partir de la base de datos.
- ReportViewer: para crear informes que contengan la información detallada del usuario.
- MVC 4: ya que es una aplicación web utiliza un patrón de arquitectura como una composición de tres funciones: Modelo, Vista, Controlador.
- SQL Server 2008 R2.

El Sistema SISBECA (Sistema del Servicio Becario) fue desarrollado haciendo uso del modelo de programación MVC (Modelo, Vista, Controlador), utilizando el software de desarrollo de aplicaciones Visual Studio Community Edition 2015, también nos apoyamos de ASP.NET MVC como el framework de desarrollo aplicaciones web y el MSSQLSERVER como servicio.

El sistema faculta al usuario la capacidad de almacenar registros de los catálogos y servicio social de una manera ordenada y fácil de analizar, además,

dispone del módulo de administración de usuario y roles para garantizar la seguridad de los datos.

La Universidad Nacional de Ingeniería “UNI”, nos proporcionó un espacio en la Oficina de Acción Social para la puesta en marcha del Sistema de Información SISBECA, es importante mencionar que este Sistema no estará vinculado a ningún otro sistema, por los mecanismos de seguridad que la DITI ha establecido.

El Sistema de Información SISBECA correrá en un ambiente de prueba en la Oficina de Acción Social, lo cual es una buena práctica para los usuarios, para efectos de capacitación y evaluar el resultado de determinadas acciones del sistema en un ambiente seguro, el cual ayuda a no causar ningún daño o distorsión a datos reales.

### 6.4.2. Pruebas

En esta sección se detalla el procedimiento de pruebas durante la verificación y validación del software, desde los tipos de pruebas seleccionados junto con las justificaciones de sus respectivas elecciones, así como la estrategia desarrollada.

#### 6.4.2.1. Estrategia de Pruebas

El objetivo global de la estrategia de pruebas es demostrar el funcionamiento completo del software a nivel de eficiencia de código y funcionalidad. En otras palabras, verificar la interacción e integración de los componentes y validar la implementación de todos los requerimientos funcionales de producto especificados anteriormente.

Para el cumplimiento de lo descrito anteriormente, la estrategia establecida será realizar casos de pruebas.

### 6.4.2.2. Tipos de Pruebas

En esta sección se describen los tipos de prueba empleados en la estrategia de pruebas.

#### 6.4.2.2.1. Registrar Catálogo Recinto

| Prueba 1: Registro de Recinto | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra catálogos del Sistema, Recinto. |
| Precondición | Iniciar sesión en el sistema. |
| Descripción de la prueba | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Recinto<br>3. Dar Click en agregar Recinto<br>4. LLenar el formulario correspondiente a los datos del Recinto<br>I. Nombre del Recinto: Recinto Univeristario Simón Bolivar.<br>II. Abreviatura: RUSB.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de recintos, inclusive el recinto agregado. |

Tabla 12: Registro Catálogo Recinto

#### 6.4.2.2.2. Registrar Catálogo Facultad

| Prueba 2: Registro de Facultad | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra catálogos del Sistema, Facultad. |
| Precondición | Iniciar sesión en el sistema.<br>Exista registro en catalogo de recintos. |
| Descripción de la prueba | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Facultad<br>3. Dar Click en agregar Facultad<br>4. LLenar el formulario correspondiente a los datos de la facultad<br>I. Selecciona Recinto: Recinto Univeristario Simón Bolivar.<br>II. Nombre: Facultad de Computación y Electrónica.<br>III. Abreviatura: FEC.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de facultades, inclusive la facultad agregada. |

Tabla 13: Registro Catálogo Facultad

#### 6.4.2.2.3. Registrar Catálogo Carrera

| Prueba 3: Registro de Carreras | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra catálogos del Sistema, Carreras. |
| Precondición | Iniciar sesión en el sistema.<br>Existan registros en catálogos de facultades. |
| Descripción de la prueba | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Carreras<br>3. Dar Click en agregar Carrera<br>4. LLenar el formulario correspondiente a los datos de la carrera:<br>I. Selecciona Facultad: Facultad de Computación y Electrónica.<br>II. Nombre: Ingeniería en Computación.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de carreras, inclusive la carrera agregada. |

Tabla 14: Registro Catálogo Carrera

#### 6.4.2.2.4. Registrar Catálogo Beneficiario

| Prueba 4: Registro de Beneficiario | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra catálogos del Sistema, Beneficiarios. |
| Precondición | Iniciar sesión en el sistema. |
| Descripción de la prueba | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Beneficiario<br>3. Dar Click en agregar Beneficiario<br>4. LLenar el formulario correspondiente a los datos del beneficiario:<br>I. Nombre: Biblioteca RUSB.<br>II. Representante: Carlos Flores Rojas.<br>III. Teléfono: 2250-0848.<br>IV. Celular: 89981317.<br>V. Dirección: RUSB.<br>VI. Correo: biblioteca@uni.edu.ni.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de beneficiarios, inclusive el beneficiario agregado. |

Tabla 15: Registro Catálogo Beneficiario

#### 6.4.2.2.5. Registrar Catálogo Tipos de Becas

| Prueba 5: Registro de Tipos Becas | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra catálogos del Sistema, Tipos Becas. |
| Precondición | Iniciar sesión en el sistema. |
| Descripción de la prueba | 1. Selecciona menu Catálogos.<br>2. Dar Click en el submenu Tipos Becas.<br>3. Dar Click en agregar Tipos Becas.<br>4. LLenar el formulario correspondiente a los datos del tipo de beca:<br>I. Nombre: Beca Residencia.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de tipos de becas, inclusive el registro de tipo de beca agregado. |

Tabla 16: Registrar Catálogo Tipos de Becas

#### 6.4.2.2.6. Registrar Catálogo Evaluación

| Prueba 6: Registro de Evaluación | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra catálogos del Sistema, Evaluación. |
| Precondición | Iniciar sesión en el sistema. |
| Descripción de la prueba | 1. Selecciona menu Catálogos.<br>2. Dar Click en el submenu Evaluaciones.<br>3. Dar Click en agregar Evaluaciones.<br>4. LLenar el formulario correspondiente a los datos del registro de evaluación:<br>I. Nombre: Excelente.<br>II. Valor:5<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado del catálogo de evaluación, inclusive el registro de evaluación agregado. |

Tabla 17: Registro Catálogo Evaluación

#### 6.4.2.2.7. Registrar Catálogo Periodo Académico

| Prueba 7: Registro de Periodos Académicos | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra catálogos del Sistema, Periodos Académicos. |
| Precondición | Iniciar sesión en el sistema. |
| Descripción de la prueba | 1. Selecciona menu Catálogos.<br>2. Dar Click en el submenu Periodos Académicos.<br>3. Dar Click en agregar Periodos Académicos.<br>4. LLenar el formulario correspondiente a los datos del registro de evaluación:<br>I. Ciclo: I Semestre.<br>II. Año: 2015<br>III. Estado: Activo.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado del catálogo Periodos Académicos, inclusive el registro de periodo académico agregado. |

Tabla 18: Registro Catálogo Periodo Académico

#### 6.4.2.2.8. Registro Catálogo Alumnos

| Prueba 8: Registro de Alumnos | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra alumno al Sistema. |
| Precondición | Iniciar sesión en el sistema. |
| Descripción de la prueba | 1. Selecciona menu Servicios de Becario.<br>2. Dar Click en el submenu Alumnos.<br>3. Dar Click en agregar Alumnos.<br>4. LLenar el formulario correspondiente a los datos del alumno:<br>I. Carnet: 2010-10011.<br>II. Nombres: Jamie Lissette.<br>III. Apellidos: Membreño Duarte.<br>IV. Sexo: F.<br>V. Correo: vacio.<br>VI. Direccion: Tipitapa, Bo. Yuri Ordoñez, calle Puente el amor, 1½c. arriba.<br>VII. Red Social: vacio.<br>VIII. Convencional: vacio.<br>IX. Movistar: vacio.<br>X. Claro: vacio.<br>XI. Presidente: Falso.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado del catálogo Alumnos, inclusive el registro del nuevo alumno agregado. |

Tabla 19: Registro Catálogo Alumnos

#### 6.4.2.2.9. Registro de Alumnos por Periodo

| Prueba 9: Registro de Alumnos por Periodo | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social asocia la lista de alumno becado a un periodo académico, carrera y tipo de beca. |
| Precondición | Iniciar sesión en el sistema.<br>Listado de Carreras.<br>Listado de Tipos de Becas.<br>Listado de Periodos Académicos. |
| Descripción de la prueba | 1. Selecciona menu Servicios de Becario.<br>2. Dar Click en el submenu Alumnos x Periodo.<br>3. Dar Click en Agregar Asociación Alumno Periodo.<br>4. LLenar el formulario correspondiente a los datos del alumno:<br>I. Selección Periodo Académico: 2015.<br>II. Selección Carrera: Ingenieria en Computación.<br>III. Selección Tipo Beca: Beca Residencia.<br>IV. Buscar alumno por nombre: Jamie Lissette.<br>V. Seleccionar alumno: Jamie Lissette Membreño Duarte.<br>VI. Buscar alumno por carnet: 2010-40016.<br>VII. Seleccionar alumno: Pedro Roberto Morales Ruiz.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado asociación de alumnos por periodo, incluidos los alumnos recién agregados. |

Tabla 20: Registro de Alumnos por Periodo

#### 6.4.2.2.10. Registrar Tareas

| Prueba 10: Registro de Tareas | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra tareas o actividades que los beneficiarios necesitan. |
| Precondición | Iniciar sesión en el sistema.<br>Listado Tipo de Tarea.<br>Listado de Beneficiarios. |
| Descripción de la prueba | 1. Selecciona menu Servicios de Becario.<br>2. Dar Click en el submenu Tareas.<br>3. Dar Click en Agregar Tareas.<br>4. LLenar el formulario correspondiente a los datos de la tarea:<br>I. Selección Tipo Tarea: Interna.<br>II. Selección Beneficiario: Biblioteca RUSB.<br>III. Selección Tipo Beca: Beca Residencia.<br>IV. Nombre: Asistente de Biblioteca.<br>V. Lugar: Segundo Piso Biblioteca.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de tareas, incluso la tarea agregada. |

Tabla 21: Registro de Tareas

#### 6.4.2.2.11. Registrar Asignación de Tarea

| Prueba 11: Registro Asignación de Tarea | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social asigna tarea a un alumno becado. |
| Precondición | Iniciar sesión en el sistema.<br>Listado de Tareas.<br>Listado de Alumno asociado a un periodo académico. |
| Descripción de la prueba | 1. Selecciona menu Servicios de Becario.<br>2. Dar Click en el submenu Asignación de Tareas.<br>3. Dar Click en Agregar Asignación de Tareas.<br>4. LLenar el formulario correspondiente a los datos de la asignación de tarea:<br>I. Selección Tarea: Asistente de Biblioteca.<br>II. Selección Alumno: Jamie Lissette Membreño Duarte.<br>III. Fecha Inicio: 2015-02-04.<br>IV. Fecha Fin: 2015-02-04.<br>V. Horas: 4.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de asignaciones de tareas, incluso la asignación agregada. |

Tabla 22: Registro de Asignación de Tarea

#### 6.4.2.2.12. Registrar Asistencia

| Prueba 12: Registro de Asistencia | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra asistencia de alumno para realizar tarea. |
| Precondición | Iniciar sesión en el sistema.<br>Alumno tenga asignado tarea.<br>Listado de Alumno asociado a un periodo académico. |
| Descripción de la prueba | 1. Selecciona menu Servicios de Becario.<br>2. Dar Click en el submenu Asistencia.<br>3. Dar Click en Agregar Asistencia.<br>4. LLenar el formulario correspondiente a los datos de una asistencia:<br>I. Selección Alumno con tarea: Jamie Lissette Membreño.<br>II. Fecha: 2015-02-24.<br>III. Hora Inicio: 2015-02-23 08:00:00.000.<br>IV. Hora Fin: 2015-02-23 10:00:00.000.<br>V. Observacion: vacio.<br>5. Dar Click en Guardar |
| Resultados Esperados | El sistema muestra listado de asistencia, incluso la asistencia agregada. |

Tabla 23: Registro de Asistencia

#### 6.4.2.2.13. Registrar Evaluación del Estudiante

| Prueba 13: Registro para Evaluar Estudiante | |
|---|---|
| Objetivo Prueba | El Responsable de Acción Social registra evaluación de tarea realizada por alumno becado. |
| Precondición | Iniciar sesión en el sistema.<br>Alumno tenga asignado tarea.<br>Lista de Evaluación.<br>Listado de Alumno asociado a un periodo académico. |
| Descripción de la prueba | 1. Selecciona menu Servicios de Becario.<br>2. Dar Click en el submenu Evaluar Estudiante.<br>3. Dar Click en Agregar Evaluación de Estudiante.<br>4. LLenar 2 formularios correspondiente a los datos de evaluacion a la tarea realizada y evaluación a los aspectos del becario:<br>I. Selección Alumno con tarea: Jamie Lissette Membreño.<br>II. Selección Evaluacion: Excelente.<br>III. Observación: beneficiario emitio documento de recomendación.<br>5. Dar Click en Guardar y Evaluar Aspectos.<br>6. Seleccionar evaluacion para los aspectos de Desempeño Laboral, Factor Humano y Habilidaddes.<br>7. Dar click en Guardar Evaluación Aspectos. |
| Resultados Esperados | El sistema muestra listado de evaluaciones a becarios, incluso la evaluación agregada. |

Tabla 24: Registro de Evaluación Estudiante

# 7. Conclusiones

La combinación de las metodologías de investigación orientada a objetos, la ingeniería de software, las entrevistas, las consultas, las observaciones y documentación proporcionada por Responsable de la Oficina de Acción Social, permitieron determinar las necesidades y requerimientos funcionales básicos de operación del proceso para realizar la documentación del análisis y diseño del Sistema de Información, aplicando la notación UML, específicamente diagramas y modelos de Casos de Uso.

Adicionalmente se redefinieron algunos procesos para contar con la información necesaria de las actividades realizadas por los becarios, esto mejora su seguimiento y valoración para agilizar el proceso de asignación o aprobación de becas para el siguiente periodo.

El sistema fue instalado en un equipo de la Oficina de Acción Social, donde se procedió a capacitar al usuario, posteriormente nos apoyó en la realización de las pruebas finales y después de realizar los ajustes necesarios se validó que la aplicación web cumple con los requerimientos iniciales del usuario.

La realización de este trabajo monográfico, permitió desarrollar el Sistema de Información SISBECA para la gestión de información del Programa del Servicio Social Becario para la Universidad Nacional de Ingeniería “UNI”.

Con el diseño de este nuevo sistema se logró mejorar el tiempo de procesamiento de información y de respuestas, ya que la información ahora está centralizada, facilitando la búsqueda de información, impresión de reportes a las autoridades de la UNI y control de los becarios en sus asignaciones.

# 8. Recomendaciones

Se recomienda vincular el Sistema “SISBECA” con los Sistemas existentes en la Universidad Nacional de Ingeniería “UNI”, como son el SIRA y el Sistema de Becas para aprovechar la información que manejan sobre los estudiantes y los becarios, esto evitará que los usuarios del Sistema tengan que ingresar toda la información de los estudiantes becados de nuevo ingreso.

# 9. Bibliografía


1. BERNAL, CESAR. 2008. Metodología de la investigación social. Editorial Prentice Hall.

2. HERNÁNDEZ SAMPIERI, Roberto, FERNÁNDEZ COLLADO, Carlos, BAPTISTA LUCIO, Pilar. 2010. Metodología de la investigación. 5ta Edición. México, D.F. McGraw-Hill/Interamericana.

3. Castañeda, De la Torre, Morán, Lara; Metodología de la investigación, Editorial Mc Graw Hill.

4. Piura Julio. 2008. Metodología de la investigación. Segunda edición. Managua, Nicaragua.

5. Sharon Chinchilla. (Julio 2012). http://estatico.uned.ac.cr/bienestar/vida/becas/tipo.shtml Recuperado el 08 de octubre del 2014, de http://estatico.uned.ac.cr/bienestar/vida/becas/documentos.shtml

6. Universidad Nacional de Ingeniería (UNI). http://www.dbe.uni.edu.ni/programa/ass/serviciosdelbecario.html Recuperado el 01 de Octubre del 2014, de http://www.uni.edu.ni/Vida_Estudiantil

7. Universidad Nacional de Ingeniería (UNI). Reglamento de la Universidad Nacional de Ingeniería. Recuperado el 01 de Octubre del 2014, de la Oficina de Acción Social.

8. MSc. Belkis Iglesias Asencio. 2013. TIC y la gestión comercial, Documento PDF, Recuperado el 10 de Septiembre del 2014, UAM, DPEC, Maestría en Recursos Humanos.

9. MSc. Belkis Iglesias Asencio. 2013. Herramientas de Informática para la Gestión Empresarial, Documento PDF, Recuperado el 10 de Septiembre del 2014, UAM, DPEC, Maestría en Recursos Humanos.

10. http://www.desarrolloweb.com/articulos/que-es-mvc.html, 02 de enero de 2014, Recuperado el 01 de septiembre del 2014, http://www.desarrolloweb.com.

11. Roger S. Pressman. 2002. Ingeniería del software, Un enfoque práctico"; McGraw-Hill

12. Jorge Cortés Álvarez. 2012, Universidad de Cartagena, Metodologías de desarrollo del Software. http://es.slideshare.net/cortesalvarez/metodologa-rup, Recuperado el 17 de septiembre del 2014.

13. http://www.uni.edu.ni/Alma_Mater/Quienes_somos

# 10. Anexos

## A. Manual de Usuario

### A.1. Introducción

Actualmente el uso de sistemas de información ha dejado de ser un lujo para convertirse en una realidad, tanto para las empresas, como para las instituciones. El uso de estas aplicaciones ha logrado optimizar el trabajo de los funcionarios, facilitando sus labores y mejorando su rendimiento, ya que permiten mantener el control, disponibilidad e integridad de la información.

**SISBECA** es un sistema formado por un conjunto de normas y procedimientos utilizados en las distintas etapas del proceso de control y registro de la asistencia social que realizan los becarios en su tiempo de servicio. Constituyendo de esta manera un ciclo integrado e interactivo de acciones articuladas a los objetivos de desarrollo de la oficina de acción social de la UNI, donde la compresión y aplicación de cada una de ellas es gravitante en la calidad de la gestión y el uso efectivo y racional de los recursos de la universidad.

Con el fin de facilitar el uso de este programa se redactó el presente manual de usuario que indica los pasos a seguir para el manejo correcto del sistema, sirviendo de apoyo al usuario en la manipulación de los datos.

## A.2. Acceso al Sistema

Para tener acceso al Sistema del Programa del Servicio Becario (SISBECA), se puede utilizar cualquiera de los 3 navegadores web más utilizados (Google Chrome, Internet Explorer y/o Mozilla Firefox). Aunque existen otros navegadores, las pruebas del sistema se realizaron con los navegadores antes mencionados. La dirección para acceder a la aplicación es http://SISBECA.


Figura Nº 23: Navegadores WEB

## A.3. Validación de Usuario

Para el ingreso al sistema el usuario debe introducir el nombre de usuario y la contraseña correspondiente en la ventana de inicio de sesión, y luego hacer clic en el botón Iniciar Sesión o bien presionar la tecla , como se puede apreciar en la **Figura N° 24.**

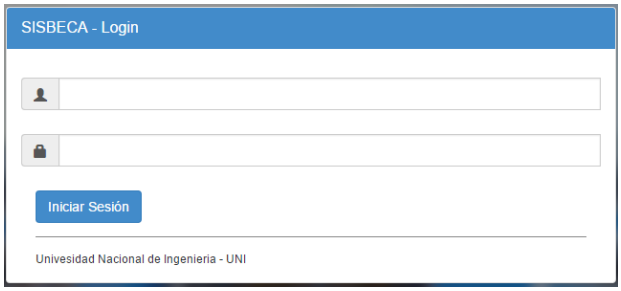



Figura Nº 24: Acceso al Sistema

Si el usuario y contraseña son válidos, podrá acceder a la pantalla principal del sistema. En caso que el usuario o la contraseña no sean correctos, el sistema presentará el siguiente mensaje en la ventana del login “El nombre de usuario o la contraseña especificados son incorrectos. Si la contraseña no se digita se mostrará el siguiente mensaje: “El campo Contraseña es obligatorio”. Si a pesar de digitar el usuario y contraseña correctamente, el sistema no permite el acceso, contáctese con el administrador funcional del sistema, es posible que unos de los datos que se les ha proporcionado no sea el correcto.

## A.4. Pantalla Principal del Sistema


Figura Nº 25: Pantalla Principal del Sistema

La pantalla principal del sistema **(Ver Figura N° 25)**, se compone de los siguientes elementos:

- **Figura N° 26, Parte 1: Barra de Menú o Menú Principal:** Se encuentra en la parte izquierda de la pantalla y contiene de forma agrupada (Catálogos del sistema, Servicio de Becario, Reportes y Seguridad) todas las opciones disponibles para que el usuario interactué con el sistema.

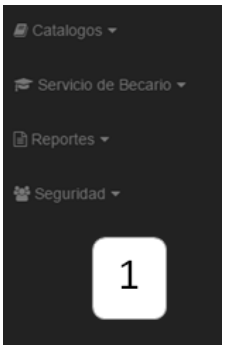



Figura Nº 26: Menú Principal

- **Figura N° 27, Parte 2: Menú Contextual del Usuario:** Está ubicado en la parte superior derecha de la pantalla, en ella se podrá notar un saludo al usuario. Al presionar clic en este enlace se mostrará un menú contextual, con enlace al formulario de cambio de contraseña, y al cierre de sesión.


Figura Nº 27: Menú Contextual del Usuario

- **Figura N° 28, Parte 3: Información para el Administrador:** El recuadro de fondo blanco, en él se presenta información de relevancia para el administrador funcional del sistema, tales como, el periodo activo, cantidades de alumnos sin tareas asignadas y tareas que no tienen alumnos asignados.

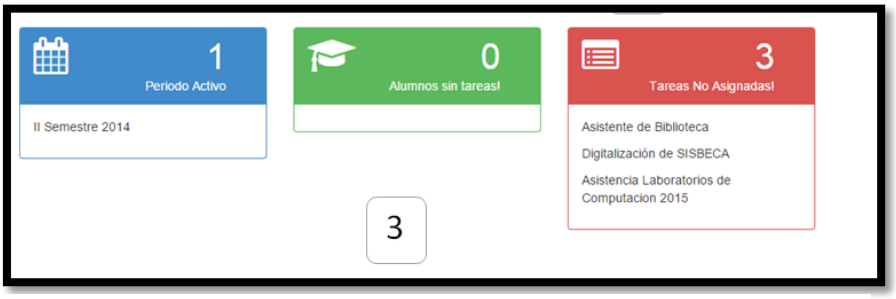



Figura Nº 28: Información para el Administrador

## A.5. Accesos del Menú Principal

El menú principal esta agrupado y ordenado según el tipo de formulario al que pertenece el sistema: formulario de catálogo, formulario de procesos de Servicio de Becario, formulario para reportería y formularios para la administración de la seguridad.

Tomando en consideración que la pantalla de catálogos, es similar en presentación y funcionalidad, se mostrara el uso de un único catálogo del sistema:

## A.6. Catálogos del Sistema

Los catálogos del Sistema son los primeros formularios que se deberán utilizar, es decir, que desde aquí se ingresaran los primeros registros, porque representan la información base con la que funciona el sistema, tales como la información de recintos, facultades, entre otros.

A continuación se presenta el orden en que deberá registrarse la información de los catálogos:

1. Recintos
2. Facultades
3. Carreras
4. Beneficiario
5. Tipos Becas
6. Tipos Tareas
7. Evaluaciones
8. Periodos Académicos
9. Alumnos

El sistema indicará al usuario cuáles serán los campos obligatorios y cuales datos no son válidos a través de mensajes ilustrativos.

## A.7. Catálogo Recinto

Al presionar clic sobre el Menú-> Catálogos-> Recintos, se mostrará la lista de recintos registrados en el sistema (Ver Figura N° 29). Desde este formulario se podrá seleccionar el recinto y presionar clic en el botón editar que está ubicado en la columna “Acción” para modificar la información del recinto (Ver Figura N° 30) o presionar clic sobre el botón azul “Agregar Recinto” para registrar un nuevo Recinto (Ver Figura N° 31).


Figura Nº 29: Listado de Recinto

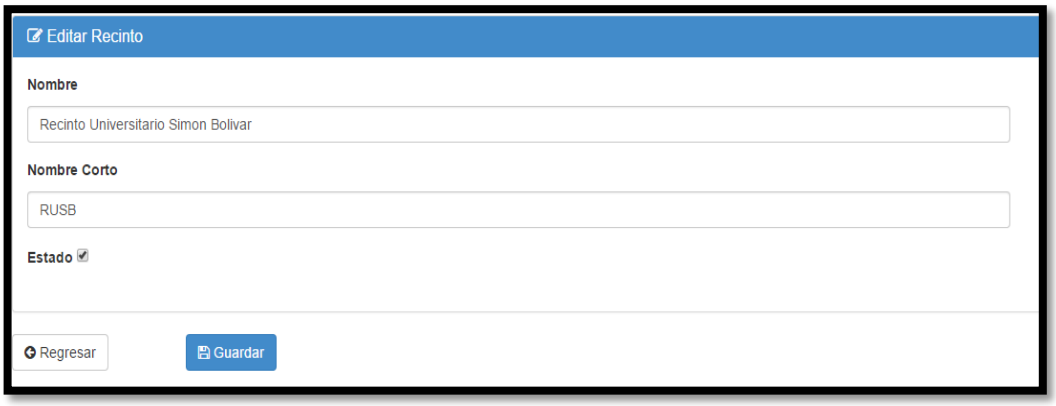



Figura Nº 30: Editar Recinto

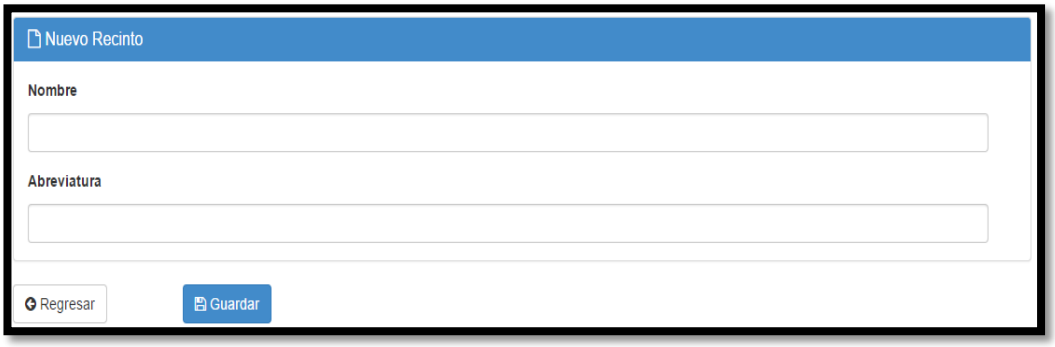



Figura Nº 31: Agregar Recinto

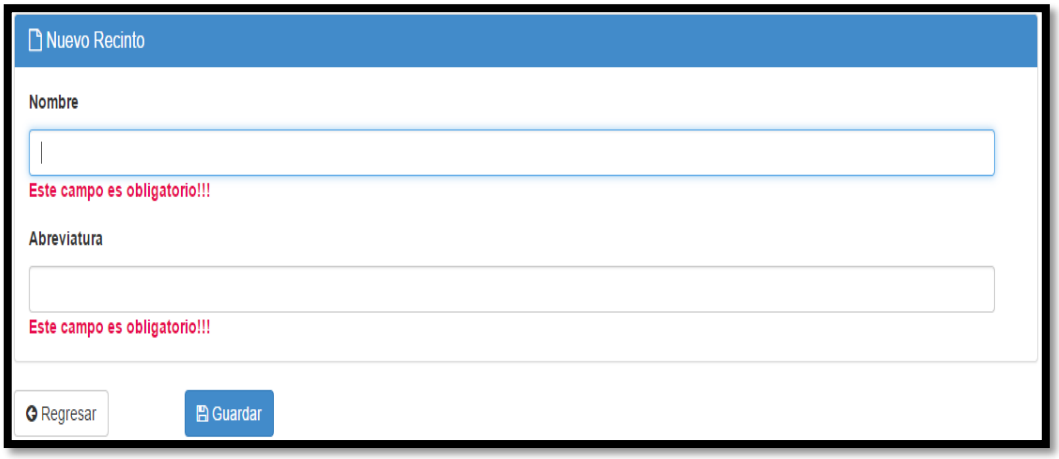



Figura Nº 32: Mensajes de Validación del Sistema

## A.8. Catálogo Periodo Académico

Al presionar clic sobre el Menú-> Catálogos-> Periodos Académicos, se mostrará la lista de Periodos Académicos registrados en el sistema (Ver Figura N° 33). Desde este formulario se podrá seleccionar el periodo y presionar clic sobre el botón editar que está ubicado en la columna "Acción" para modificar la información del periodo académico (Ver Figura N° 435) o presionar clic sobre el botón azul "Agregar Periodo Académico" para registrar un nuevo Recinto (Ver Figura N° 34).

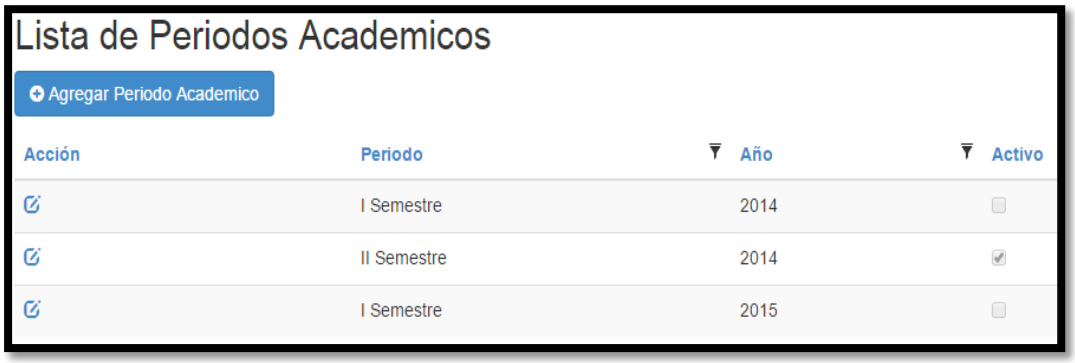



Figura Nº 33: Lista de Periodos Académicos

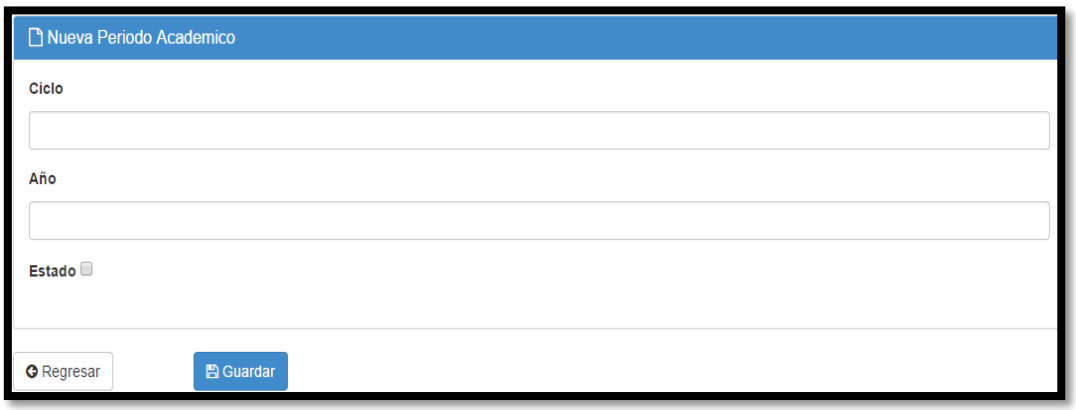



Figura Nº 34: Agregar Periodo Académico

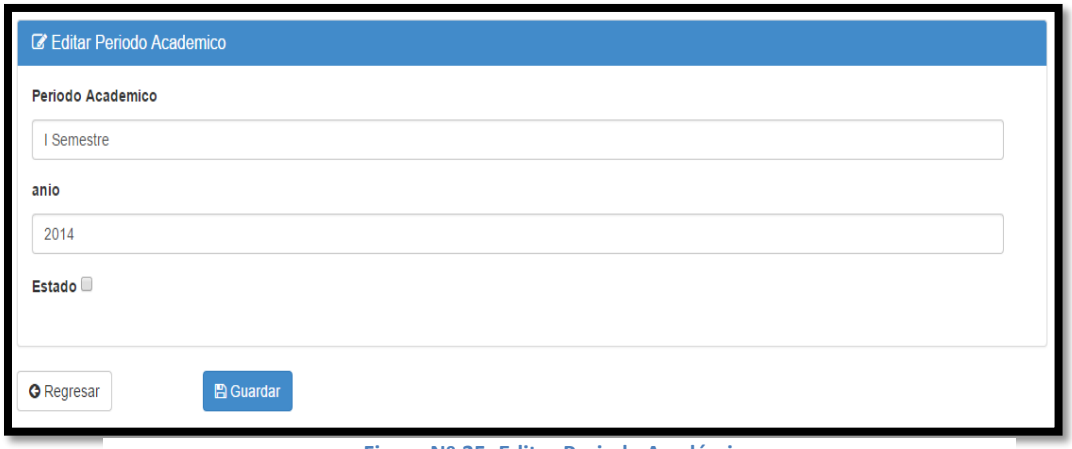



Figura Nº 35: Editar Periodo Académico

Cuando se modifica el estado del periodo académico de inactivo (descheckeado) a activo (checkeado) el sistema actualiza el campo estado del restos de los periodos académicos, para pasarlo a inactivos. De esta forma se asegura que se cumpla la norma de que en el sistema solo un periodo académico puede estar activo.

En el sistema, todos los procesos que involucren el control de la información de servicio social del becario, toman en consideración únicamente al periodo activo. Es decir que cuando se registre una asignación de tarea, registro de asistencia y evaluaciones, la información que se mostrará será la concerniente al periodo que este activo en el sistema.

### A.9. Servicio Becario

Desde este grupo del menú principal (Ver Figura N° 36), se administra la información de los procesos de servicios becarios, las actividades a realizar, evaluación a su desempeño, y la vinculación de los alumnos con el periodo, carrera y tipo de beca que posee el becario.

Como propuesta se ha ubicado en este grupo del Menú, el catálogo de Alumnos. Esto porque en la concepción del sistema se planeaba tomar a los estudiantes del actual sistema con el que la universidad cuenta.

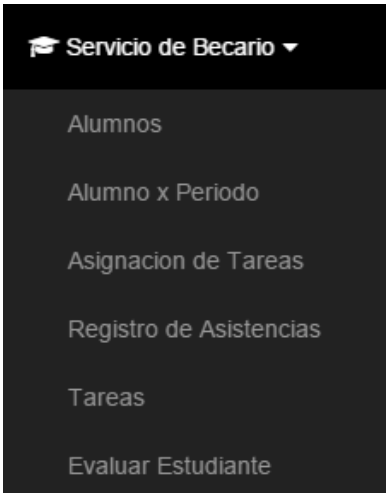



Figura Nº 36: Menú Servicio Becario

Pero en la actualidad en SISBECA, se cuenta con un formulario para alimentar el catálogo de alumno, donde solo se ingresan sus datos personales, sin incluir información académica.

## A.10. Mantenimiento de Alumnos

Para acceder al mantenimiento de alumnos ir al Menú Servicio Becario -> Alumno, el sistema muestra la lista de los alumnos registrados en el sistema. Desde este formulario (Ver Figura N° 37) se podrá editar o agregar un nuevo alumno.

En la lista (Ver Figura N° 37) se muestran alumnos por grupos de 10 registros, para navegar los próximos diez registros, o más adelante, se deben utilizar los botones ubicados en la parte inferior izquierda de la lista. También podrá ordenar de forma ascendente y descendente presionando clic sobre cualquiera de los encabezados de las columnas. Por último a la derecha de los encabezados de las columnas se podrá observar un icono en forma de embudo, este botón sirve para realizar filtro de búsqueda, los filtros permitidos son “igual”, “contienen”, “comienza por” y “termina por”. En la lista de alumnos, los filtros permiten hacer una búsqueda por nombres, apellidos, email y el estado activo. Para quitar el filtro, se debe repetir el paso y limpiar o dejar vacío las casillas de texto.

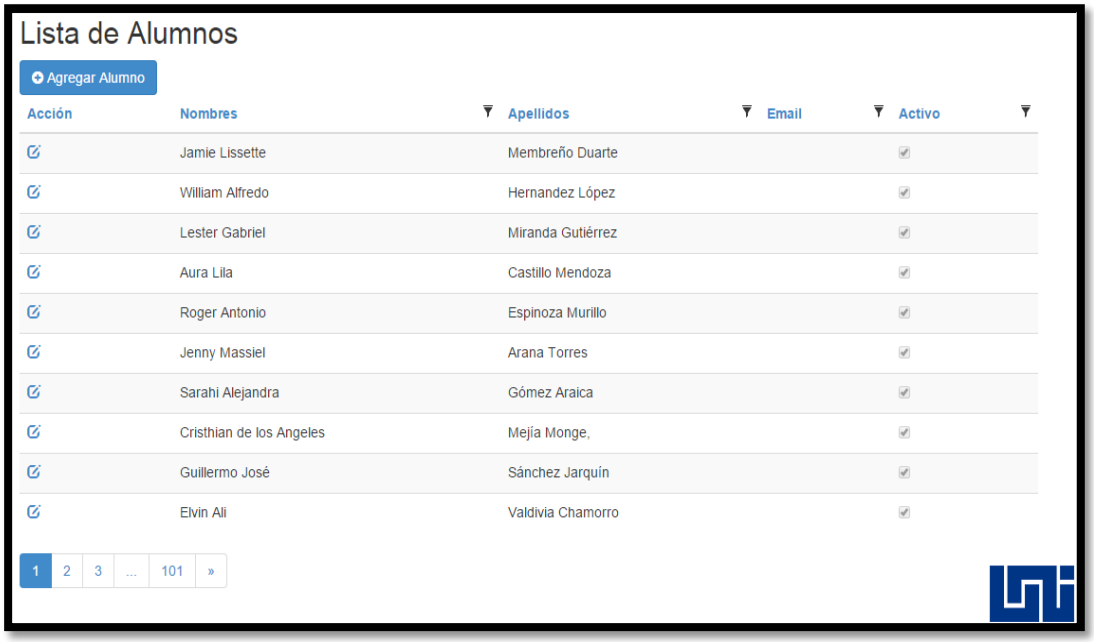



Figura Nº 37: Lista de Alumnos

### A.10.1. Agregar un Nuevo Alumno

1. Presione clic en el botón “Agregar Alumno”. El botón es azul y está ubicado en la parte superior izquierda del formulario (Ver Figura N° 37).
2. Se deben llenar los campos requeridos del alumno. Hay que tener en cuenta que si ya existe un alumno con número de carnet en el sistema, el sistema no permitirá guardar el nuevo alumno. En la Figura N° 38 se muestran los campos requeridos.
3. Una vez que los campos requeridos estén llenos, se debe presionar clic en el botón azul “Guardar”.

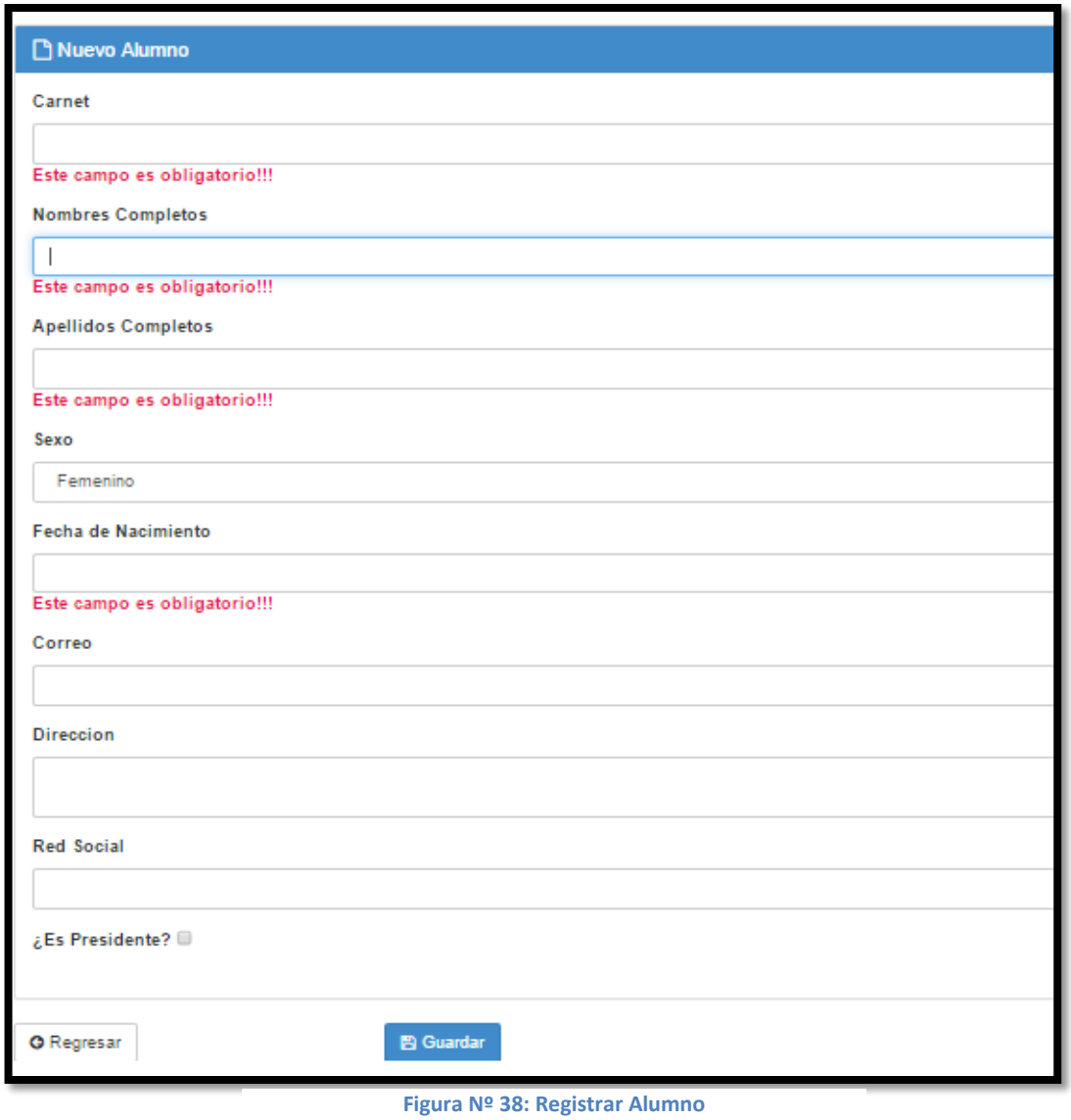



Figura Nº 38: Registrar Alumno

### A.10.2. Editar Alumno

1. Desde la lista de alumno (Ver Figura N° 37), ubicarse en la fila en la que está el alumno, y dar clic en el botón que está en la columna “Acción”. Se nos cargara el formulario de edición de alumnos.
2. En el formulario (Ver Figura N° 39) editaremos la información que se desea modificar, y luego daremos clic en el botón azul “Guardar”. Este guardara el cambio y nos devolverá a la lista de alumnos.

En todos los formularios de mantenimiento de datos, si el usuario presiona clic sobre el botón gris “Regresar”, está dando por cancelada la operación para guardar un registro, y en la mayoría de los casos le retorna a la lista de datos.

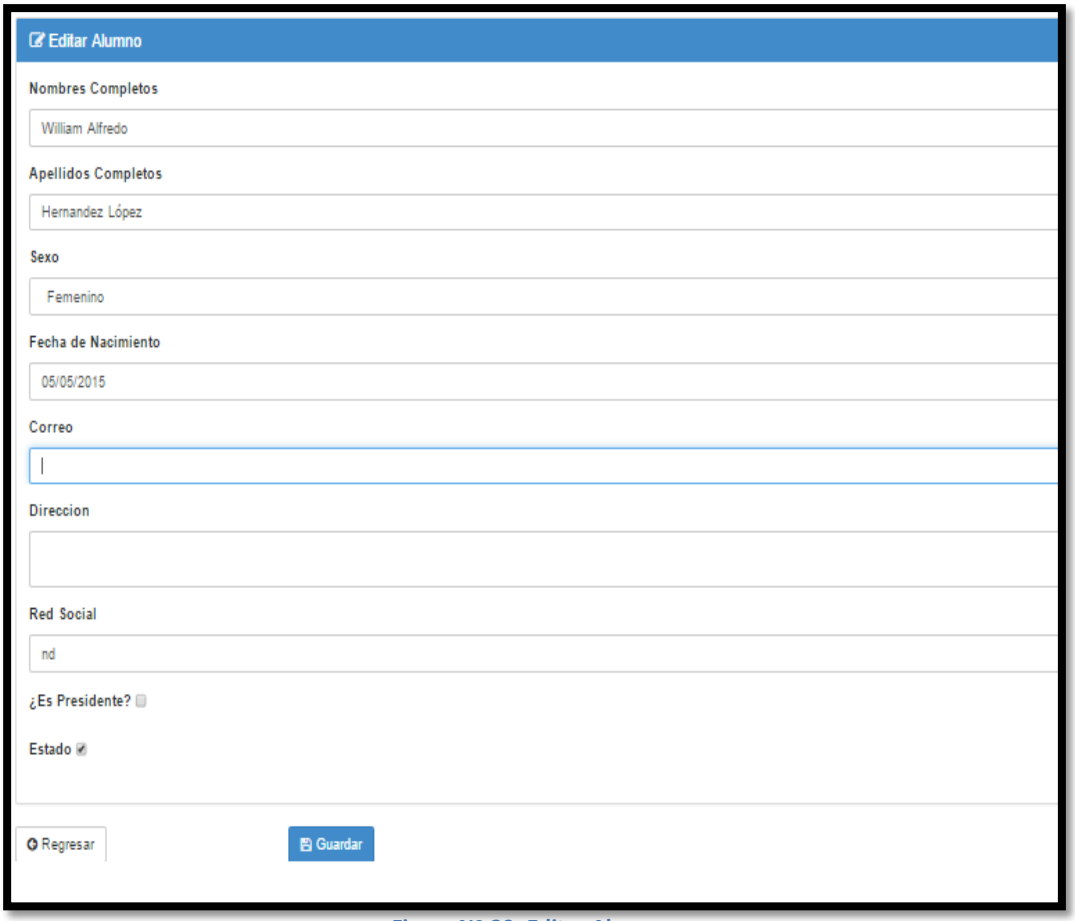



Figura Nº 39: Editar Alumno

## A.11. Alumnos por Periodo

El proceso de alumno por periodos, permite vincular o asociar al alumno con la carrera y el tipo de beca para un determinado periodo académico. Este es un paso importante y crítico para el control de los registros de servicio becario del alumno. Desde el formulario de lista de alumnos por periodos (ver Figura N° 40), se podrá agregar y editar la asociación de alumno por periodo.

Para este proceso solo se consideran los alumnos en estado activo. No puede existir más de un registro de alumno en el mismo periodo académico.


Figura Nº 40: Lista de Alumnos por Periodo

## A.12. Asociar Alumno a un Periodo

1. En la lista de alumno, presione clic sobre el botón azul “Agregar Asociación Alumno Periodo”. Esta acción nos lleva a un nuevo formulario (Ver Figura N° 40), donde se podrán seleccionar varios alumnos.
2. En el primer recuadro de la parte superior del formulario, se debe seleccionar el periodo, la carrera y el tipo de beca del alumno.
3. En el primer recuadro se debe buscar por nombres o número de carnet el alumno a asociar, y luego se debe presionar Enter o clic. Esta acción enlista al alumno en el segundo Recuadro del Formulario. Se puede repetir estos pasos tantas veces sea necesario asociar a los alumnos.
4. El último paso a realizar será guardar la lista de alumnos asociados, presionando clic sobre el botón azul “Guardar”, que está ubicado en el segundo recuadro del formulario.

Es importante tener en cuenta que en este formulario la asociación del alumno queda guardada hasta que se presiona clic sobre el botón azul “Guardar”. Si el usuario presiona clic sobre el botón gris “Regresar”, el sistema lo envía a la lista de alumnos por periodos, y no podrá recuperar la lista de alumno que anteriormente pudo realizar.


Figura Nº 41: Asociación Alumnos al Periodo Académico, Carrera y Tipo de Beca

## A.13. Tareas

En el Sistema las tareas son consideras actividades potenciales, que los alumnos podrán realizar. Este se considera como un catálogo del sistema, pero a manera de propuesta está ubicado en el grupo de “Servicio Becario”, esto con el fin de que el usuario logre identificar de manera más fácil las relaciones entre las entidades que se encuentran en el grupo de Servicio Becario.

Como se puede observar en la Figura N° 42, desde el formulario de tareas se pueden agregar nuevas tareas y editar las existentes. También se puede inactivar desde el formulario de edición de tareas.


Figura Nº 42: Lista de Tarea

Aprovechando que el Sistema trabaja bajo un esquema de interfaz de presentación y validación de información similar para todos los formularios de mantenimiento de datos (sobre todo en los catálogos), obviaremos los pasos de agregar y editar tareas.

## A.14. Asignación de Tareas

Este proceso tiene como objetivo la asociación del alumno por periodo con la tarea. El formulario que se presenta es similar a los anteriores, se muestra la lista de alumnos con tareas asignadas (Ver Figura N° 43). Se podrá realizar una asignación de tarea presionando clic sobre el botón azul “Agregar Asignación” y editar presionado clic sobre el botón editar que está en la columna de acción. Pero también tiene una leve diferencia, este es el botón gris “Ver Alumnos Sin Asignación” que permite ver a los alumnos del periodo activo que no tienen tarea asignada (Figura N° 44).


Figura Nº 43: Lista de Asignaciones


Figura Nº 44: Lista de Alumnos sin Tareas Asignadas

### A.14.1. Asignar Tarea a un Alumno con Periodo Académico Asociado

1. Desde la lista de asignaciones (Ver Figura N° 43), presionar clic sobre el botón azul “Agregar Asignación”.
2. Una vez en el formulario (Ver Figura N° 55), se debe seleccionar de la lista de tareas a asignar.
3. En la casilla “Buscar” se debe digitar el nombre o carnet del alumno para seleccionarlo.
4. Seleccionar la ficha inicial y final de la ejecución de la tarea. Al presionar clic en estas casillas emergerá un calendario del cual el usuario debe seleccionar o buscar la fecha.
5. Presionar clic en el botón azul “Guardar”. Este nos lleva nuevamente a la lista de asignación de tareas.

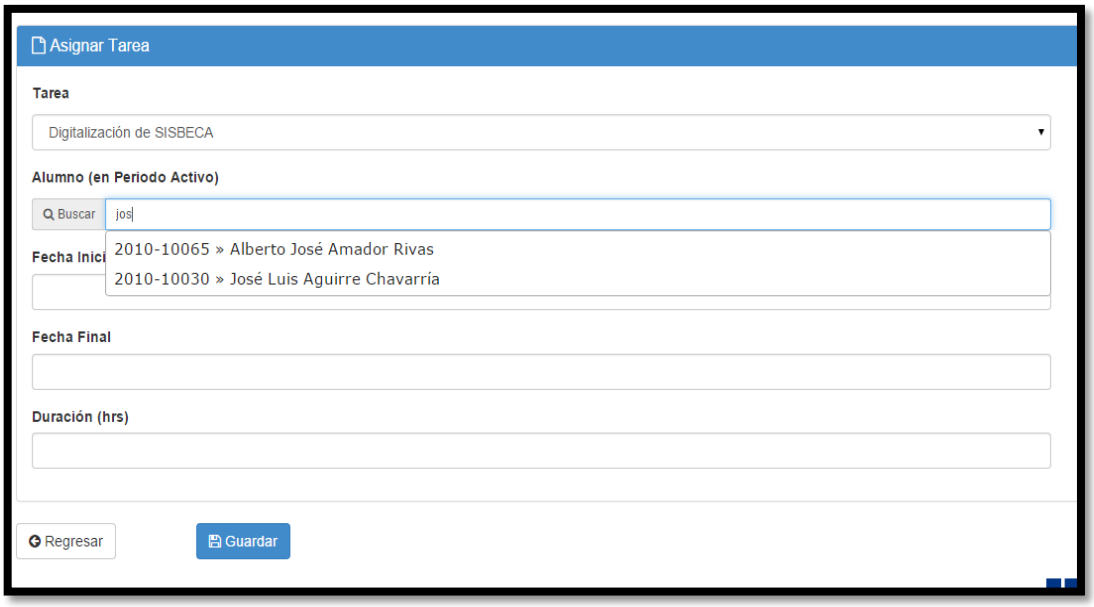



Figura Nº 45: Asignación de Tarea

### A.14.2. Asignación de Tarea desde la Lista de Alumnos sin Asignaciones

1. Se debe buscar al alumno al cual se le hará una asignación (Ver Figura N° 44), y presionar clic sobre el botón gris “Asignación”.
2. Una vez en el formulario (Ver Figura N° 45), seleccionar de la lista de tareas la tarea a asignar.
3. Realizar los pasos 4 y 5 de la anterior lista.

## A.15. Registro de Asistencia

Cuando los Becarios ya tienen tareas asignadas, en el sistema se deberá de registrar su asistencia, se ingresa: fecha de asistencia, hora de entrada, hora de salida y alguna observación que se considere agregar. Como se aprecia en la Figura N° 46, el Registro de Asistencia muestra la Lista de Asistencia, el botón azul de “Registrar Asistencia” y el botón de editar asistencia.

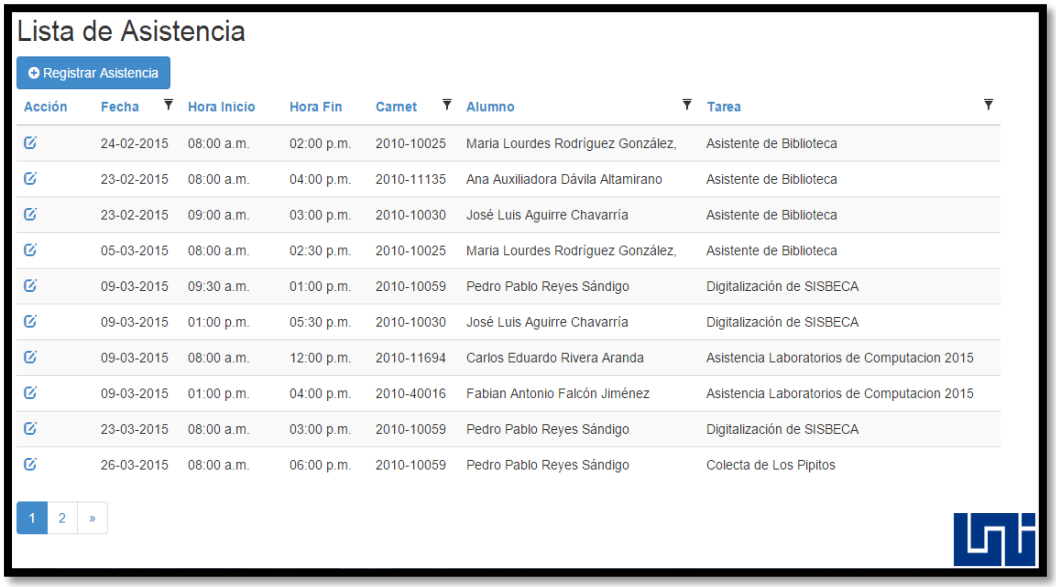



Figura Nº 46: Lista de Asistencia del Becario

1. En la lista de registro de asistencia (Ver Figura N° 46) presionar clic sobre el botón azul “Registrar Asistencia” de la lista de Asistencia.
2. En el formulario de registro de asistencia (Ver Figura N° 47), primero buscar al alumno en la casilla de búsqueda.

3. Seleccionar la fecha de asistencia, digitar hora de inicio (entrada) y fin (salida) para la asistencia. Si considera necesario puede ingresar una observación.
4. Presione clic sobre el botón azul “Guardar”.

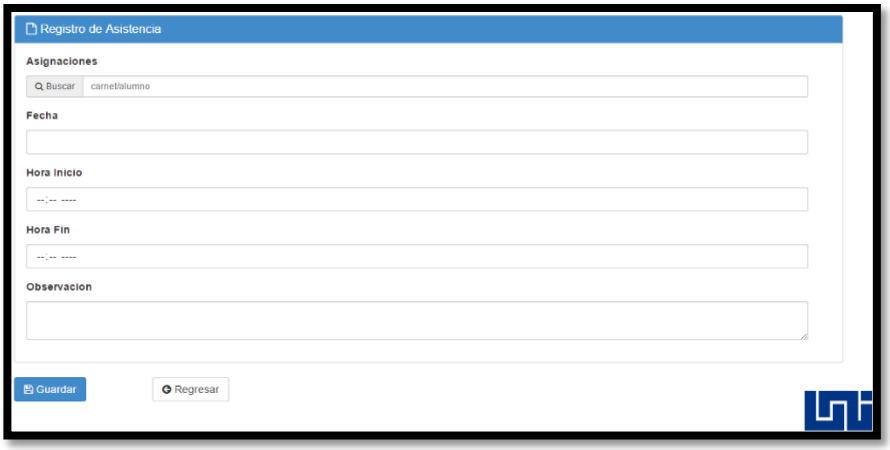



Figura Nº 47: Registro de Asistencia

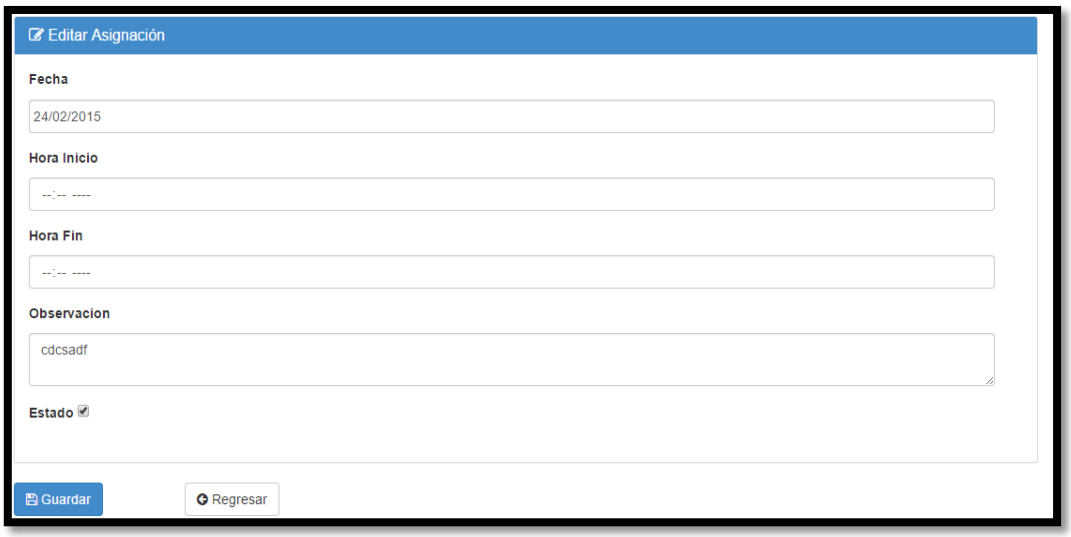



Figura Nº 48: Editar Asistencia del Estudiante

## A.16. Evaluar al Estudiante

La evaluación al estudiante se realiza para cada tarea que se le asigna. Es importante tener en cuenta que las tareas asignadas y becarios que se evaluarán, serán los becados en el periodo académico activo. La evaluación se realiza en dos partes, o más bien se puede ver como dos tipos de evaluaciones; la primera evaluación es a la tarea que el alumno realiza, y la segunda es una evaluación más detallada a distintos aspectos del becario. Para evaluar al estudiante lo primero que veremos será la lista de evaluaciones al estudiante o becario (Ver Figura N° 49).

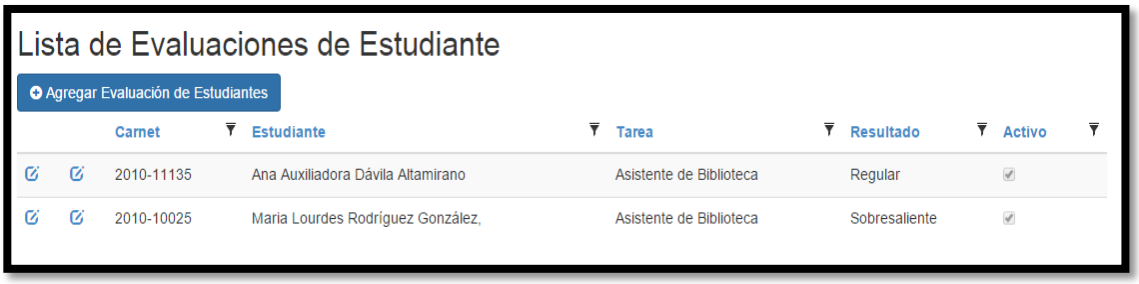



Figura Nº 49: Lista de Evaluaciones del Estudiante

1. Desde la lista de evaluaciones de estudiante (Ver Figura N° 49), presione clic sobre el botón azul “Agregar Evaluación de Estudiantes”. Este nos lleva al primer formulario de evaluación (Ver Figura N° 50).
2. Ya ubicado en el primer formulario de evaluación (Ver Figura N° 50), de la lista de asignación de estudiante, seleccione el estudiante, seleccione la evaluación con la que se valora la tarea, agregue opcionalmente observaciones y presione clic sobre el botón azul “Guardar y Evaluar Aspectos”.

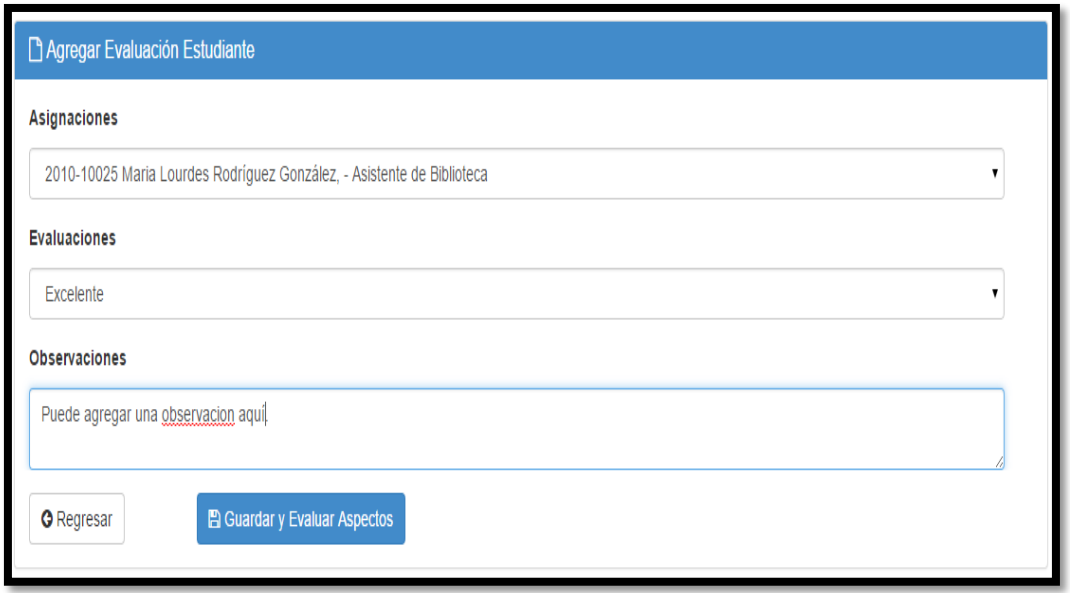



Figura Nº 50: Agregar Evaluación al Estudiante

## A.18. Evaluar Aspectos

Luego de evaluar al estudiante (primer formulario de evaluación), se cargara la evaluación a los aspectos de desempeño laboral, factor humano y habilidades (este es el segundo formulario de evaluación). Los pasos a seguir son:

1. Seleccionar la valoración de cada uno de los aspectos que se detallan en el formulario (Ver Figuras N° 51 y 52).
2. Presione clic sobre el botón azul “Guardar Evaluación Aspectos” (Ver Figura N° 52). Una vez realizada esta acción se cargara el primer formulario de la evaluación, dando opción al usuario de volver al listado de evaluaciones o editar la evaluación de la tarea.

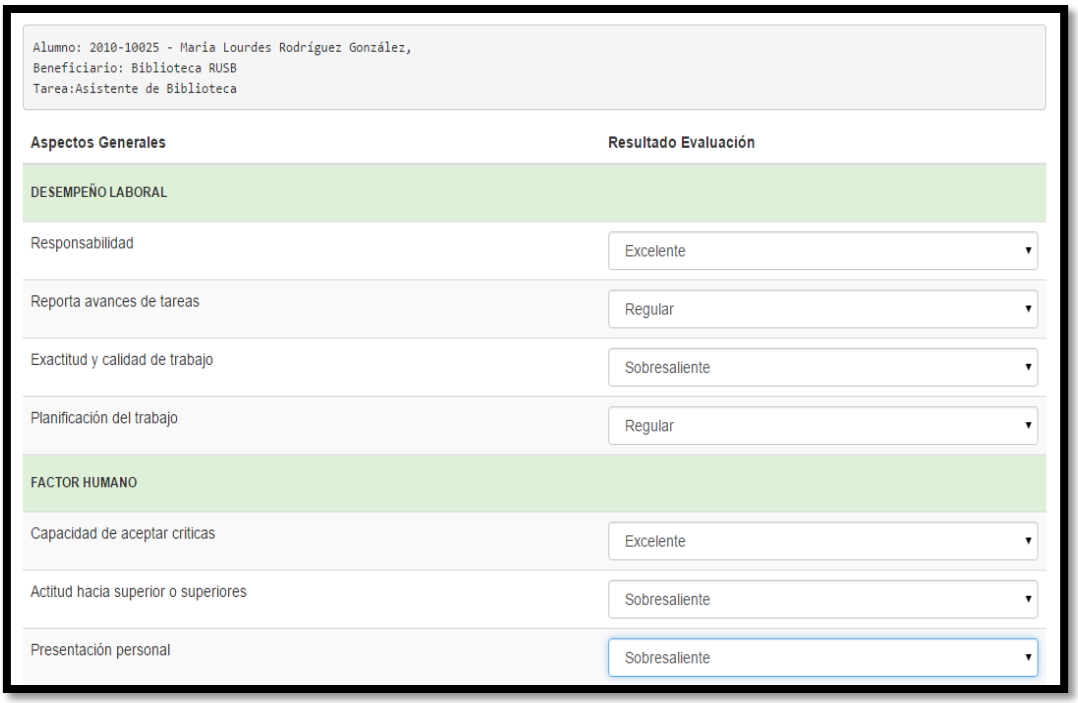



Figura Nº 51: Registrar Aspectos Generales

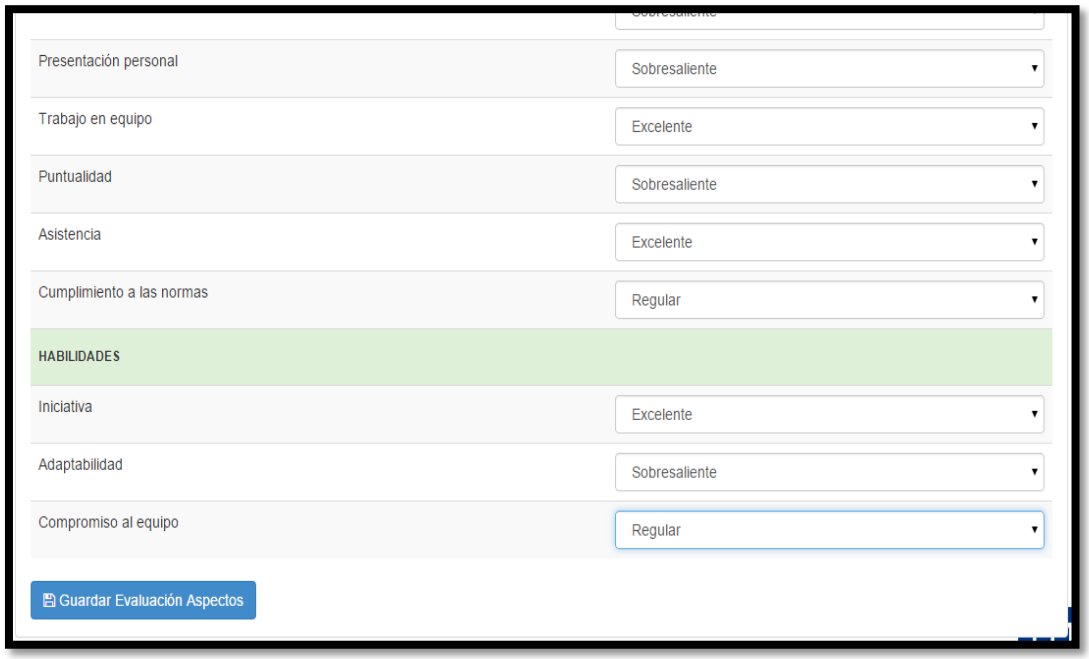



Figura Nº 52: Registrar Aspectos Generales

## A.19. Reportes

Para imprimir reportes ya sea en pantalla o archivos, lo primero que debemos hacer es ir al Menú Reportes y seleccionar con un Clic el reporte que deseamos ver.


Figura Nº 53: Menú Reporte

### A.19.1. Reporte Catálogo Alumnos

1. Seleccionamos el Menú Reportes.
2. Presionamos Clic sobre el Sub-Menú Reporte  Catálogo Alumno
3. Imprimir Reporte


Figura Nº 54: Reporte Catálogo Alumnos

### A.19.2. Reporte Consolidado de Horas por Periodo

1. Seleccione el Menú Reportes.
2. Presione Clic en el Sub-Menú Reporte Consolidado de horas por periodo
3. Seleccione el Periodo Académico
4. Imprime Reporte

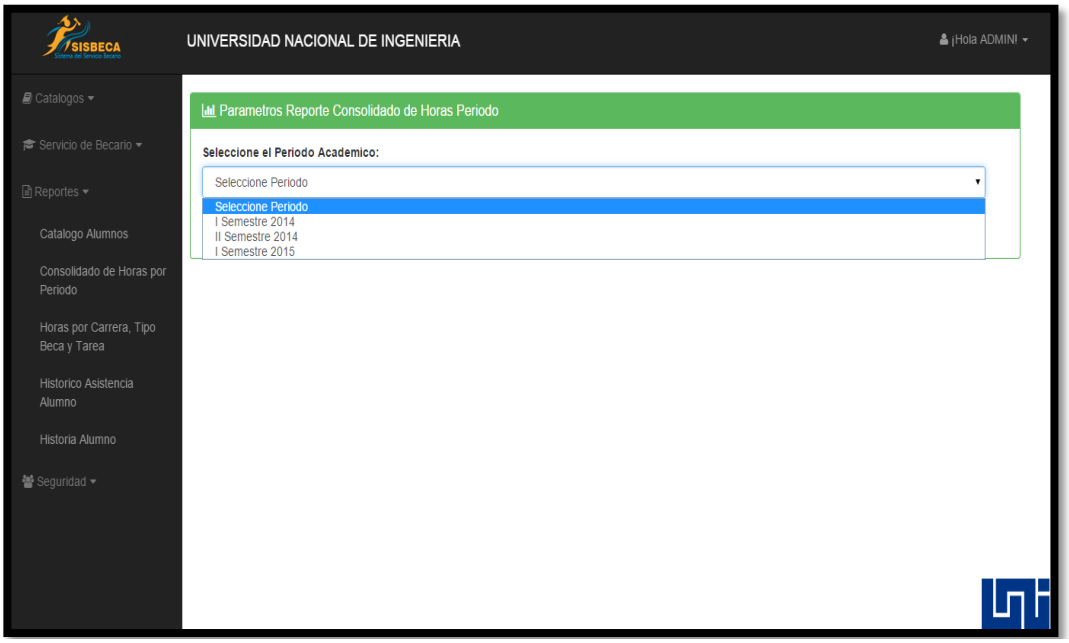



Figura Nº 55: Reporte Consolidado de Horas por Periodo

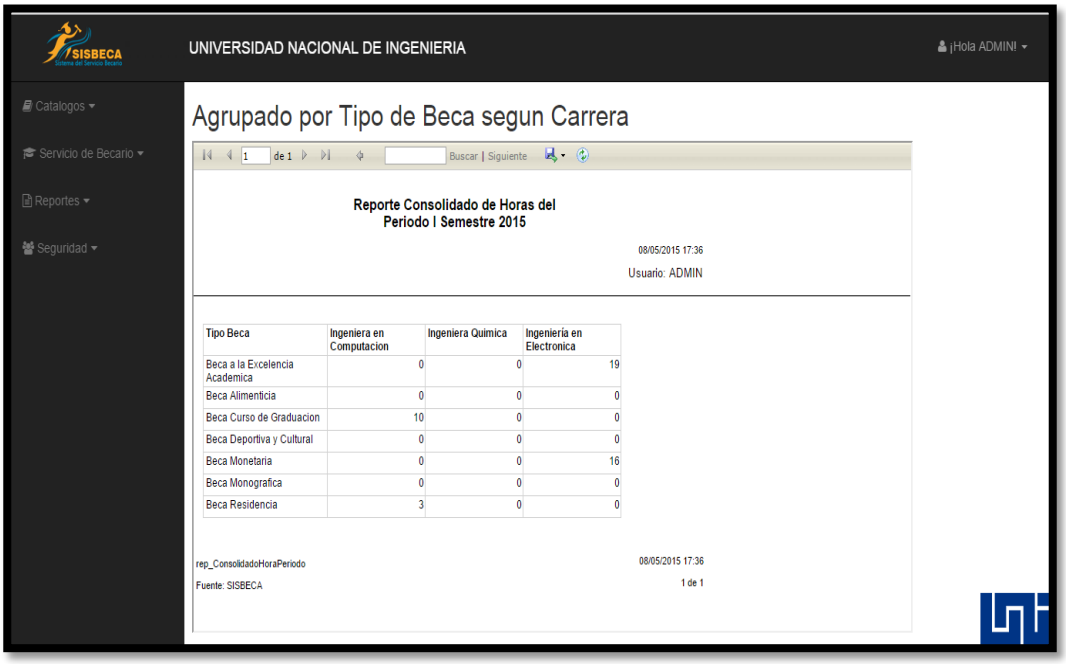



Figura Nº 56: Reporte Consolidado de Horas Agrupado por Tipo Beca

### A.19.3. Reporte de Historia Asistencia del Alumno

1. Seleccione el Menú Reportes.
2. Presione Clic sobre el Sub-Menú Reporte Historia Asistencia del Alumno
3. Seleccione el Carné del Alumno
4. Imprimir Reporte

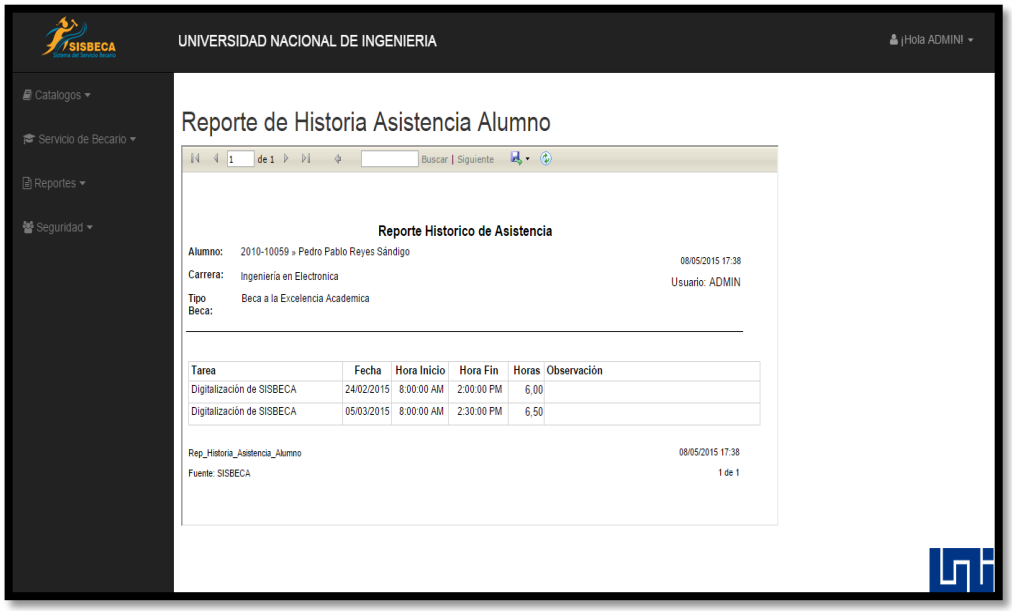



Figura Nº 57: Reporte Historia de Asistencia del Alumno

## A.20. Seguridad

Para Acceder al Menú Seguridad lo primero que debe hacer es presionar Clic sobre el Menú Seguridad y Acceder a los Sub-Menú deseados.


Figura Nº 58: Menú Seguridad

### A.20.1. Accesos

1. Presione Clic sobre el Menú Seguridad
2. Seleccione el Sub-Menú Accesos
3. Agregue el Usuario
4. Llene los campos
5. Aceptar

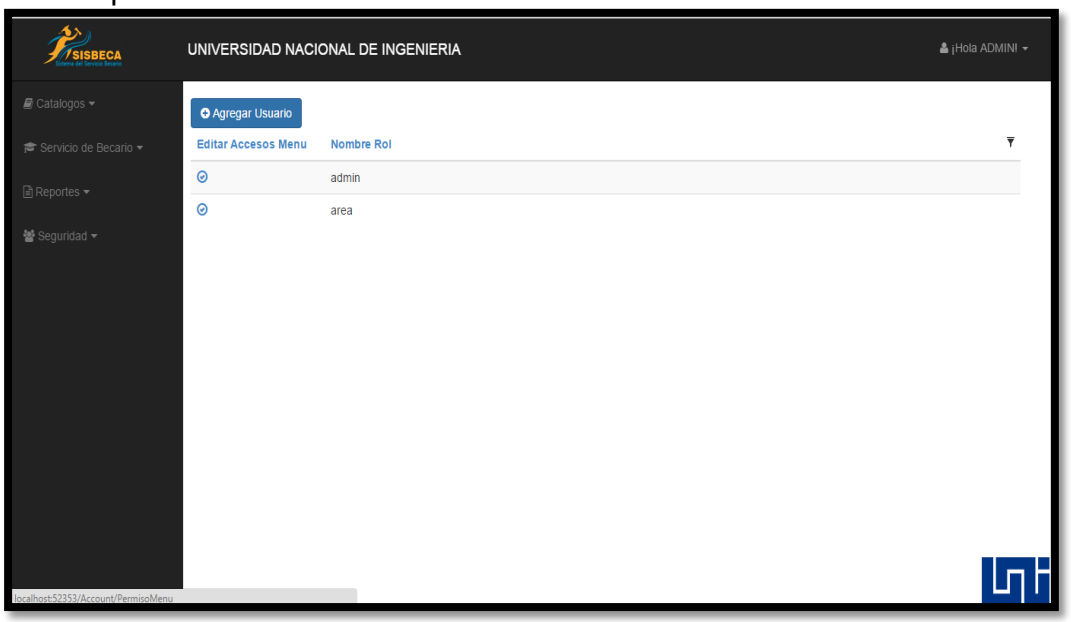



Figura Nº 59: Lista de Usuarios

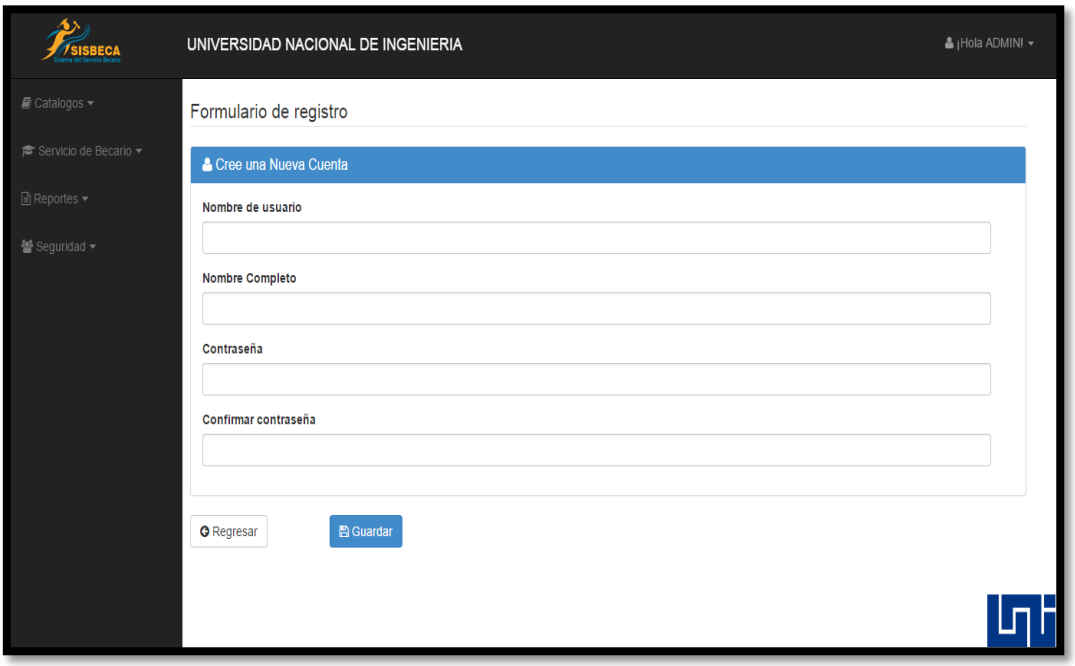



Figura Nº 60: Agregar Usuario

### A.20.2. Asignar Rol a Usuario

1. Presione Clic sobre el Menú Seguridad
2. Seleccione el Sub- Menú Administración
3. Se mostraran la lista de usuarios activos
4. Presione Clic sobre la Barra de Menú Asignar Rol
5. Seleccione el Rol del Usuario
6. Agregar Rol


Figura Nº 61: Lista de Usuarios

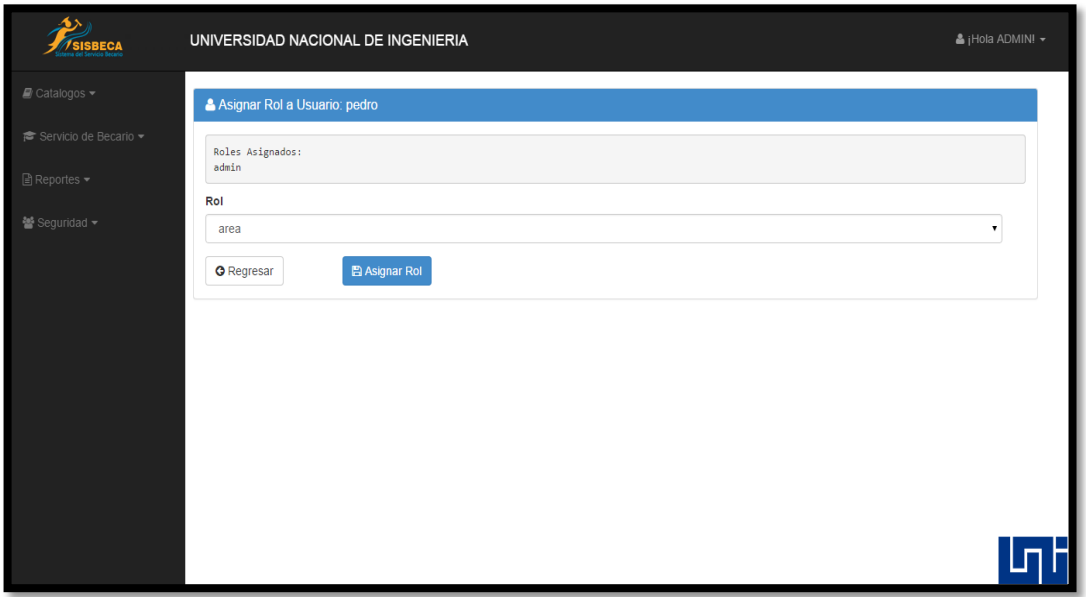



Figura Nº 62: Asignar Rol

# MANUAL TÉCNICO

## B. Concepción del Trabajo monográfico

### *Participantes:*

| Nombre | Cargo |
|---|---|
| María Lourdes Montes | Tutora |
| Erick Navarrete | Tesista |
| Pedro Morales | Tesista |
| Giselle Calero | Responsable Oficina Acción Social |

Tabla 25: Participantes del Proyecto

### *Reuniones estratégicas*

Lugar: Oficina Ing. María Lourdes Montes, Secretaria FEC.

Como parte de la primera reunión en la oficina de la Ing. María Lourdes Montes, se nos planteó el problema que existe en la Oficina de Acción Social, respecto al control del tiempo de servicio social que tienen que cumplir los estudiante becados de la UNI, planteándose la necesidad de un sistema para el seguimiento que se le tiene que dar al becario, en relación al tiempo de servicio y la oportunidad para desarrollar este sistema como un proyecto monográfico, con el fin de agilizar este proceso para la Oficina de Acción Social.

En la segunda reunión se realizó una entrevista no estructurada con la Responsable de la Oficina de Acción Social, Ing. Giselle Calero, donde se consultaron los siguientes aspectos: ¿Cuantos estudiantes becados atienden?, ¿Cuantos realizan servicio becario?, ¿Cuáles eran las causas principales del incumplimiento del servicio social?; a lo que ella respondió que atiende a más de 2,000 estudiantes becados, donde solo un 15% de los becarios realiza el servicio becario completo, las causas principales son la falta de interés y la falta de cumplimiento al llamado que realiza la Oficina de Acción Social.

Adicionalmente, nos entregó los formatos con los que cuenta la Oficina de Acción Social, donde se lleva el registro del estudiante, la actividad que está desempeñando, la hora de inicio-fin y la firma del becario, este formato se encuentra en las áreas que han solicitado el servicio social.

## C. Concepción del Sistema SISBECA

En la tercera reunión teniendo clara la situación del problema y la información relevante del control que se lleva en el Área, se planteó con la Ing. María Lourdes Montes, desarrollar un Sistema que ayude al control del Servicio Social Becario, posteriormente nos comprometimos con dicho proyecto.

En la cuarta reunión con la Ing. María Lourdes Montes, se llevó un avance sobre las funcionalidades del Sistema, también se platearon interrogantes como por ejemplo, información real de los estudiantes, donde se instalará el sistema, quien proporcionaría esta información de los estudiantes y si el sistema solo iba a estar instalado en la PC de la Responsable de la Oficina de Servicio Social Becario.

La Ing. María Lourdes Montes, se comunicaría con las instancias correspondientes para dar solución a las interrogantes planteadas por los monografistas.

Se habló de los Sistema existentes en la UNI como lo son el SIRA y el Sistema de Beca y que podríamos conectarnos a la base de datos de algunos de estos Sistema para tomar la información que alimentará nuestro Sistema.

En la quinta reunión la Ing. María Lourdes Montes ya se había reunido con el personal de la DITI y en base a esa reunión se llegó a lo siguiente:

### Acuerdos y Compromisos

1. Desarrollar el Sistema del Programa del Servicio Becario para la Oficina de Acción Social.
2. El sistema será en plataforma WEB.
3. El sistema se alojará en los servidores de la FEC.
4. No se tendrá acceso a la base de datos de los Sistema SIRA y Sistema de Beca por los mecanismos de seguridad existentes en la DITI.
5. El Sistema que se desarrolle para la Oficina de Acción Social será propiedad de la UNI, por lo tanto se tendrá que entregar código fuente, base de datos y todo material de ayuda que haya servido para el desarrollo del Sistema.
6. La DITI una vez entregado el sistema en la UNI, creará una interfaz de conexión entre el SIRA y SISBECA, lo que será responsabilidad de la DITI.
7. Capacitar a la Ing. Giselle Calero, para el uso del Sistema

## D. Formatos de evaluación y seguimiento del servicio de becario

UNIVERSIDAD NACIONAL DE INGENIERIA UNI
Dirección de Bienestar Estudiantil.
Rectoría

Servicio de Becario

**Formato de evaluación y desempeño del servicio de becario**

El presente documento es un instrumento para evaluar al becario en el servicio que desempeña actualmente, con el objetivo de medir la actitud del estudiante y su desempeño en el trabajo realizado.

La objetividad con que efectúe la evaluación determinará las mejoras en el desempeño del becario en el futuro.

| Nombre del estudiante | Carné | Carrera | Area |
|---|---|---|---|
| | | | |
| Beneficiario del servicio | Periodo de trabajo | Fecha | Periodo académico |
| | | | |

| LISTA DE TAREAS | DESEMPENO | | | |
|---|---|---|---|---|
| | Excelente | Bueno | Regular | Insuficiente |
| | | | | |
| | | | | |
| | | | | |

**Comentarios:**

Figura Nº 63: Formato de Evaluación y Seguimiento del Servicio Social Becario

UNIVERSIDAD NACIONAL DE INGENIERIA UNI
Dirección de Bienestar Estudiantil.
Rectoría

Servicio de Becario

| ASPECTOS GENERALES | NIVELES | | | |
|---|---|---|---|---|
| | Excelente | Bueno | Regular | Insuficiente |
| DESEMPENO LABORAL | | | | |
| Responsabilidad | | | | |
| Reporta avances de tareas | | | | |
| Exactitud y calidad de trabajo | | | | |
| Planificación del trabajo | | | | |
| FACTOR HUMANO | | | | |
| Capacidad de aceptar criticas | | | | |
| Actitud hacia superior o superiores. | | | | |
| Presentación personal | | | | |
| Trabajo en equipo | | | | |
| Puntualidad | | | | |
| Asistencia | | | | |
| Cumplimiento a las normas | | | | |
| HABILIDADES | | | | |
| Iniciativa | | | | |
| Adaptabilidad | | | | |
| Compromiso al equipo | | | | |

**Total de horas trabajadas:**_____

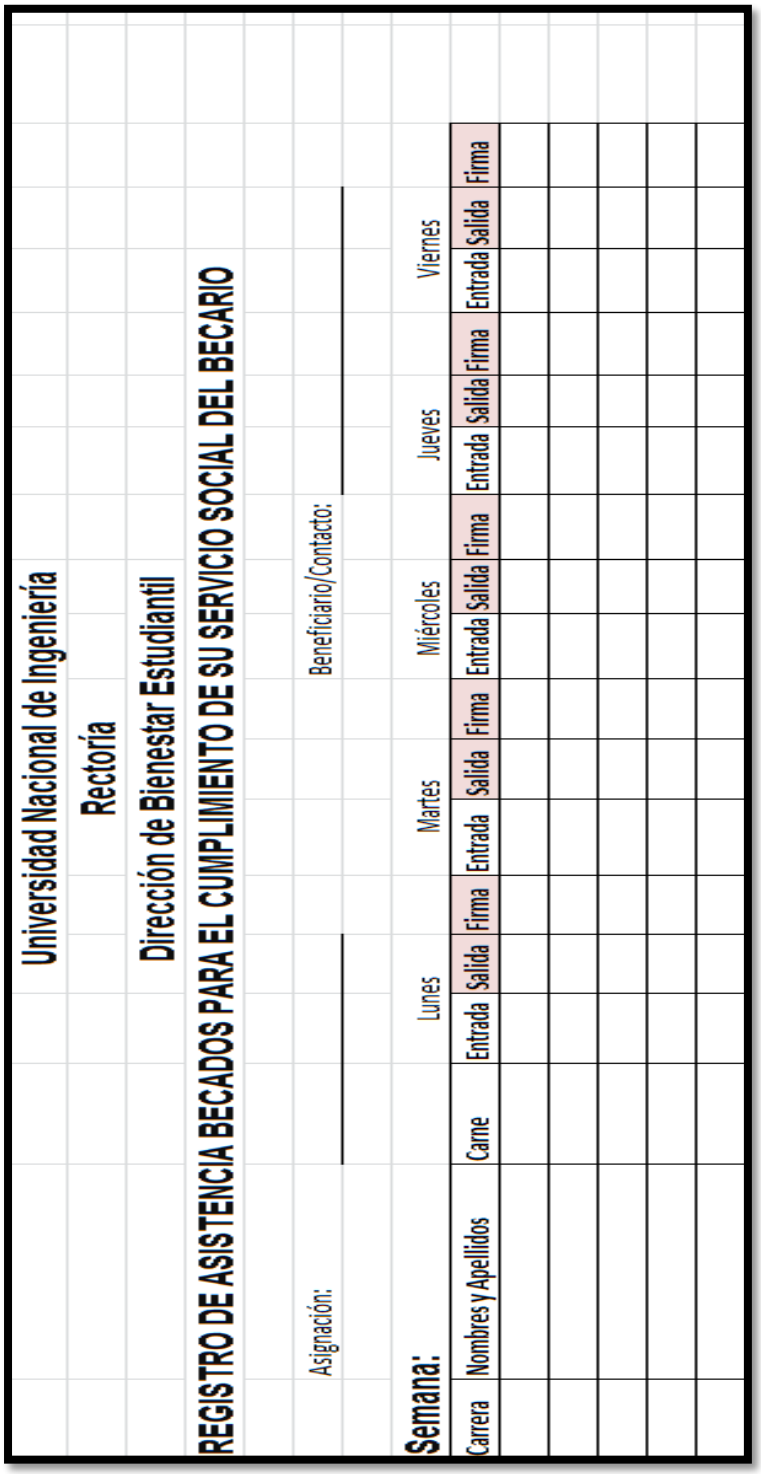
**NOTA IMPORTANTE: Las únicas evaluaciones que no condicionan el SERVICIO BECARIO, es obtener EXCELENTE o BUENO. Esta forma deberás devolverla a la DBE ya firmada.**

_____ _____

**Firma y sello del beneficiario** **Firma del Alumno**

Figura Nº 64: Formato de Evaluación y Seguimiento al Becario

## E. Formato registro de asistencia becados para el cumplimiento de su servicio social del becario.



Universidad Nacional de Ingeniería

Rectoría

Dirección de Bienestar Estudiantil

REGISTRO DE ASISTENCIA BECADOS PARA EL CUMPLIMIENTO DE SU SERVICIO SOCIAL DEL BECARIO

Asignación: _____ Beneficiario/Contacto: _____

Semana:

| Carrera | Nombres y Apellidos | Carne | Lunes | | | Martes | | | Miércoles | | | Jueves | | | Viernes | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Entrada | Salida | Firma | Entrada | Salida | Firma | Entrada | Salida | Firma | Entrada | Salida | Firma | Entrada | Salida | Firma |
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |

Figura Nº 65: Formato de Registro de Asistencia del Becario

## F. Entrevista Utilizada

UNIVERSIDAD NACIONAL DE INGENIERIA
FEC
INGENIERIA EN COMPUTACION

ENTREVISTA A LA DIRECCIÓN DE BIENESTAR ESTUDIANTIL (DBE) y OFICINA DE ACCIÓN SOCIAL

La presente entrevista tiene como finalidad conocer los procesos del DBE y la Oficina de Acción Social, para realizar un mejor diseño del sistema.

| DATOS DEL ENTREVISTADO | |
|---|---|
| **NOMBRE Y APELLIDOS** | |
| **CARGO** | |
| **TELEFONO Y/O CELULAR** | |
| **E-MAIL** | |

Tabla 26: Datos del Entrevistado

**PREGUNTAS:**

1. ¿Qué es la Dirección de bienestar Estudiantil?
2. ¿Cuál es la Función y quien está a cargo de esta Dirección?
3. ¿Cómo surge la Oficina de Acción Social?
4. ¿Cuál es la función de la oficina de Acción Social?
5. ¿Qué función tiene la Responsable de la oficina de Acción Social?
6. ¿Cuántas áreas y/o direcciones están relacionadas con la oficina de Acción Social?
7. ¿Cuáles son los tipos de becas que pueden poseer los alumnos becarios?
8. ¿Cuántos becarios atienden la oficina de Acción Social?
9. ¿Todos estos becarios realizan tiempo de servicio social becario?
10. ¿En qué áreas o actividades se desempañaran los becarios, Menciónelas?
11. ¿Cuál es el procedimiento para que las áreas y/o direcciones acepten que el alumno becado realice su tiempo de servicio social becario?
12. ¿Existe un formulario y/o formato para llevar el control del tiempo de servicio social que realizan los becarios?
13. ¿El Formulario y/o formato se asigna a un responsable del área?
14. ¿Cuál es el proceso de retorno del formulario y/o formato a la oficina de Acción social?
15. ¿Qué se hace cuando este formulario está lleno y en manos de la responsable de la Oficina de Acción social?
16. ¿Es evaluado el tiempo servicio social que realiza el becario? De ser así, ¿Quien realiza la evaluación?
17. ¿Qué técnicas e instrumentos de evaluación se realizan al cumplimiento del servicio social del alumno?
18. ¿Qué objetivo lleva la realización de la evaluación al servicio social del becario?
19. ¿A quién se rinde cuenta del tiempo de servicio social que realiza el becario?
20. ¿Qué hacen con todos los reportes de tiempo de servicio social que se dan a conocer?
21. ¿Ha habido anulación de becas por no cumplir el tiempo de servicio social becario?

## G. Diccionario de Datos

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Alumno** | Tabla de Catalogo del alumno y/o becario. | | | | |
| Carnet | PK | Identificador del Alumno | nvarchar | 32 | |
| Nombres | | Nombre del alumno | nvarchar | 200 | |
| Apellidos | | Apellidos del alumno | nvarchar | 200 | |
| Sexo | | Sexo del alumno | nvarchar | 2 | |
| FechaNacimiento | | Fecha de nacimiento del alumno | datetime | 8 | |
| FechaRegistro | | Fecha en que se registró en el sistema | datetime | 8 | |
| Correo | | Email del alumno | nvarchar | 200 | |
| Direccion | | Dirección domiciliar del alumno | nvarchar | 300 | |
| RedSocial | | Cuenta de red social del alumno | nvarchar | 200 | |
| Estado | | Estado del alumno | bit | 1 | |
| Convencional | | Teléfono del alumno | nvarchar | 24 | |
| Movistar | | Celular movistar del alumno | nvarchar | 24 | |
| Claro | | Celular claro del alumno | nvarchar | 24 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: AlumnoPeriodo** | Tabla donde asocia al alumno con: Carrera, periodo académico y tipo de beca | | | | |
| IdPeriodoAlumno | PK | Identificador de la asociación PeriodoAlumno | int | 4 | |
| Carnet | FK | Carnet del Alumno ID | nvarchar | 32 | Alumno |
| IdPeriodoAcademico | FK | Periodo Académico ID | int | 4 | Periodo Académico |
| IdCarrera | FK | Carrera ID | int | 4 | Carrera |
| IdTipoBeca | FK | Tipo de Beca ID | int | 4 | TipoBeca |
| FechaCreacion | | Fecha en que se registró la asociación | datetime | 8 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Área** | Catálogo de Área de la institución. | | | | |
| IdArea | PK | Identificador del área | int | 4 | |
| Nombre | | Nombre del área | nvarchar | 240 | |
| Estado | | Estado del área | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Asignación** | Tabla donde se registra la tarea que se asigna al becario identificado por el campo llave de asociación de alumno periodo. | | | | |
| IdAsignacion | PK | Identificador de la asignación de tarea al becario | int | 4 | |
| TareaId | FK | Tarea ID | int | 4 | Tarea |
| IdAlumnoPeriodo | FK | Asociación Alumno Periodo ID | int | 4 | AlumnoPeriodo |
| Id Evaluación | FK | Evaluación ID | int | 4 | Evaluación |
| FechaInicio | | Fecha de inicio de la asignación | datetime | 8 | |
| FechaFin | | Fecha en que se debe terminar la asignación | datetime | 8 | |
| Duración | | Duración estimada en horas de la asignación | decimal | 9 | |
| Estado | | Estado de la asignación | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Cuenta Asistencia** | Tabla donde se registra el día a día de asistencia para una asignación del becario. | | | | |
| IdAsistencia | PK | Identificador de la asistencia del alumno. | int | 4 | |
| AsignacionId | FK | Asignación ID | int | 4 | Asignación |
| Fecha | | Fecha de asistencia. | datetime | 8 | |
| FechaRegistro | | Fecha en que se registró en el sistema | datetime | 8 | |
| HoraInicio | | Hora a la que llego el becario | datetime | 8 | |
| HoraFin | | Hora de salida del becario | datetime | 8 | |
| Observación | | Información adicional que de manera opcional se puede agregar | nvarchar | 360 | |
| Estado | | Estado e la asistencia. | bit | 1 | |
| Evaluacion_IdEvaluacion | FK | Evaluación | int | 4 | Evaluación |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: AspectosEvaluar** | Tabla catálogo de aspectos del beneficiarios que se evaluaran. | | | | |
| IdAspecto | PK | Identificador del registro de aspecto a evaluar | int | 4 | |
| Tipo | FK | Tipo de Aspecto | int | 4 | TipoAspecto |
| Aspecto | | Descripción del aspecto | nvarchar | 240 | |
| Estado | | Estado del aspecto | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Auditoria** | Bitácora de transacciones de las tablas del sistema. Las de seguridad están en otra tabla. | | | | |
| Id | PK | Identificador del registro de auditoria | int | 4 | |
| Fecha | | Fecha del registro | datetime | 8 | |
| Referencia | | Número identificador del registro de la tabla | nvarchar | 40 | |
| Objeto | | Tabla o entidad del que se está realizando la auditoria | nvarchar | 40 | |
| Acción | | Indica el tipo de transacción sql | nvarchar | 100 | |
| Usuario | | Usuario del sistema que realiza la transacción | nvarchar | 30 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Beneficiario** | Catálogo de Beneficiarios de las tareas, que realizan los becarios. | | | | |
| IdBeneficiario | PK | Identificador de los beneficiarios del sistema | int | 4 | |
| Nombre | | Nombre del beneficiario | nvarchar | 200 | |
| Representante | | Nombre del representante del beneficiario | nvarchar | 200 | |
| Teléfono | | Teléfono del beneficiario | nvarchar | 200 | |
| Celular | | Celular del beneficiario | nvarchar | 200 | |
| Dirección | | Domicilio del beneficiario | nvarchar | 200 | |
| Correo | | Email del beneficiario | nvarchar | 200 | |
| Activo | | Estado del beneficiario | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Carrera** | Catálogos de la carrera de la universidad. | | | | |
| IdCarrera | PK | Identificador de la carrera | int | 4 | |
| IdFacultad | FK | Facultad ID | int | 4 | Facultad |
| Nombre | | Nombre de la carrera | nvarchar | 200 | |
| Estado | | Estado de la carrera | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Evaluación** | Catálogo de posibles resultado de las evaluaciones. | | | | |
| Id Evaluación | | Identificador de la evaluación | int | 4 | |
| Descripción | | Descripción de la evaluación | nvarchar | 60 | |
| Valor | | Valor de la evaluación | int | 4 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: EvaluacionAspectos** | Tabla de resultado de la evaluación a cada uno de los aspectos que se evalúan al becario para cada tarea asignada. | | | | |
| Id | | Identificador de la evaluación de aspecto | int | 4 | |
| IdEvaluacionEstudiante | FK | Evaluación Estudiante ID | int | 4 | EvaluacionEstudiante |
| IdAspecto | FK | Aspecto ID | int | 4 | AspectosEvaluar |
| ResultadoAspecto | FK | Resultado Aspecto ID | int | 4 | Evaluación |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: EvaluacionEstudiante** | Tabla de resultado de la evaluación de la tarea asignada del estudiante | | | | |
| Id | PK | Identificador de la evaluación a la asignación del estudiante | int | 4 | |
| Id Asignación | FK | Asignación ID | int | 4 | Asignación |
| Fecha Evaluación | | Fecha en que se realizó la evaluación | datetime | 8 | |
| ResultadoTarea | FK | Resultado Tarea ID | int | 4 | Evaluación |
| Observaciones | | Observación opcional de la evaluación | nvarchar | 300 | |
| Usuario | | Usuario que realiza la evaluación | nvarchar | 100 | |
| Estado | | Estado de la evaluación | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Facultad** | Catálogo de facultad de la universidad. | | | | |
| IdFaculta | PK | Identifica la facultad | int | 4 | |
| IdRecinto | FK | Recinto ID | int | 4 | Recinto |
| Nombre | | Nombre de la facultad | nvarchar | 200 | |
| NombreCorto | | Siglas de la facultad | nvarchar | 20 | |
| Estado | | Estado de la facultad | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Menú** | Catalogo encabezado de menú del sistema | | | | |
| Id | PK | Identificador del encabezado del menú | int | 4 | |
| Nombre | | Nombre del encabezado del menú | nvarchar | -1 | |
| glyp | | Imagen opcional del menú | nvarchar | 100 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: MenuItem** | Catalogo detalle de menú del sistema | | | | |
| Id | PK | Identificador del detalle de menú | int | 4 | |
| Nombre | | Nombre del detalle de menú | nvarchar | -1 | |
| ActionName | | Nombre de la acción del controlador | nvarchar | -1 | |
| ControllerName | | Nombre del controlador | nvarchar | -1 | |
| Url | | URL de acceso al acción del controlador, opcional | nvarchar | -1 | |
| ParentMenu | FK | Nombre del Menú ID | int | 4 | Menú |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: PeriodoAcademico** | Catálogo de Periodo académico | | | | |
| IdPeriodoAcademico | PK | Identificador Periodo Académico | int | 4 | |
| Ciclo | | Descripción del periodo | nvarchar | 200 | |
| Estado | | Estado del periodo | bit | 1 | |
| anio | | Año | int | 4 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Recinto** | Catálogo de Recinto. | | | | |
| IdRecinto | PK | Identificador del recinto | int | 4 | |
| Nombre | | Nombre del recinto | nvarchar | 200 | |
| NombreCorto | | Siglas del recinto | nvarchar | 20 | |
| Estado | | Estado del recinto | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: Tarea** | Catálogo de tareas que se asignaran a los becarios. | | | | |
| IdTarea | PK | Identificador de la tarea | int | 4 | |
| TipoTareaId | FK | Tipo de Tarea ID | int | 4 | TipoTarea |
| IdBeneficiario | FK | Beneficiario ID | int | 4 | Beneficiario |
| IdArea | FK | Área ID | int | 4 | Área |
| Nombre | | Nombre de la tarea | nvarchar | 300 | |
| Lugar | | Lugar donde se realizara la tarea | nvarchar | 300 | |
| Estado | | Estado de la tarea | bit | 1 | |
| PeriodoAcademico_IdPeriodoAcademico | FK | Periodo Académico ID | int | 4 | Periodo Académico |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: TipoAspecto** | Catálogo de tipo aspecto. | | | | |
| IdTipoAspecto | PK | Identificador tipo aspecto | int | 4 | |
| Tipo | | Nombre de tipo aspecto | nvarchar | 100 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: TipoBeca** | Catálogo de tipo de beca. | | | | |
| IdTipoBeca | PK | Identificador de tipo de beca | int | 4 | |
| Nombre | | Nombre de tipo de beca | nvarchar | 160 | |
| Estado | | Estado del tipo de beca | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: TipoTarea** | Catálogo de tipo de tarea | | | | |
| IdTipoTarea | PK | Identificador Tipo de Tarea | int | 4 | |
| Descripción | | Descripción tipo de beca | nvarchar | 160 | |
| Estado | | Estado tipo de tarea | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: UserPermisoMenu** | Tabla de asociación de acceso del usuario al menú del sistema, es decir a que funciones tendrá acceso y cuales solo podrá ver. | | | | |
| RoleId | PK | Identificador complementario de la tabla | int | 4 | |
| MenuItemId | PK | Identificador complementario de la tabla | int | 4 | |
| Acceso | | Indicador del acceso | bit | 1 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: UserProfile** | Catálogo de Usuario | | | | |
| UserId | PK | Identificador del usuario | int | 4 | |
| UserName | | Nombre con el que el usuario accede al sistema | nvarchar | max | |
| NombreCompleto | | Nombre real del usuario | nvarchar | 200 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Tabla: webpages_Membership** | Tabla de bitácora de seguridad del sistema. | | | | |
| UserId | PK | Identificador del usuario | int | 4 | |
| CreateDate | | Fecha de creación | datetime | 8 | |
| ConfirmationToken | | Identificador de confirmación para transacciones con la contraseña | nvarchar | 256 | |
| IsConfirmed | | Para verificar confirmación | bit | 1 | |
| LastPasswordFailureDate | | Ultimo día de password fallido | datetime | 8 | |
| PasswordFailuresSinceLastSuccess | | Intentos fallidos de acceso al sistema | int | 4 | |
| Password | | Valor de contraseña | nvarchar | 256 | |
| PasswordChangedDate | | Fecha en que se cambió la contraseña | datetime | 8 | |
| PasswordVerificationToken | | Identificador de verificación de token | nvarchar | 256 | |
| PasswordVerificationTokenExpirationDate | | Identificador de verificación de la fecha en que caduca del password | datetime | 8 | |

| ATRIBUTO | LLAVE | DESCRIPCION | TIPO DE DATO | LONGITUD | ENTIDAD FORANEA |
|---|---|---|---|---|---|
| **Cuenta webpages_Roles** | Catálogo de Roles | | | | |
| RoleId | PK | Identificador del rol | int | 4 | |
| RoleName | | Descripción del rol | nvarchar | 512 | |
| **ATRIBUTO** | **LLAVE** | **DESCRIPCION** | **TIPO DE DATO** | **LONGITUD** | **ENTIDAD FORANEA** |
| **Cuenta webpages_UsersInRoles** | Tabla de asociación de roles y usuarios | | | | |
| UserId | PK | Identificador de Usuario | int | 4 | UserProfile |
| RoleId | PK | Identificador de Rol | int | 4 | Webpages_ Roles |

**Tabla 27: Diccionario de Datos**

# H. Descripción de casos de uso

## H.1. Registrar Usuarios

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU1)** | : | Registrar Usuarios | | |
| **Definición** | : | Registra los datos generales de los usuarios, guardar clave de acceso. | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Administrador | | Encargado de registrar los datos generales de los usuarios y Claves. | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de información de los usuarios | | |
| **Pre-Condiciones** | : | Administrador solicita información de los usuarios y asigna la Clave. | | |
| **Iniciado por** | : | Administrador | | |
| **Finalizado por** | : | Administrador | | |
| **Post-Condiciones** | : | El Administrador puede registrar exitosamente a los usuarios del sistema y asignación de Clave. | | |
| **Operaciones** | : | 1. Selecciona menu Seguridad<br>2. Dar click en el Sub Menu Usuario<br>3. Dar click en el Botón Agregar Usuario<br>4. Llenar el formulario de usuarios<br>5. Guardar los datos de los usuarios | | |
| **Excepciones** | : | 6. Guardar : Datos incorrectos. La base de datos no ingresa los registros correspondientes de los usuarios.<br>7. Guardar: No se guardan usuarios con nombres vacios.<br>8. Guardar: No se agrega un Usuario que ya existe. | | |

Tabla 28: Caso de Uso Registrar Usuario

## H.2. Gestionar Roles

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU2)** | : | Gestionar Roles | | |
| **Definición** | : | Registra los datos de los Roles | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Administrador | Encargado de registrar los datos de los roles en el sistema. | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos de los roles | | |
| **Pre-Condiciones** | : | Administrador solicita información de los roles para los usuarios | | |
| **Iniciado por** | : | Administrador | | |
| **Finalizado por** | : | Administrador | | |
| **Post-Condiciones** | : | El administrador puede registrar exitosamente los roles en el sistema | | |
| **Operaciones** | : | 1. Selecciona menu Seguridad<br>2. Dar Click en el submenu Rol<br>3. Dar click en Agregar Rol<br>4. Llenar el formulario del Rol<br>5. Guardar los datos de los Rol | | |
| **Excepciones** | : | 6. Guardar : Datos incorrectos. La base de datos no ingresa los datos correspondientes de los roles.<br>7. Guardar: El Nombre del Rol no se puede repetir<br>8. Guardar: El nombre del Rol no puede ir vacio. | | |

Tabla 29: Caso de Uso Registrar Roles

## H.3. Gestionar Permisos de Roles

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU3)** | : | Gestionar Permisos de Roles | | |
| **Definición** | : | Asignación de Roles | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Administrador | | Encargado de Asignar el Rol a los Usuarios del Sistema | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de Asignación de Rol | | |
| **Pre-Condiciones** | : | Responsable de la Oficina de Acción Social define los permisos que tendrán los Usuarios en el Sistema. | | |
| **Iniciado por** | : | Administrador | | |
| **Finalizado por** | : | Administrador | | |
| **Post-Condiciones** | : | El Administrador puede Asignar exitosamente los permisos de cada Rol del Sistema | | |
| **Operaciones** | : | 1. Selecciona menu Seguridad<br>2. Dar Click en el submenu Administración de Usuario<br>3. Dar Click en el Icono Asignar Rol<br>4. Seleccionar el Rol<br>5. Dar Click en Asignar Rol para Guardar los datos | | |
| **Excepciones** | : | Guardar : Datos incorrectos. La base de datos no ingresa los permisos correspondientes de los roles. | | |

Tabla 30: Caso de Uso Gestionar Permisos de Roles

### H.4. Gestionar Catálogo Recinto

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU4)** | : | Gestionar Catálogo Recinto | | |
| **Definición** | : | Registra la información del Recinto de la Institución. | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social | | Encargado de registrar los datos del Recinto. | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos del Recinto. | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información del Recinto Universitario. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos del Recinto Universitario. | | |
| **Operaciones** | : | 4 Selecciona menu Catálogo<br>5 Dar Click en el submenu Recinto<br>6 Dar Click en agregar Recinto<br>7 LLenar el formulario correspondiente a los datos del Recinto<br>8 Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del Recinto.<br>2. Guardar: No se registran Recintos sin nombres ni duplicados. | | |

Tabla 31: Caso de Uso Gestionar Catálogo Recinto

## H.5. Gestionar Catálogo de Facultad

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU5)** | : | Gestionar Catálogo de Facultad | | |
| **Definición** | : | Registra la información de la Facultad | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos de la Facultad | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos de la Facultad | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información de la Facultad, Ademas tiene que estar ingresado en el sistema los datos del Recinto. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos de la Facultad. | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Facultades<br>3. Dar Click en agregar Facultad<br>4. LLenar el formulario correspondiente a los datos de la Facultad<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos de la Facultad.<br>2. Guardar: No se registran Facultades sin Nombre. | | |

Tabla 32: Caso de uso Gestionar Catálogo Facultad

## H.6. Gestionar Catálogo de Carrera

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU6)** | : | Gestionar Catálogo de Carrera | | |
| **Definición** | : | Registra la información de la Carrera | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos de la Carrera | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos de la Carrera | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información de la Facultad, Ademas tienen que estar ingresados en el sistema los datos del Recinto y la Facultad. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos de la Carrera. | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Carreras<br>3. Dar Click en agregar Carrera<br>4. LLenar el formulario correspondiente a los datos de la Facultad<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos de la Carrera.<br>2. Guardar: No se registran carreras sin nombre ni duplicadas. | | |

Tabla 33: Caso de Uso Gestionar Catálogo Carrera

### H.7. Gestionar Catálogo Periodo Académico

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU7)** | : | Gestionar Catálogo Periodo Académico | | |
| **Definición** | : | Registra la información del Periodo Académico | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos del Periodo Académico | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos del Periodo Académico | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información Periodo Académico | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos del Periodo Académico | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Periodo Académico<br>3. Dar Click en agregar Periodo Académico<br>4. LLenar el formulario correspondiente a los datos del Periodo Académico<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del Periodo Académico.<br>2. Guardar: Solo debe existir un periodo académico activo.<br>3. Guardar: No se registran periodos Académicos con el mismo nombre. | | |

Tabla 34: Caso de Uso Gestionar Catálogo Periodo Académico

## H.8. Gestionar Catálogo de Tipo de Becas

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU8)** | : | Gestionar Catálogo del Tipo de Beca | | |
| **Definición** | : | Registra la información del Tipo de Beca | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos de la Beca | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos del Tipo de Beca | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información de los tipos de becas | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos de la Beca | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Tipo de Beca<br>3. Dar Click en agregar Tipo Beca<br>4. LLenar el formulario correspondiente a los datos del Tipo de Beca<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del tipo de Beca.<br>2. Guardar: No se registran Tipos de Becas con nombre vacio. | | |

Tabla 35: Caso de uso Gestionar Catálogo Tipo de Beca

## H.9. Gestionar Catálogo Estudiante

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU9)** | : | Gestionar Catálogo del Estudiante | | |
| **Definición** | : | Registra la información del Estudiante | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos del Estudiante | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos del Estudiante | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información del Estudiante, Ademas tienen que estar registrado en el Sistema los datos del Recinto, Facultad, Carrera, Periodo Académico y Tipo de Beca | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos del Estudiante | | |
| **Operaciones** | : | 1. Selecciona menu Servicio Social<br>2. Dar Click en el submenu Estudiante<br>3. Dar Click en agregar Estudiante<br>4. LLenar el formulario correspondiente a los datos del Estudiante<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del Estudiante.<br>2. Guardar: No se puede Registrar un estudiante que ya existe.<br>3. Guardar: No se puede Registrar un estudiante que no tenga Nombre o Apellido.<br>4. Guardar: No se registra un estudiante que no tenga carné | | |

Tabla 36: Caso de Uso Gestionar Catálogo Estudiante

## H.10. Gestionar Catálogo de Tarea

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU10)** | : | Gestionar Catálogo de Tarea | | |
| **Definición** | : | Registra la información del Tarea | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos de la Tarea | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos de la Tarea | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información de las Tareas a las Areas, deben estar registrado en el sistema el Tipo Tarea, Beneficiario, Area. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos de la Tarea | | |
| **Operaciones** | : | 1. Selecciona menu Servicio Social<br>2. Dar Click en el submenu Tarea<br>3. Dar Click en agregar Tarea<br>4. LLenar el formulario correspondiente a los datos de la Tarea<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos de la Tarea.<br>2. Guardar: No se registran tarea con nombres vacios. | | |

Tabla 37: Caso de uso Gestionar Catálogo Tarea

### H.11. Gestionar Catálogo Tipo de Tarea

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU11)** | : | Gestionar Catálogo Tipo de Tarea | | |
| **Definición** | : | Registra la información del Tipo de Tarea | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos del Tipo de Tarea | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos del Tipo de Tarea | | |
| **Pre-Condiciones** | : | Oficina de Acción Social analiza información de la tareas | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos del Tipo de Tarea | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Tipo de Tarea<br>3. Dar Click en agregar Tipo de Tarea<br>4. LLenar el formulario correspondiente a los datos del Tipo de Tarea<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del Tipo de Tarea.<br>2. Guardar: No se registran Tipo de Tarea que no tengan descripción. | | |

Tabla 38: Caso de Uso Gestionar Catálogo Tipo Tarea

## H.12. Gestionar Catálogo de Área

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU12)** | : | Gestionar Catálogo Área | | |
| **Definición** | : | Registra la información del Área | | |
| **Prioridad** | : | Vital | Importante | Conveniente |
| **Urgencia** | : | Inmediata | Necesario | Puede Esperar |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social | | Encargado de registrar los datos del Área | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos del Área | | |
| **Pre-Condiciones** | : | Oficina de Acción Social Solicita información de las Áreas | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos del Área | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Área<br>3. Dar Click en agregar Área<br>4. LLenar el formulario correspondiente a los datos del Área<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del Área.<br>2. Guardar: No se Registran Areas con Nombres Vacios | | |

Tabla 39: Caso de Uso Gestionar Catálogo Área

## H.13. Gestionar Catálogo de Beneficiario

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU13)** | : | Gestionar Catálogo de Beneficiario | | |
| **Definición** | : | Registra la información del Beneficiario | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos del Beneficiario | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos del Beneficiario | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información del Beneficiario. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos del Beneficiario | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Beneficiario<br>3. Dar Click en agregar Beneficiario<br>4. LLenar el formulario correspondiente a los datos del Beneficiario<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del Beneficiario.<br>2. Guardar: No se Registran Beneficiarios con nombres vacios. | | |

Tabla 40: Caso de uso Gestionar Catálogo Beneficiario

### H.14. Gestionar Catálogo Evaluación

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU14)** | : | Gestionar Catálogo de Evaluación | | |
| **Definición** | : | Registra la información de la Evaluación | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | **Definición** | | | |
| Responsable Oficina Acción Social | Encargado de registrar los datos de la Evaluación | | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos de la Evaluación | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información de las Evaluaciones | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos de la Evaluación | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Evaluación<br>3. Dar Click en agregar Evaluación<br>4. LLenar el formulario correspondiente a los datos de la Evaluación<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos de la Evaluación.<br>2. Guardar: No se registran evaluaciones con nombres vacios. | | |

Tabla 41: Caso de uso Gestionar Catálogo Evaluación

## H.15. Gestionar Catálogo Aspectos Generales a Evaluar

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU15)** | : | Gestionar Catálogo Aspectos Generales a Evaluar | | |
| **Definición** | : | Registra la información de los Aspectos Generales a Evaluar | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social | | Encargado de registrar los datos de los Aspectos Generales a Evaluar | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de datos de los Aspectos Generales a Evaluar | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información de los Aspectos Generales a Evaluar, Ademas en el Sistema tiene que estar registrada la información correspondiente a la Evaluación y la Tarea. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos de los Aspectos Generales a Evaluar | | |
| **Operaciones** | : | 1. Selecciona menu Catálogo<br>2. Dar Click en el submenu Aspectos Generales a Evaluar<br>3. Dar Click en agregar Aspectos Generales<br>4. LLenar el formulario correspondiente a los datos de los Aspectos Generales<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos de los Aspectos Generales a Evaluar.<br>2. Guardar: No se registran Aspectos generales con nombres vacios. | | |

Tabla 42: Caso de uso Gestionar Catálogo Aspectos Generales

### H.16. Gestionar Asignar Periodo-Carrera

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU16)** | : | Gestionar Asignar Periodo-Carrera | | |
| **Definición** | : | Registra la información del Periodo, Carrera y Tipo de Beca al Estudiante. | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social | | Encargado de Asignar la información del Periodo, Carrera y Tipo de Beca al Estudiante. | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de Asignación del Periodo, Carrera y Tipo de Beca al Estudiante. | | |
| **Pre-Condiciones** | : | Oficina de Acción Social solicita información del Estudiante | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos del Periodo, Carrera y Tipo de Beca del Estudiante. | | |
| **Operaciones** | : | 1. Selecciona menu Servicio Social<br>2. Dar Click en el submenu Asignar Periodo-Carrera<br>3. Dar Click en agregar Periodo-Carrera<br>4. Seleccionar los datos del Periodo, Carrera y Tipo de Beca, seguido buscar el Estudiante.<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | 1. Guardar : Datos incorrectos. La base de datos no ingresa los datos del Teléfono. | | |

Tabla 43: Caso de uso Gestionar Asignación Periodo-Carrera

### H.17. Gestionar Asignaciones de Tareas al Becario

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU17)** | : | Gestionar Asignaciones de Tareas al Becario | | |
| **Definición** | : | Registra las Asignaciones de Tarea al Becario | | |
| **Prioridad** | : | Vital  | Importante  | Conveniente  |
| **Urgencia** | : | Inmediata  | Necesario  | Puede Esperar  |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social | | Encargado de registrar los datos de las Asignaciones de Tarea al Becario | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Registro de Asignaciones de Tarea al Becario | | |
| **Pre-Condiciones** | : | Oficina de Acción Social debe tener registrado al Becario para el periodo activo y la Tarea. | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social puede registrar exitosamente los datos de las Asignaciones de Tarea al Becario | | |
| **Operaciones** | : | 1. Selecciona menu Servicio Social<br>2. Dar Click en el submenu Asignaciones de Tarea<br>3. Dar Click en agregar Asignación<br>4. LLenar el formulario correspondiente a los datos de las Asignación de Tarea<br>5. Dar Click en Guardar | | |
| **Excepciones** | : | Guardar : Datos incorrectos. La base de datos no ingresa los datos de las asignaciones de laTarea del Becario | | |

Tabla 44: Caso de uso Gestionar Asignación de Tarea

## H.18. Emitir Reportes del Programa de Servicio Social

| | | | | |
|---|---|---|---|---|
| **Caso de Uso (CU18)** | : | Emitir Reportes del Programa de Servicio Social | | |
| **Definición** | : | Imprimir Reportes del Programa del Servicio Social | | |
| **Prioridad** | : | Vital ⬭ | Importante ⬬ | Conveniente ⬭ |
| **Urgencia** | : | Inmediata ⬭ | Necesario ⬬ | Puede Esperar ⬭ |
| **ACTORES** | | | | |
| **Nombre** | | **Definición** | | |
| Responsable Oficina Acción Social y Área | | Encargado de Imprimir los datos del Programa del Servicio Social | | |
| **ESCENARIOS** | | | | |
| **Nombre** | : | Imprimir Reportes del Programa del Servicio Social | | |
| **Pre-Condiciones** | : | Oficina de Acción Social y Área Seleccionarán los parámetros para la impresión de reportes | | |
| **Iniciado por** | : | Responsable de Oficina de Acción Social y Área | | |
| **Finalizado por** | : | Responsable de oficina de Acción Social y Área | | |
| **Post-Condiciones** | : | El Responsable de la Oficina de Acción Social y/o Área, pueden imprimir exitosamente los reportes | | |
| **Operaciones** | : | 1. Selecciona menu Reporte<br>2. Dar Click en el submenu Generar Reportes<br>3. Seleccionar Parámetros correspondiente a los datos que se desean imprimir<br>4. Dar Click imprimir | | |
| **Excepciones** | : | Imprimir : Datos incorrectos. El Sistema no generará reporte del Programa del Servicio Social. | | |

Tabla 45: Caso de uso Emitir Reportes

# I. Diagramas de secuencia

## I.1. Diagrama de Secuencia de la Interfaz Catálogo

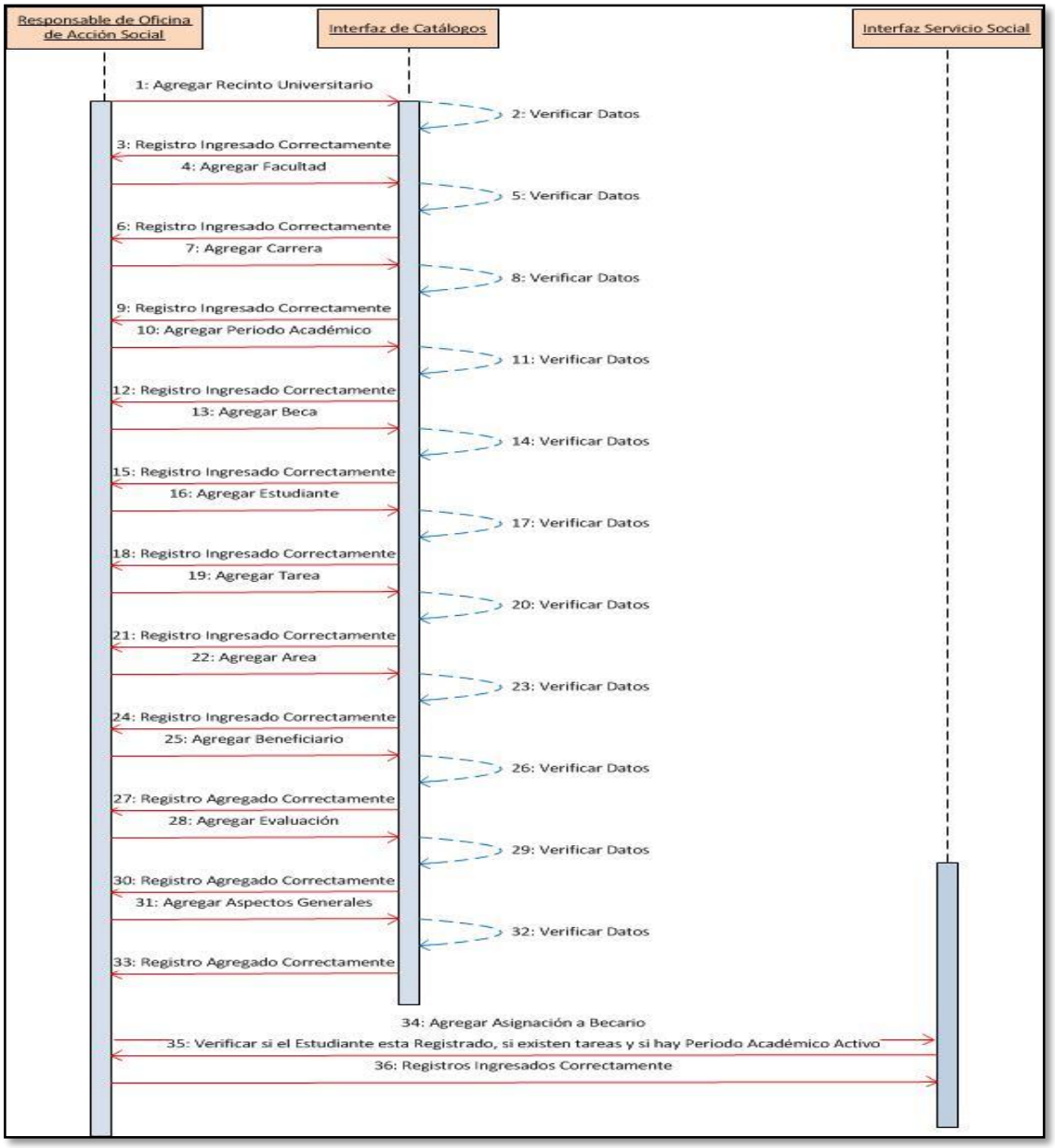



Figura Nº 66: Diagrama de Secuencia de la Interfaz Catálogos

## I.2. Diagrama de Secuencia de la Interfaz Servicio Social

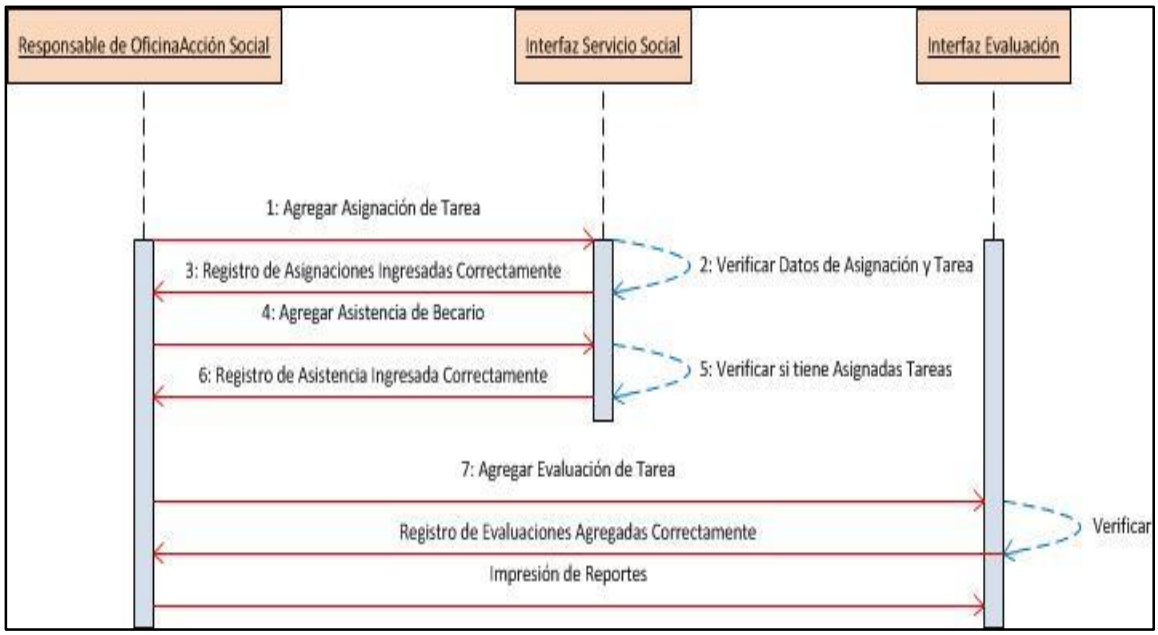



Figura Nº 67: Diagrama de Secuencia Interfaz Servicio Social.

# J. Guia para Instalar la aplicación web SISBECA en ISS

## J.1. Abrir la aplicación IIS (Administrador de Internet Information).

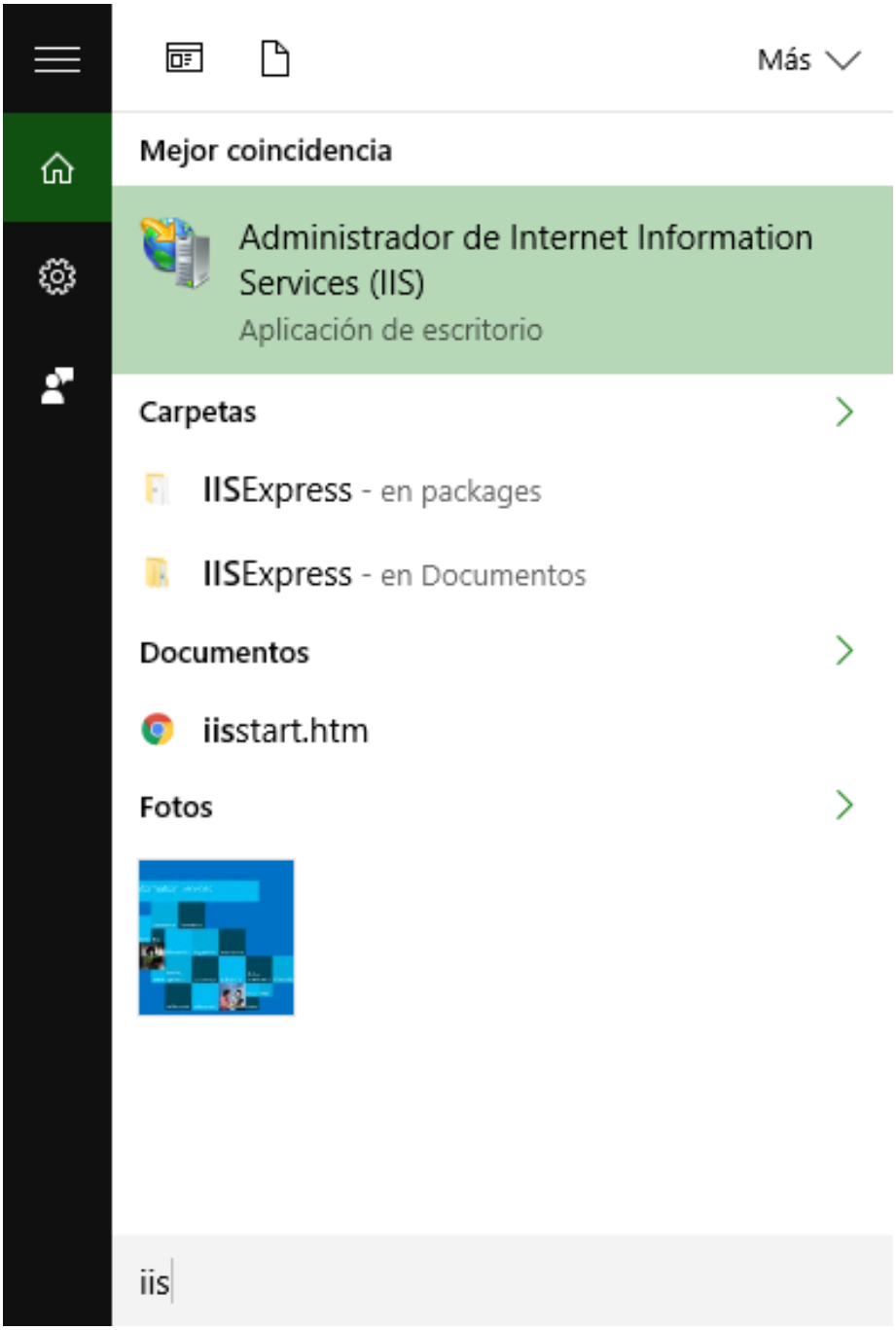



Figura Nº 68: IIS

### J.1.1. En el IIS ubicarse en el panel izquierdo en la carpeta "sitio", dar clic derecho y seleccionar la opción "Agregar Sitio Web".

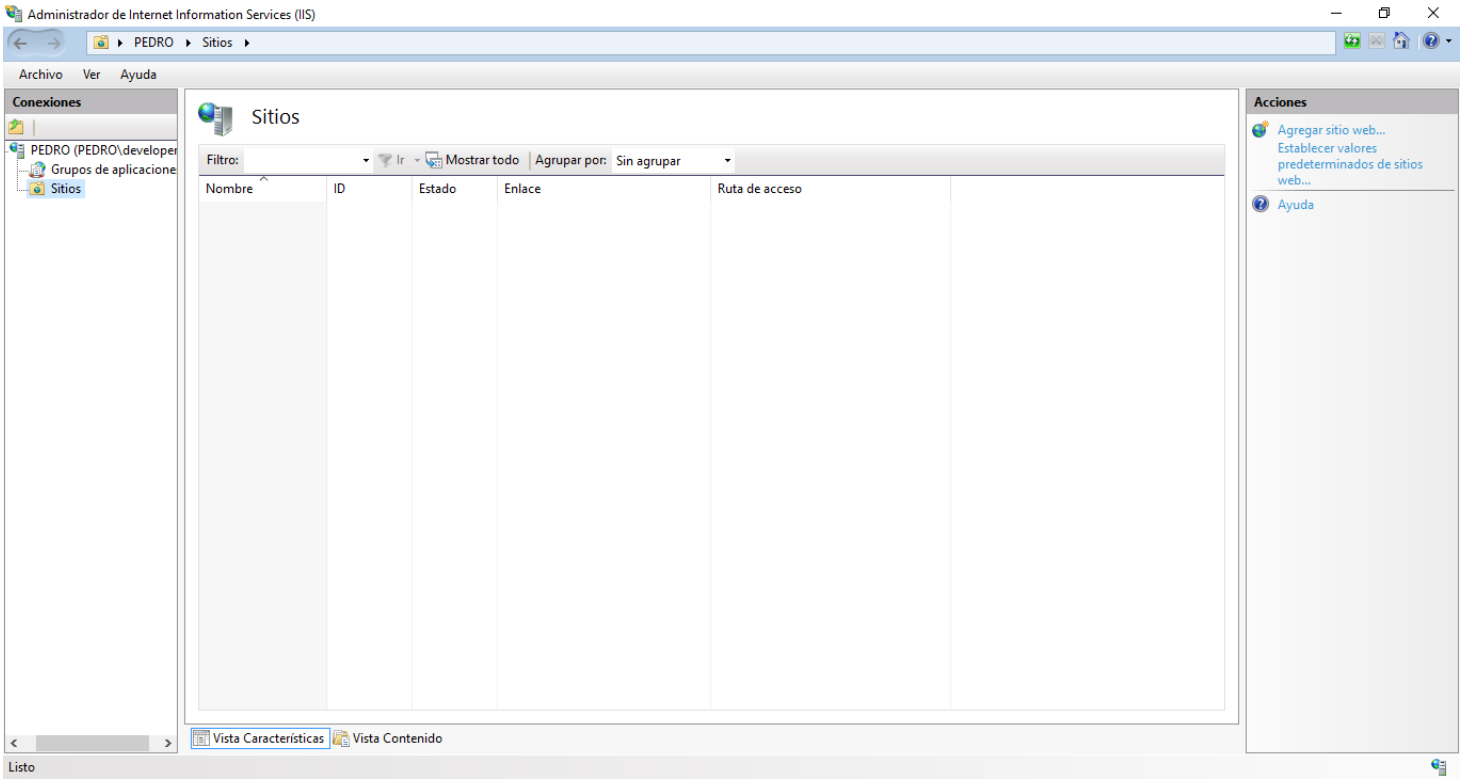



Figura Nº 69: Administrador de IIS

### J.1.2. Al darle clic se desplegara una ventana emergente, el la cual se deberan de realizar los siguientes pasos:

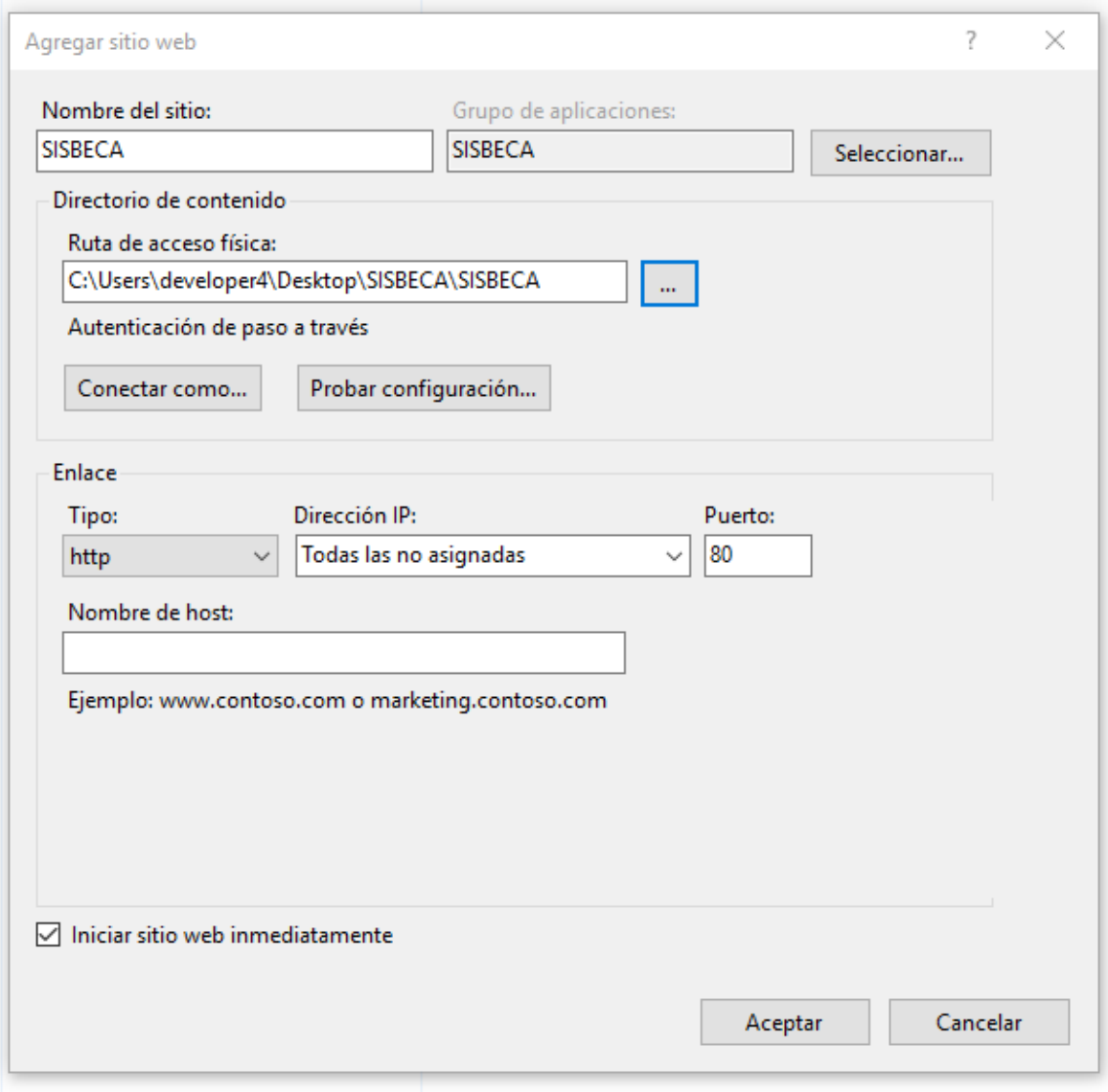



Figura Nº 70: Configuración de Sitio WEB en IIS

- Digitar el nombre con que identificara la aplicación.
- Indicar a traves del navegador, la ruta de la carpeta de la aplicación con el codigo fuente. Tambien solo puede ser la aplicación compilada. Suele ser (C:\inetpub\wwwroot\WebSite).
- Asignar puerto.
- Indicar nombre del Host.
- Dar clic en el boton Aceptar

- Luego de esto para verificar damos clic en el panel derecho del IIS al comando “Examinar” y se ejecutara la aplicación en el navegador por defecto, a continuacion la pantalla:

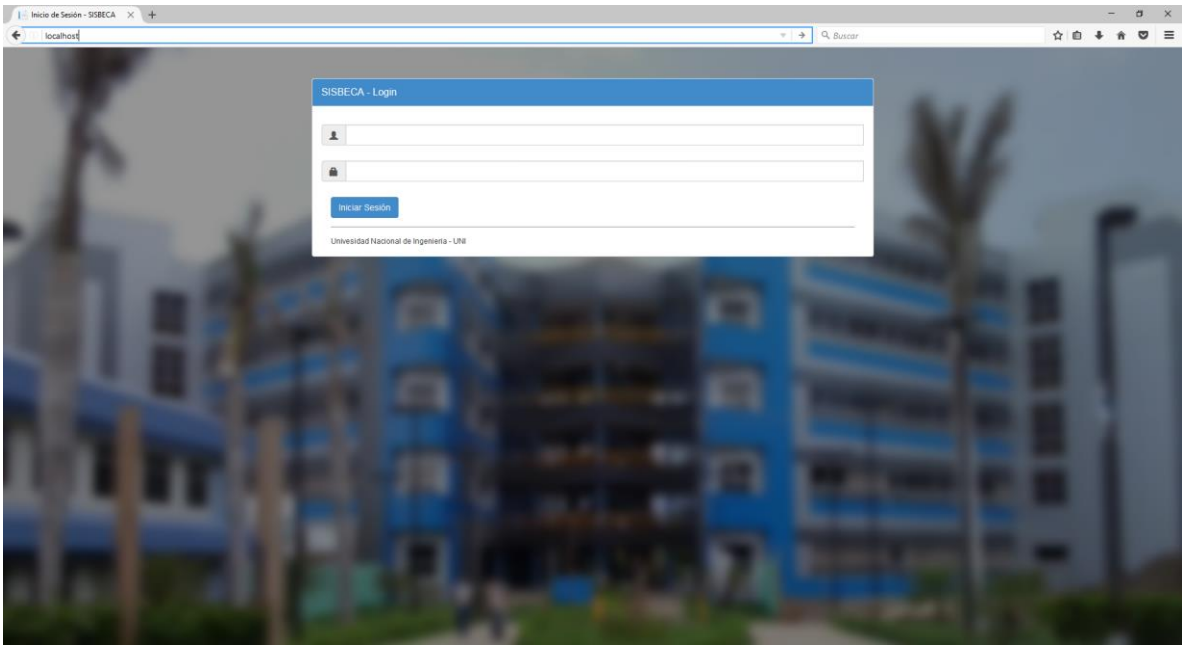



Figura Nº 71: Ejecución de la Aplicación en IIS

## K. Listado de archivos de código fuente

| Número | Documento | Líneas de Código |
|---|---|---|
| 1 | AccountController.cs | 607 |
| 2 | AlumnoController.cs | 149 |
| 3 | AperiodoController.cs | 187 |
| 4 | AreaController.cs | 123 |
| 5 | AsignacionController.cs | 218 |
| 6 | AsistenciaController.cs | 190 |
| 7 | AspectoController.cs | 129 |
| 8 | BeneficiarioController.cs | 124 |
| 9 | CarreraController.cs | 129 |
| 10 | EvaluacionAspectoController.cs | 168 |
| 11 | EvaluacionController.cs | 123 |
| 12 | EvaluaEstudianteController.cs | 225 |
| 13 | FacultadController.cs | 129 |
| 14 | HomeController.cs | 176 |
| 15 | PeriodoAcademicoController.cs | 205 |
| 16 | RecintoController.cs | 125 |
| 17 | ReportesController.cs | 498 |
| 18 | TareaController.cs | 141 |
| 19 | TipoBecaController.cs | 123 |
| 20 | TipoTareaController.cs | 123 |
| 21 | ValidacionesController.cs | 35 |
| 22 | _layout.cshtml | 181 |
| 23 | _layoutLogin.cshtml | 155 |
| 24 | _loginPartial.cshtml | 36 |
| 25 | _MenuLayout.cshtml | 26 |
| 26 | _Grid.cshtml | 65 |
| 27 | boostrap.css | 6265 |
| 28 | sb-admin.css | 291 |
| 29 | datepicker.css | 182 |
| 30 | web.config | 86 |
| 31 | common.js | 152 |
| 32 | gridmvc.js | 724 |
| 33 | boostrap.css | 2114 |

Tabla 46: Listado de Archivos de Código

## L. Código fuente

**Fuente de Controlador: AccountController**

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Transactions;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;
using DotNetOpenAuth.AspNet;
using Microsoft.Web.WebPages.OAuth;
using WebMatrix.WebData;
using Mono.Models;
using System.Data.SqlClient;

//using Mono.Filters;


namespace Mono.Controllers
{
    [Authorize]
    //[InitializeSimpleMembership]
    public class AccountController : Controller

    {

        [Authorize(Roles = "Admin")]
        public ActionResult Index()
        {
            MonoDB db = new MonoDB();
            var usuarios = db.UserProfiles.ToList();

            return View(usuarios);
        }

        [Authorize(Roles = "Admin")]
        public ActionResult PermisoMenu()
        {
            MonoDB db = new MonoDB();
            IEnumerable<UserRol> roles = db.Database.SqlQuery<UserRol> ("SELECT *
FROM dbo.webpages_Roles " +
                                "  WHERE RoleId NOT IN (SELECT upm.RoleId FROM
UserPermisoMenu AS upm )");
            if (roles.Count()<1)
            {
                TempData["sinrol"] = "1";
                return RedirectToAction("RolIndex", "Account");
            }

            List<MenuItem> menu = db.MenuItems.Include(x =>
x.Menus).OrderBy(x=>x.ParentMenu).ToList();
```

```
            List<UserPermisoMenu> lupm = new List<UserPermisoMenu>();

            foreach(MenuItem mi in menu)
            {
                lupm.Add( new UserPermisoMenu { RoleId = 0, MenuItemId = mi.Id,
Acceso = false });
            }

            ViewBag.RoleId = new SelectList(roles, "RoleId", "RoleName");
            ViewBag.menu = menu;


            return View(lupm);
        }

        [Authorize(Roles = "Admin")]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult PermisoMenu(List<UserPermisoMenu> upm, int RoleId)
        {
            if(upm.Count()>0)
            {
                MonoDB db = new MonoDB();

                foreach(UserPermisoMenu pm in upm)
                {
                    //db.UserPermisosMenu.Remove(pm);
                    pm.RoleId = RoleId;
                    db.UserPermisosMenu.Add(pm);
                }
                db.SaveChanges();
            }
            return RedirectToAction("RolIndex", "Account");
        }


        [Authorize(Roles = "Admin")]
        public ActionResult EditPermisoMenu(int id)
        {
            MonoDB db = new MonoDB();

            UserRol rol=db.Database.SqlQuery<UserRol>("SELECT * FROM
dbo.webpages_Roles WHERE RoleId ='"+ id.ToString() +"'").First();
            List<MenuItem> menu = db.MenuItems.Include(x => x.Menus).OrderBy(x =>
x.ParentMenu).ToList();

            List<UserPermisoMenu> lupm =
db.UserPermisosMenu.Where(pm=>pm.RoleId==id).ToList();


            ViewBag.menu = menu;
            ViewBag.rolname = rol.RoleName;
```

```
            return View(lupm);
        }

        [Authorize(Roles = "Admin")]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult EditPermisoMenu(List<UserPermisoMenu> upm)
        {
            if (upm.Count() > 0)
            {
                MonoDB db = new MonoDB();

                foreach (UserPermisoMenu pm in upm)
                {
                    //db.UserPermisosMenu.Remove(pm);
                    db.Entry(pm).State = EntityState.Modified;

                }
                db.SaveChanges();
            }
            return RedirectToAction("RolIndex", "Account");
        }

        [Authorize(Roles = "Admin")]
        public ActionResult UsuarioRol(int id)
        {
            MonoDB db=new MonoDB();

            IEnumerable<UserRol> data = db.Database.SqlQuery<UserRol>
                ("SELECT * FROM dbo.webpages_Roles WHERE dbo.webpages_Roles.RoleId
NOT IN " +
                 " (SELECT wuir.RoleId FROM webpages_UsersInRoles AS wuir WHERE
wuir.UserId=@p0)", new SqlParameter("p0", id));

            UserProfile usuario = db.UserProfiles.Find(id);

            ViewBag.RoleName = new SelectList(data, "RoleName", "RoleName");


            string[] asignado= Roles.GetRolesForUser(usuario.UserName);


            ViewBag.id = usuario.UserId;
            ViewBag.usuario = usuario.UserName;
            ViewBag.nombre = usuario.NombreCompleto;

            ViewBag.asignado = asignado as string[];


            return View();

        }

        [Authorize(Roles = "Admin")]
        [HttpPost]
        [ValidateAntiForgeryToken]
```

```
        public ActionResult UsuarioRol(string usuario, string RoleName)
        {
            if (!Roles.IsUserInRole(usuario, RoleName))
            {
                Roles.AddUserToRole(usuario, RoleName);
                return RedirectToAction("Index", "Account");
            }

            return null;
        }


        //
        // GET: /Account/Login

        [AllowAnonymous]
        public ActionResult Login(string returnUrl)
        {
            //ESTO lo puedo usar para cuando se me olvida la contraseña
            /*
            var token = WebSecurity.GeneratePasswordResetToken("pedro", 1440);
            WebSecurity.ResetPassword(token, "xxx");
            */

            ViewBag.ReturnUrl = returnUrl;
            return View();
        }


        public ActionResult RolIndex()
        {
            string val = TempData["sinrol"] as string;
            if(val=="1")
            {
                ModelState.AddModelError("sinrol", "No existe rol que no tenga
asignado permiso. Intente editar el permiso por Rol!");
            }

            //var roless = Roles.GetAllRoles();
            MonoDB db = new MonoDB();
            IEnumerable<UserRol> roles = db.Database.SqlQuery<UserRol>("SELECT *
FROM dbo.webpages_Roles");

            return View(roles);
        }

           [AllowAnonymous]
        public ActionResult SetRol()
        {
            return View();
        }


           //[AllowAnonymous]
        [Authorize(Roles = "Admin")]
        [HttpPost]
        [ValidateAntiForgeryToken]
```

```
        public ActionResult SetRol(string Rolname)
        {
            var roles = (SimpleRoleProvider)Roles.Provider;


            if (!roles.RoleExists(Rolname))
            {
                roles.CreateRole(Rolname);
            }
            return RedirectToAction("Index", "Home");
        }

        //
        // POST: /Account/Login

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult Login(LoginModel model,  string returnUrl)
        {
            if (ModelState.IsValid && WebSecurity.Login(model.UserName,
model.Password, persistCookie: model.RememberMe))
            {
                return RedirectToLocal(returnUrl);
            }

            // Si llegamos a este punto, es que se ha producido un error y
volvemos a mostrar el formulario
            ModelState.AddModelError("", "El nombre de usuario o la contraseña
especificados son incorrectos.");
            return View(model);
        }

        //
        // POST: /Account/LogOff

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LogOff()
        {
            WebSecurity.Logout();

            return RedirectToAction("Login", "Account");
        }

        //
        // GET: /Account/Register

         [Authorize(Roles = "Admin")]
        [AllowAnonymous]
        public ActionResult Register()
        {
            return View();
        }

        //
        // POST: /Account/Register
```

```
        [HttpPost]
        [Authorize(Roles = "Admin")]
        [ValidateAntiForgeryToken]
        public ActionResult Register(RegisterModel model)
        {
            if (ModelState.IsValid)
            {
                // Intento de registrar al usuario
                try
                {
                    WebSecurity.CreateUserAndAccount(model.UserName,
model.Password, new  { NombreCompleto=model.NombreCompleto });

                    //WebSecurity.Login(model.UserName, model.Password);
                    return RedirectToAction("Index", "Account");
                }
                catch (MembershipCreateUserException e)
                {
                    ModelState.AddModelError("", ErrorCodeToString(e.StatusCode));
                }
            }

            // Si llegamos a este punto, es que se ha producido un error y
volvemos a mostrar el formulario
            return View(model);
        }

        //
        // POST: /Account/Disassociate

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Disassociate(string provider, string providerUserId)
        {
            string ownerAccount = OAuthWebSecurity.GetUserName(provider,
providerUserId);
            ManageMessageId? message = null;

            // Desasociar la cuenta solo si el usuario que ha iniciado sesión es
el propietario
            if (ownerAccount == User.Identity.Name)
            {
                // Usar una transacción para evitar que el usuario elimine su
última credencial de inicio de sesión
                using (var scope = new
TransactionScope(TransactionScopeOption.Required, new TransactionOptions {
IsolationLevel =System.Transactions.IsolationLevel.Serializable }))
                {
                    bool hasLocalAccount =
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
                    if (hasLocalAccount ||
OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name).Count > 1)
                    {
                        OAuthWebSecurity.DeleteAccount(provider, providerUserId);
                        scope.Complete();
                        message = ManageMessageId.RemoveLoginSuccess;
                    }
```

```
                }
            }

            return RedirectToAction("Manage", new { Message = message });
        }

        //
        // GET: /Account/Manage

        public ActionResult Manage(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.ChangePasswordSuccess ? "La contraseña
se ha cambiado."
                : message == ManageMessageId.SetPasswordSuccess ? "Su contraseña
se ha establecido."
                : message == ManageMessageId.RemoveLoginSuccess ? "El inicio de
sesión externo se ha quitado."
                : "";
            ViewBag.HasLocalPassword =
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
            ViewBag.ReturnUrl = Url.Action("Manage");
            return View();
        }

        //
        // POST: /Account/Manage

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Manage(LocalPasswordModel model)
        {
            bool hasLocalAccount =
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
            ViewBag.HasLocalPassword = hasLocalAccount;
            ViewBag.ReturnUrl = Url.Action("Manage");
            if (hasLocalAccount)
            {
                if (ModelState.IsValid)
                {
                    // ChangePassword iniciará una excepción en lugar de devolver
false en determinados escenarios de error.
                    bool changePasswordSucceeded;
                    try
                    {
                        changePasswordSucceeded =
WebSecurity.ChangePassword(User.Identity.Name, model.OldPassword,
model.NewPassword);
                    }
                    catch (Exception)
                    {
                        changePasswordSucceeded = false;
                    }

                    if (changePasswordSucceeded)
                    {
                        return RedirectToAction("Manage", new { Message =
ManageMessageId.ChangePasswordSuccess });
```

```
                    }
                    else
                    {
                        ModelState.AddModelError("", "La contraseña actual es
incorrecta o la nueva contraseña no es válida.");
                    }
                }
            }
            else
            {
                // El usuario no dispone de contraseña local, por lo que debe
quitar todos los errores de validación generados por un
                // campo OldPassword
                ModelState state = ModelState["OldPassword"];
                if (state != null)
                {
                    state.Errors.Clear();
                }

                if (ModelState.IsValid)
                {
                    try
                    {
                        WebSecurity.CreateAccount(User.Identity.Name,
model.NewPassword);
                        return RedirectToAction("Manage", new { Message =
ManageMessageId.SetPasswordSuccess });
                    }
                    catch (Exception)
                    {
                        ModelState.AddModelError("", String.Format("No se puede
crear una cuenta local. Es posible que ya exista una cuenta con el nombre
\"{0}\".", User.Identity.Name));
                    }
                }
            }

            // Si llegamos a este punto, es que se ha producido un error y
volvemos a mostrar el formulario
            return View(model);
        }

        //
        // POST: /Account/ExternalLogin

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
        {
            return new ExternalLoginResult(provider,
Url.Action("ExternalLoginCallback", new { ReturnUrl = returnUrl }));
        }

        //
        // GET: /Account/ExternalLoginCallback

        [AllowAnonymous]
```

```
        public ActionResult ExternalLoginCallback(string returnUrl)
        {
            AuthenticationResult result =
OAuthWebSecurity.VerifyAuthentication(Url.Action("ExternalLoginCallback", new {
ReturnUrl = returnUrl }));
            if (!result.IsSuccessful)
            {
                return RedirectToAction("ExternalLoginFailure");
            }

            if (OAuthWebSecurity.Login(result.Provider, result.ProviderUserId,
createPersistentCookie: false))
            {
                return RedirectToLocal(returnUrl);
            }

            if (User.Identity.IsAuthenticated)
            {
                // Si el usuario actual ha iniciado sesión, agregue la cuenta
nueva
                OAuthWebSecurity.CreateOrUpdateAccount(result.Provider,
result.ProviderUserId, User.Identity.Name);
                return RedirectToLocal(returnUrl);
            }
            else
            {
                // El usuario es nuevo, solicitar nombres de pertenencia deseados
                string loginData =
OAuthWebSecurity.SerializeProviderUserId(result.Provider, result.ProviderUserId);
                ViewBag.ProviderDisplayName =
OAuthWebSecurity.GetOAuthClientData(result.Provider).DisplayName;
                ViewBag.ReturnUrl = returnUrl;
                return View("ExternalLoginConfirmation", new
RegisterExternalLoginModel { UserName = result.UserName, ExternalLoginData =
loginData });
            }
        }

        //
        // POST: /Account/ExternalLoginConfirmation

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLoginConfirmation(RegisterExternalLoginModel
model, string returnUrl)
        {
            string provider = null;
            string providerUserId = null;

            if (User.Identity.IsAuthenticated ||
!OAuthWebSecurity.TryDeserializeProviderUserId(model.ExternalLoginData, out
provider, out providerUserId))
            {
                return RedirectToAction("Manage");
            }

            if (ModelState.IsValid)
```

```
            {
                // Insertar un nuevo usuario en la base de datos
                using (MonoDB db = new MonoDB())
                {
                    UserProfile user = db.UserProfiles.FirstOrDefault(u =>
u.UserName.ToLower() == model.UserName.ToLower());
                    // Comprobar si el usuario ya existe
                    if (user == null)
                    {
                        // Insertar el nombre en la tabla de perfiles
                        db.UserProfiles.Add(new UserProfile { UserName =
model.UserName });
                        db.SaveChanges();

                        OAuthWebSecurity.CreateOrUpdateAccount(provider,
providerUserId, model.UserName);
                        OAuthWebSecurity.Login(provider, providerUserId,
createPersistentCookie: false);

                        return RedirectToLocal(returnUrl);
                    }
                    else
                    {
                        ModelState.AddModelError("UserName", "El nombre de usuario
ya existe. Escriba un nombre de usuario diferente.");
                    }
                }
            }

            ViewBag.ProviderDisplayName =
OAuthWebSecurity.GetOAuthClientData(provider).DisplayName;
            ViewBag.ReturnUrl = returnUrl;
            return View(model);
        }

        //
        // GET: /Account/ExternalLoginFailure

        [AllowAnonymous]
        public ActionResult ExternalLoginFailure()
        {
            return View();
        }

        [AllowAnonymous]
        [ChildActionOnly]
        public ActionResult ExternalLoginsList(string returnUrl)
        {
            ViewBag.ReturnUrl = returnUrl;
            return PartialView("_ExternalLoginsListPartial",
OAuthWebSecurity.RegisteredClientData);
        }

        [ChildActionOnly]
        public ActionResult RemoveExternalLogins()
        {
            ICollection<OAuthAccount> accounts =
OAuthWebSecurity.GetAccountsFromUserName(User.Identity.Name);
```

```
            List<ExternalLogin> externalLogins = new List<ExternalLogin>();
            foreach (OAuthAccount account in accounts)
            {
                AuthenticationClientData clientData =
OAuthWebSecurity.GetOAuthClientData(account.Provider);

                externalLogins.Add(new ExternalLogin
                {
                    Provider = account.Provider,
                    ProviderDisplayName = clientData.DisplayName,
                    ProviderUserId = account.ProviderUserId,
                });
            }

            ViewBag.ShowRemoveButton = externalLogins.Count > 1 ||
OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name));
            return PartialView("_RemoveExternalLoginsPartial", externalLogins);
        }

        #region Aplicaciones auxiliares
        private ActionResult RedirectToLocal(string returnUrl)
        {
            if (Url.IsLocalUrl(returnUrl))
            {
                return Redirect(returnUrl);
            }
            else
            {
                return RedirectToAction("Index", "Home");
            }
        }

        public enum ManageMessageId
        {
            ChangePasswordSuccess,
            SetPasswordSuccess,
            RemoveLoginSuccess,
        }

        internal class ExternalLoginResult : ActionResult
        {
            public ExternalLoginResult(string provider, string returnUrl)
            {
                Provider = provider;
                ReturnUrl = returnUrl;
            }

            public string Provider { get; private set; }
            public string ReturnUrl { get; private set; }

            public override void ExecuteResult(ControllerContext context)
            {
                OAuthWebSecurity.RequestAuthentication(Provider, ReturnUrl);
            }
        }

        private static string ErrorCodeToString(MembershipCreateStatus
createStatus)
```

```
        {
            // Vaya a http://go.microsoft.com/fwlink/?LinkID=177550 para
            // obtener una lista completa de códigos de estado.
            switch (createStatus)
            {
                case MembershipCreateStatus.DuplicateUserName:
                    return "El nombre de usuario ya existe. Escriba un nombre de
usuario diferente.";

                case MembershipCreateStatus.DuplicateEmail:
                    return "Ya existe un nombre de usuario para esa dirección de
correo electrónico. Escriba una dirección de correo electrónico diferente.";

                case MembershipCreateStatus.InvalidPassword:
                    return "La contraseña especificada no es válida. Escriba un
valor de contraseña válido.";

                case MembershipCreateStatus.InvalidEmail:
                    return "La dirección de correo electrónico especificada no es
válida. Compruebe el valor e inténtelo de nuevo.";

                case MembershipCreateStatus.InvalidAnswer:
                    return "La respuesta de recuperación de la contraseña
especificada no es válida. Compruebe el valor e inténtelo de nuevo.";

                case MembershipCreateStatus.InvalidQuestion:
                    return "La pregunta de recuperación de la contraseña
especificada no es válida. Compruebe el valor e inténtelo de nuevo.";

                case MembershipCreateStatus.InvalidUserName:
                    return "El nombre de usuario especificado no es válido.
Compruebe el valor e inténtelo de nuevo.";

                case MembershipCreateStatus.ProviderError:
                    return "El proveedor de autenticación devolvió un error.
Compruebe los datos especificados e inténtelo de nuevo. Si el problema continúa,
póngase en contacto con el administrador del sistema.";

                case MembershipCreateStatus.UserRejected:
                    return "La solicitud de creación de usuario se ha cancelado.
Compruebe los datos especificados e inténtelo de nuevo. Si el problema continúa,
póngase en contacto con el administrador del sistema.";

                default:
                    return "Error desconocido. Compruebe los datos especificados e
inténtelo de nuevo. Si el problema continúa, póngase en contacto con el
administrador del sistema.";
            }
        }
        #endregion
    }
}
```

## Fuente de Controlador de Alumno

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;
using System.Web.UI;

namespace Mono.Controllers

{

    public class AlumnoController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Alumno/

        public ActionResult Index()
        {
            return View(db.Alumnos.ToList());
        }

        //
        // GET: /Alumno/Details/5

        public ActionResult Details(string id = null)
        {
            Alumno alumno = db.Alumnos.Find(id);
            if (alumno == null)
            {
                return HttpNotFound();
            }
            return View(alumno);
        }

        //
        // GET: /Alumno/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Alumno/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Alumno alumno)
        {
            int existe = db.Alumnos.Where(a => a.Carnet == alumno.Carnet).Count();
```

```
            if (existe>0)
            {
                ModelState.AddModelError("repetido", "Ya existe alumno con este
numero de carnet.");
            }

            if (ModelState.IsValid)
            {
                alumno.FechaRegistro = DateTime.Now;
                db.Alumnos.Add(alumno);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(alumno);
        }


        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public JsonResult ValidaCarnet(string Carnet)
        {
            int existe= db.Alumnos.Where(x => x.Carnet == Carnet).Count();

            if (existe > 0)
            {
                return Json("Ya existe un alumno con este numero de carnet.",
JsonRequestBehavior.AllowGet);
            }
            else
            {
                return Json(true, JsonRequestBehavior.AllowGet);
            }
        }

        //
        // GET: /Alumno/Edit/5

        public ActionResult Edit(string id = null)
        {
            Alumno alumno = db.Alumnos.Find(id);
            if (alumno == null)
            {
                return HttpNotFound();
            }
            return View(alumno);
        }

        //
        // POST: /Alumno/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Alumno alumno)
        {
            if (ModelState.IsValid)
            {
                db.Entry(alumno).State = EntityState.Modified;
                db.SaveChanges();
```

```
                return RedirectToAction("Index");
            }
            return View(alumno);
        }

        //
        // GET: /Alumno/Delete/5

        public ActionResult Delete(string id = null)
        {
            Alumno alumno = db.Alumnos.Find(id);
            if (alumno == null)
            {
                return HttpNotFound();
            }
            return View(alumno);
        }

        //
        // POST: /Alumno/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(string id)
        {
            Alumno alumno = db.Alumnos.Find(id);
            db.Alumnos.Remove(alumno);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador Alumno Periodo

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class AperiodoController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Aperiodo/

        public ActionResult Index()
        {
            var alumnosperiodo = db.AlumnosPeriodo.Include(a =>
a.Alumnos).Include(a => a.Periodos).Include(a => a.Carreras).Include(a =>
a.Becas);
            return View(alumnosperiodo.ToList());
        }



        public JsonResult Autocomplete(String term)
        {
            int Tip=0;
            Boolean Convierte= int.TryParse( term.Substring(0,1), out Tip);


            if (Convierte)
            {
                 var modelnomb = db.Alumnos
                    .Where(p => p.Carnet.Contains(term))
                    .Take(10)
                    .Select(p => new
                    {
                        label = p.Carnet + " >> " + p.Nombres + " " + p.Apellidos
                    });
                 JsonResult result = Json(modelnomb);
                 result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
                 return result;
            }
            else
            {
                var modelcarnet = db.Alumnos
                    .Where(p => p.Nombres.Contains(term))
                    .Take(10)
                    .Select(p => new
                    {
                        label =p.Carnet + " >> " + p.Nombres + " " + p.Apellidos
```

```
                    });
                JsonResult result = Json(modelcarnet);
                result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
                return result;
            }

            //result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
            //return result;
        }

        //
        // GET: /Aperiodo/Details/5
        //Se descontinua ta muy complejo implementar esto en un modal con un grid
y ajax x la paginacion
        //public ActionResult GetAlumno(int id=0)
        //{
        //    var alu = db.Alumnos;
        //    return View(alu.ToList());
        //}

        public ActionResult Details(int id = 0)
        {
            AlumnoPeriodo alumnoperiodo = db.AlumnosPeriodo.Find(id);
            if (alumnoperiodo == null)
            {
                return HttpNotFound();
            }
            return View(alumnoperiodo);
        }

        //
        // GET: /Aperiodo/Create

        public ActionResult Create()
        {
            ViewBag.Carnet = new SelectList(db.Alumnos, "Carnet", "Nombres");
            ViewBag.IdPeriodoAcademico = new SelectList(db.Periodo_Academicos,
"IdPeriodoAcademico", "Nombre_Unico");
            ViewBag.IdCarrera = new SelectList(db.Carreras, "IdCarrera",
"Nombre");
            ViewBag.IdTipoBeca = new SelectList(db.TiposBecas, "IdTipoBeca",
"Nombre");
            return View();
        }

        //
        // POST: /Aperiodo/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(AlumnoPeriodo alumnoperiodo)
        {
            //Console.Write(Request('Carnet'));
            if (ModelState.IsValid)
            {
                alumnoperiodo.FechaCreacion = DateTime.Now;
                db.AlumnosPeriodo.Add(alumnoperiodo);
                db.SaveChanges();
```

```
                return RedirectToAction("Index");
            }

            ViewBag.Carnet = new SelectList(db.Alumnos, "Carnet", "Nombres",
alumnoperiodo.Carnet);
            ViewBag.IdPeriodoAcademico = new SelectList(db.Periodo_Academicos,
"IdPeriodoAcademico", "Ciclo", alumnoperiodo.IdPeriodoAcademico);
            ViewBag.IdCarrera = new SelectList(db.Carreras, "IdCarrera", "Nombre",
alumnoperiodo.IdCarrera);
            ViewBag.IdTipoBeca = new SelectList(db.TiposBecas, "IdTipoBeca",
"Nombre", alumnoperiodo.IdTipoBeca);
            return View(alumnoperiodo);
        }

        //
        // GET: /Aperiodo/Edit/5

        public ActionResult Edit(int id = 0)
        {
            AlumnoPeriodo alumnoperiodo = db.AlumnosPeriodo.Find(id);
            if (alumnoperiodo == null)
            {
                return HttpNotFound();
            }
            ViewBag.Carnet = new SelectList(db.Alumnos, "Carnet", "Nombres",
alumnoperiodo.Carnet);
            ViewBag.IdPeriodoAcademico = new SelectList(db.Periodo_Academicos,
"IdPeriodoAcademico", "Ciclo", alumnoperiodo.IdPeriodoAcademico);
            ViewBag.IdCarrera = new SelectList(db.Carreras, "IdCarrera", "Nombre",
alumnoperiodo.IdCarrera);
            ViewBag.IdTipoBeca = new SelectList(db.TiposBecas, "IdTipoBeca",
"Nombre", alumnoperiodo.IdTipoBeca);
            return View(alumnoperiodo);
        }

        //
        // POST: /Aperiodo/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(AlumnoPeriodo alumnoperiodo)
        {
            if (ModelState.IsValid)
            {
                db.Entry(alumnoperiodo).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.Carnet = new SelectList(db.Alumnos, "Carnet", "Nombres",
alumnoperiodo.Carnet);
            ViewBag.IdPeriodoAcademico = new SelectList(db.Periodo_Academicos,
"IdPeriodoAcademico", "Ciclo", alumnoperiodo.IdPeriodoAcademico);
            ViewBag.IdCarrera = new SelectList(db.Carreras, "IdCarrera", "Nombre",
alumnoperiodo.IdCarrera);
            ViewBag.IdTipoBeca = new SelectList(db.TiposBecas, "IdTipoBeca",
"Nombre", alumnoperiodo.IdTipoBeca);
            return View(alumnoperiodo);
        }
```

```
        //
        // GET: /Aperiodo/Delete/5

        public ActionResult Delete(int id = 0)
        {
            AlumnoPeriodo alumnoperiodo = db.AlumnosPeriodo.Find(id);
            if (alumnoperiodo == null)
            {
                return HttpNotFound();
            }
            return View(alumnoperiodo);
        }

        //
        // POST: /Aperiodo/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            AlumnoPeriodo alumnoperiodo = db.AlumnosPeriodo.Find(id);
            db.AlumnosPeriodo.Remove(alumnoperiodo);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Catalogo Área

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class AreaController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Area/

        public ActionResult Index()
        {
            return View(db.Areas.ToList());
        }

        //
        // GET: /Area/Details/5

        public ActionResult Details(int id = 0)
        {
            Area area = db.Areas.Find(id);
            if (area == null)
            {
                return HttpNotFound();
            }
            return View(area);
        }

        //
        // GET: /Area/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Area/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Area area)
        {
            if (ModelState.IsValid)
            {
                db.Areas.Add(area);
                db.SaveChanges();
```

```
                return RedirectToAction("Index");
            }

            return View(area);
        }

        //
        // GET: /Area/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Area area = db.Areas.Find(id);
            if (area == null)
            {
                return HttpNotFound();
            }
            return View(area);
        }

        //
        // POST: /Area/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Area area)
        {
            if (ModelState.IsValid)
            {
                db.Entry(area).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(area);
        }

        //
        // GET: /Area/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Area area = db.Areas.Find(id);
            if (area == null)
            {
                return HttpNotFound();
            }
            return View(area);
        }

        //
        // POST: /Area/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Area area = db.Areas.Find(id);
            db.Areas.Remove(area);
            db.SaveChanges();
```

```
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Asignaciones de Tarea

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class AsignacionController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Asignacion/

        public ActionResult Index()
        {
            //var asignaciones =
db.Asignaciones.Where(s=>s.AlumnosPeriodos.Periodos.Estado==true).
            //                    Include(a => a.AlumnosPeriodos).Include(a =>
a.Tareas).Include(a => a.Evaluaciones);
            var asignaciones = db.Asignaciones.Include(a =>
a.AlumnosPeriodos).Include(a => a.Tareas).Include(a => a.Evaluaciones);
            return View(asignaciones.ToList());
        }


        public JsonResult Autocomplete(String term)
        {
            int Tip = 0;
            Boolean Convierte = int.TryParse(term.Substring(0, 1), out Tip);


            if (Convierte)
            {
                var modelnomb = db.AlumnosPeriodo.Where(a =>
a.Carnet.Contains(term) && a.Periodos.Estado == true)
                    .Take(10)
                    .Select(p => new
                    {
                        label = p.Carnet + " » " + p.Alumnos.Nombres + " " +
p.Alumnos.Apellidos ,
                        id = p.IdPeriodoAlumno
                        //+ "  " + p.IdPeriodoAlumno.ToString()
                    });

                JsonResult result = Json(modelnomb);
                result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
                return result;
            }
            else
            {
```

```
                var modelcarnet = db.AlumnosPeriodo.Where(a =>
a.Alumnos.Nombres.Contains(term) && a.Periodos.Estado == true)
                    .Take(10)
                    .Select(p => new
                    {
                        label = p.Carnet + " » " + p.Alumnos.Nombres + " " +
p.Alumnos.Apellidos ,
                        id = p.IdPeriodoAlumno
                        //+ "  " + p.IdPeriodoAlumno.ToString()
                    });


                JsonResult result = Json(modelcarnet);
                result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
                return result;
            }

            //result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
            //return result;
        }

        //
        // GET: /Asignacion/Details/5

        public ActionResult Details(int id = 0)
        {
            Asignacion asignacion = db.Asignaciones.Find(id);
            if (asignacion == null)
            {
                return HttpNotFound();
            }
            return View(asignacion);
        }

        //
        // GET: /Asignacion/Create

        public ActionResult Create()
        {
            ViewBag.IdAlumnoPeriodo = new SelectList(db.AlumnosPeriodo,
"IdPeriodoAlumno", "IdPeriodoAlumno");
            ViewBag.TareaId = new SelectList(db.Tareas, "IdTarea", "Nombre");
            ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones, "IdEvaluacion",
"Descripcion");
            return View();
        }

        //
        // POST: /Asignacion/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Asignacion asignacion)
        {


            if (ModelState.IsValid)
```

```
            {
                asignacion.Estado = true;
                asignacion.IdEvaluacion = 6;

                db.Asignaciones.Add(asignacion);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdAlumnoPeriodo = new
SelectList(db.AlumnosPeriodo.Where(a=>a.Periodos.Estado==true));
            ViewBag.TareaId = new SelectList(db.Tareas, "IdTarea", "Nombre",
asignacion.TareaId);
            ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones, "IdEvaluacion",
"Descripcion", asignacion.IdEvaluacion);
            return View(asignacion);
        }

        //
        // GET: /Asignacion/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Asignacion asignacion = db.Asignaciones.Find(id);
            if (asignacion == null)
            {
                return HttpNotFound();
            }
            ViewBag.IdAlumnoPeriodo = new SelectList(db.AlumnosPeriodo,
"IdPeriodoAlumno", "Carnet", asignacion.IdAlumnoPeriodo);
            ViewBag.TareaId = new SelectList(db.Tareas, "IdTarea", "Nombre",
asignacion.TareaId);
            ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones, "IdEvaluacion",
"Descripcion", asignacion.IdEvaluacion);
            return View(asignacion);
        }

        //
        // POST: /Asignacion/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Asignacion asignacion)
        {
            if (ModelState.IsValid)
            {
                db.Entry(asignacion).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.IdAlumnoPeriodo = new SelectList(db.AlumnosPeriodo,
"IdPeriodoAlumno", "Carnet", asignacion.IdAlumnoPeriodo);
            ViewBag.TareaId = new SelectList(db.Tareas, "IdTarea", "Nombre",
asignacion.TareaId);
            ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones, "IdEvaluacion",
"Descripcion", asignacion.IdEvaluacion);
            return View(asignacion);
        }
```

```
        //
        // GET: /Asignacion/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Asignacion asignacion = db.Asignaciones.Find(id);
            if (asignacion == null)
            {
                return HttpNotFound();
            }
            return View(asignacion);
        }

        //
        // POST: /Asignacion/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Asignacion asignacion = db.Asignaciones.Find(id);
            db.Asignaciones.Remove(asignacion);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Asistencias

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Mono.Models
{
    public class AsistenciaController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Asistencia/

        public ActionResult Index()
        {
            var asistencias = db.Asistencias.Include(a => a.Asignaciones);
            return View(asistencias.ToList());
        }


        public JsonResult Autocomplete(String term)
        {
            int Tip = 0;
            Boolean Convierte = int.TryParse(term.Substring(0, 1), out Tip);


            if (Convierte)
            {


                var modelnomb =  db.Asignaciones.Where(
                    a=> a.Estado==true &&
                        a.Estado==true &&
                        a.Tareas.Estado==true &&
                        a.AlumnosPeriodos.Periodos.Estado==true &&
                        a.AlumnosPeriodos.Alumnos.Carnet.Contains(term) )
                    .Take(10)
                    .Select(p => new
                    {
                        label = p.AlumnosPeriodos.Alumnos.Carnet + " >> " +
                        p.AlumnosPeriodos.Alumnos.Nombres + " " +
p.AlumnosPeriodos.Alumnos.Apellidos +
                         " >> " + p.Tareas.Nombre,
                        id = p.IdAsignacion
                    });

                JsonResult result = Json(modelnomb);
                result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
                return result;
            }
            else
```

```
            {
                var modelcarnet = db.Asignaciones.Where(
                    a => a.Estado == true &&
                        a.Estado == true &&
                        a.Tareas.Estado == true &&
                        a.AlumnosPeriodos.Periodos.Estado == true &&
                        a.AlumnosPeriodos.Alumnos.Nombres.Contains(term))
                    .Take(10)
                    .Select(p => new
                    {
                        label = p.AlumnosPeriodos.Alumnos.Carnet + " >> " +
                        p.AlumnosPeriodos.Alumnos.Nombres + " " +
p.AlumnosPeriodos.Alumnos.Apellidos +
                         " >> " + p.Tareas.Nombre,
                        id = p.IdAsignacion
                    });
                JsonResult result = Json(modelcarnet);
                result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
                return result;
            }

            //result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
            //return result;
        }

        //
        // GET: /Asistencia/Details/5

        public ActionResult Details(int id = 0)
        {
            Asistencia asistencia = db.Asistencias.Find(id);
            if (asistencia == null)
            {
                return HttpNotFound();
            }
            return View(asistencia);
        }


        //
        // GET: /Asistencia/Create

        public ActionResult Create()
        {
            ViewBag.AsignacionId = new
SelectList(db.Asignaciones.Where(a=>a.AlumnosPeriodos.Periodos.Estado==true),
"IdAsignacion", "IdAsignacion");
            ViewBag.EvaluacionId = new SelectList(db.Evaluaciones, "IdEvaluacion",
"Descripcion");
            return View();
        }

        //
        // POST: /Asistencia/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Asistencia asistencia)
```

```
        {
            if (ModelState.IsValid)
            {
                asistencia.Estado = true;
                asistencia.FechaRegistro = DateTime.Now;
                //asistencia.EvaluacionId = 6;
                db.Asistencias.Add(asistencia);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.AsignacionId = new SelectList(db.Asignaciones, "IdAsignacion",
"IdAsignacion", asistencia.AsignacionId);
            //ViewBag.EvaluacionId = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", asistencia.EvaluacionId);
            return View(asistencia);
        }

        //
        // GET: /Asistencia/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Asistencia asistencia = db.Asistencias.Find(id);
            if (asistencia == null)
            {
                return HttpNotFound();
            }
            ViewBag.AsignacionId = new SelectList(db.Asignaciones, "IdAsignacion",
"IdAsignacion", asistencia.AsignacionId);
            //ViewBag.EvaluacionId = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", asistencia.EvaluacionId);
            return View(asistencia);
        }

        //
        // POST: /Asistencia/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Asistencia asistencia)
        {
            if (ModelState.IsValid)
            {
                db.Entry(asistencia).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.AsignacionId = new SelectList(db.Asignaciones, "IdAsignacion",
"IdAsignacion", asistencia.AsignacionId);
            //ViewBag.EvaluacionId = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", asistencia.EvaluacionId);
            return View(asistencia);
        }

        //
        // GET: /Asistencia/Delete/5
```

```
        public ActionResult Delete(int id = 0)
        {
            Asistencia asistencia = db.Asistencias.Find(id);
            if (asistencia == null)
            {
                return HttpNotFound();
            }
            return View(asistencia);
        }

        //
        // POST: /Asistencia/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Asistencia asistencia = db.Asistencias.Find(id);
            db.Asistencias.Remove(asistencia);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Catalogo de Aspectos a Evaluar

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class AspectoController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Aspecto/

        public ActionResult Index()
        {
            var aspectos = db.Aspectos.Include(a => a.TiposAspectos);
            return View(aspectos.ToList());
        }

        //
        // GET: /Aspecto/Details/5

        public ActionResult Details(int id = 0)
        {
            AspectosEvaluar aspectosevaluar = db.Aspectos.Find(id);
            if (aspectosevaluar == null)
            {
                return HttpNotFound();
            }
            return View(aspectosevaluar);
        }

        //
        // GET: /Aspecto/Create

        public ActionResult Create()
        {
            ViewBag.Tipo = new SelectList(db.TiposAspectos, "IdTipoAspecto",
"Tipo");
            return View();
        }

        //
        // POST: /Aspecto/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(AspectosEvaluar aspectosevaluar)
        {
            if (ModelState.IsValid)
```

```
            {
                aspectosevaluar.Estado = true;
                db.Aspectos.Add(aspectosevaluar);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.Tipo = new SelectList(db.TiposAspectos, "IdTipoAspecto",
"Tipo", aspectosevaluar.Tipo);
            return View(aspectosevaluar);
        }

        //
        // GET: /Aspecto/Edit/5

        public ActionResult Edit(int id = 0)
        {
            AspectosEvaluar aspectosevaluar = db.Aspectos.Find(id);
            if (aspectosevaluar == null)
            {
                return HttpNotFound();
            }
            ViewBag.Tipo = new SelectList(db.TiposAspectos, "IdTipoAspecto",
"Tipo", aspectosevaluar.Tipo);
            return View(aspectosevaluar);
        }

        //
        // POST: /Aspecto/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(AspectosEvaluar aspectosevaluar)
        {
            if (ModelState.IsValid)
            {
                db.Entry(aspectosevaluar).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.Tipo = new SelectList(db.TiposAspectos, "IdTipoAspecto",
"Tipo", aspectosevaluar.Tipo);
            return View(aspectosevaluar);
        }

        //
        // GET: /Aspecto/Delete/5

        public ActionResult Delete(int id = 0)
        {
            AspectosEvaluar aspectosevaluar = db.Aspectos.Find(id);
            if (aspectosevaluar == null)
            {
                return HttpNotFound();
            }
            return View(aspectosevaluar);
        }
```

```
        //
        // POST: /Aspecto/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            AspectosEvaluar aspectosevaluar = db.Aspectos.Find(id);
            db.Aspectos.Remove(aspectosevaluar);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador Catalogo de Beneficiarios

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class BeneficiarioController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Beneficiario/

        public ActionResult Index()
        {
            return View(db.Beneficiarios.ToList());
        }

        //
        // GET: /Beneficiario/Details/5

        public ActionResult Details(int id = 0)
        {
            Beneficiario beneficiario = db.Beneficiarios.Find(id);
            if (beneficiario == null)
            {
                return HttpNotFound();
            }
            return View(beneficiario);
        }

        //
        // GET: /Beneficiario/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Beneficiario/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Beneficiario beneficiario)
        {
            if (ModelState.IsValid)
            {
```

```
                beneficiario.Activo = true;
                db.Beneficiarios.Add(beneficiario);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(beneficiario);
        }

        //
        // GET: /Beneficiario/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Beneficiario beneficiario = db.Beneficiarios.Find(id);
            if (beneficiario == null)
            {
                return HttpNotFound();
            }
            return View(beneficiario);
        }

        //
        // POST: /Beneficiario/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Beneficiario beneficiario)
        {
            if (ModelState.IsValid)
            {
                db.Entry(beneficiario).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(beneficiario);
        }

        //
        // GET: /Beneficiario/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Beneficiario beneficiario = db.Beneficiarios.Find(id);
            if (beneficiario == null)
            {
                return HttpNotFound();
            }
            return View(beneficiario);
        }

        //
        // POST: /Beneficiario/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
```

```
            Beneficiario beneficiario = db.Beneficiarios.Find(id);
            db.Beneficiarios.Remove(beneficiario);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador Catálogo de Carreras

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class CarreraController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Carrera/

        public ActionResult Index()
        {
            var carreras = db.Carreras.Include(c => c.Facultades);
            return View(carreras.ToList());
        }

        //
        // GET: /Carrera/Details/5

        public ActionResult Details(int id = 0)
        {
            Carrera carrera = db.Carreras.Find(id);
            if (carrera == null)
            {
                return HttpNotFound();
            }
            return View(carrera);
        }

        //
        // GET: /Carrera/Create

        public ActionResult Create()
        {
            ViewBag.IdFacultad = new SelectList(db.Facultades, "IdFaculta",
"Nombre");
            return View();
        }

        //
        // POST: /Carrera/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Carrera carrera)
        {
            if (ModelState.IsValid)
```

```
            {
                carrera.Estado = true;
                db.Carreras.Add(carrera);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdFacultad = new SelectList(db.Facultades, "IdFaculta",
"Nombre", carrera.IdFacultad);
            return View(carrera);
        }

        //
        // GET: /Carrera/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Carrera carrera = db.Carreras.Find(id);
            if (carrera == null)
            {
                return HttpNotFound();
            }
            ViewBag.IdFacultad = new SelectList(db.Facultades, "IdFaculta",
"Nombre", carrera.IdFacultad);
            return View(carrera);
        }

        //
        // POST: /Carrera/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Carrera carrera)
        {
            if (ModelState.IsValid)
            {
                db.Entry(carrera).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.IdFacultad = new SelectList(db.Facultades, "IdFaculta",
"Nombre", carrera.IdFacultad);
            return View(carrera);
        }

        //
        // GET: /Carrera/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Carrera carrera = db.Carreras.Find(id);
            if (carrera == null)
            {
                return HttpNotFound();
            }
            return View(carrera);
        }
```

```
        //
        // POST: /Carrera/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Carrera carrera = db.Carreras.Find(id);
            db.Carreras.Remove(carrera);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Proceso de Evaluación de Estudiante

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class EvaluacionAspectoController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /EvaluacionAspecto/

        public ActionResult Index()
        {
            var evaluandoaspectos = db.EvaluandoAspectos.Include(e =>
e.AspectosaEvaluar).Include(e => e.Evaluaciones).Include(e =>
e.EvaluacionEstudiantes);
            return View(evaluandoaspectos.ToList());
        }

        //
        // GET: /EvaluacionAspecto/Details/5

        public ActionResult Details(int id = 0)
        {
            EvaluacionAspectos evaluacionaspectos = db.EvaluandoAspectos.Find(id);
            if (evaluacionaspectos == null)
            {
                return HttpNotFound();
            }
            return View(evaluacionaspectos);
        }

        //
        // GET: /EvaluacionAspecto/Create

        public ActionResult Create()
        {
            ViewBag.IdAspecto = new SelectList(db.Aspectos, "IdAspecto",
"Aspecto");
            ViewBag.ResultadoAspecto = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion");
            ViewBag.IdEvaluacionEstudiante = new
SelectList(db.EvaluandoEstudiante, "Id", "Observaciones");
            return View();
        }

        //
        // POST: /EvaluacionAspecto/Create
```

```
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(EvaluacionAspectos evaluacionaspectos)
        {
            if (ModelState.IsValid)
            {
                db.EvaluandoAspectos.Add(evaluacionaspectos);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdAspecto = new SelectList(db.Aspectos, "IdAspecto",
"Aspecto", evaluacionaspectos.IdAspecto);
            ViewBag.ResultadoAspecto = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", evaluacionaspectos.ResultadoAspecto);
            ViewBag.IdEvaluacionEstudiante = new
SelectList(db.EvaluandoEstudiante, "Id", "Observaciones",
evaluacionaspectos.IdEvaluacionEstudiante);
            return View(evaluacionaspectos);
        }

        //
        // GET: /EvaluacionAspecto/Edit/5

        public ActionResult Edit(int id = 0)
        {
            if (id > 0)
            {

                List<EvaluacionEstudiante> eva = db.EvaluandoEstudiante.Where(e =>
e.Id == id)
                    .Include(a => a.Asignaciones.AlumnosPeriodos.Alumnos)
                    .Include(b => b.Asignaciones.Tareas.Beneficiarios)
                    .Include(r => r.Evaluaciones).ToList();


                var evaluacionaspectos = db.EvaluandoAspectos
                    .Where(x => x.EvaluacionEstudiantes.Id == id)
                    .Include(x=>x.AspectosaEvaluar.TiposAspectos);

                List<string> Tipo = new List<string>();
                foreach (var eval in evaluacionaspectos)
                {
                    Tipo.Add(eval.AspectosaEvaluar.TiposAspectos.Tipo);
                }


                if (evaluacionaspectos.Count()<1)
                {
                    return HttpNotFound();
                }


                ViewBag.eva = eva;
                ViewBag.ResultadoAspecto = db.Evaluaciones.ToList();
                ViewBag.Tipo = Tipo;
                ViewBag.Aspectos = db.Aspectos.ToList();
```

```
                ViewBag.ResultadoAspecto = db.Evaluaciones.ToList();

                return View(evaluacionaspectos.ToList());


            }
            return HttpNotFound();
        }

        //
        // POST: /EvaluacionAspecto/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(List<EvaluacionAspectos> evaluacionaspectos)
        {
            if (ModelState.IsValid)
            {
                foreach (EvaluacionAspectos eva in evaluacionaspectos)
                {
                    db.Entry(eva).State = EntityState.Modified;
                    db.SaveChanges();
                }

                return RedirectToAction("Index","EvaluaEstudiante");
            }
            ViewBag.IdAspecto = new SelectList(db.Aspectos, "IdAspecto",
"Aspecto", evaluacionaspectos[0].IdAspecto);
            ViewBag.ResultadoAspecto = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", evaluacionaspectos[0].ResultadoAspecto);
            ViewBag.IdEvaluacionEstudiante = new
SelectList(db.EvaluandoEstudiante, "Id", "Observaciones",
evaluacionaspectos[0].IdEvaluacionEstudiante);
            return View(evaluacionaspectos);
        }

        //
        // GET: /EvaluacionAspecto/Delete/5

        public ActionResult Delete(int id = 0)
        {
            EvaluacionAspectos evaluacionaspectos = db.EvaluandoAspectos.Find(id);
            if (evaluacionaspectos == null)
            {
                return HttpNotFound();
            }
            return View(evaluacionaspectos);
        }

        //
        // POST: /EvaluacionAspecto/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            EvaluacionAspectos evaluacionaspectos = db.EvaluandoAspectos.Find(id);
            db.EvaluandoAspectos.Remove(evaluacionaspectos);
```

```
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Proceso de Evaluación General del Estudiante

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class EvaluacionController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Evaluacion/

        public ActionResult Index()
        {
            return View(db.Evaluaciones.ToList());
        }

        //
        // GET: /Evaluacion/Details/5

        public ActionResult Details(int id = 0)
        {
            Evaluacion evaluacion = db.Evaluaciones.Find(id);
            if (evaluacion == null)
            {
                return HttpNotFound();
            }
            return View(evaluacion);
        }

        //
        // GET: /Evaluacion/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Evaluacion/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Evaluacion evaluacion)
        {
            if (ModelState.IsValid)
            {
                db.Evaluaciones.Add(evaluacion);
                db.SaveChanges();
```

```
                return RedirectToAction("Index");
            }

            return View(evaluacion);
        }

        //
        // GET: /Evaluacion/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Evaluacion evaluacion = db.Evaluaciones.Find(id);
            if (evaluacion == null)
            {
                return HttpNotFound();
            }
            return View(evaluacion);
        }

        //
        // POST: /Evaluacion/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Evaluacion evaluacion)
        {
            if (ModelState.IsValid)
            {
                db.Entry(evaluacion).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(evaluacion);
        }

        //
        // GET: /Evaluacion/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Evaluacion evaluacion = db.Evaluaciones.Find(id);
            if (evaluacion == null)
            {
                return HttpNotFound();
            }
            return View(evaluacion);
        }

        //
        // POST: /Evaluacion/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Evaluacion evaluacion = db.Evaluaciones.Find(id);
            db.Evaluaciones.Remove(evaluacion);
            db.SaveChanges();
```

```
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
```

## Fuente de Controlador de Catálogo de Facultad

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class FacultadController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Facultad/

        public ActionResult Index()
        {
            var facultades = db.Facultades.Include(f => f.Recintos);
            return View(facultades.ToList());
        }

        //
        // GET: /Facultad/Details/5

        public ActionResult Details(int id = 0)
        {
            Facultad facultad = db.Facultades.Find(id);
            if (facultad == null)
            {
                return HttpNotFound();
            }
            return View(facultad);
        }

        //
        // GET: /Facultad/Create

        public ActionResult Create()
        {
            ViewBag.IdRecinto = new SelectList(db.Recintos, "IdRecinto",
"Nombre");
            return View();
        }

        //
        // POST: /Facultad/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Facultad facultad)
        {
            if (ModelState.IsValid)
```

```
            {
                facultad.Estado = true;
                db.Facultades.Add(facultad);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdRecinto = new SelectList(db.Recintos, "IdRecinto", "Nombre",
facultad.IdRecinto);
            return View(facultad);
        }

        //
        // GET: /Facultad/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Facultad facultad = db.Facultades.Find(id);
            if (facultad == null)
            {
                return HttpNotFound();
            }
            ViewBag.IdRecinto = new SelectList(db.Recintos, "IdRecinto", "Nombre",
facultad.IdRecinto);
            return View(facultad);
        }

        //
        // POST: /Facultad/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Facultad facultad)
        {
            if (ModelState.IsValid)
            {
                db.Entry(facultad).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.IdRecinto = new SelectList(db.Recintos, "IdRecinto", "Nombre",
facultad.IdRecinto);
            return View(facultad);
        }

        //
        // GET: /Facultad/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Facultad facultad = db.Facultades.Find(id);
            if (facultad == null)
            {
                return HttpNotFound();
            }
            return View(facultad);
        }
```

```
        //
        // POST: /Facultad/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Facultad facultad = db.Facultades.Find(id);
            db.Facultades.Remove(facultad);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Home (Formulario Inicial)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

using System.Data;
using System.Data.Entity;

using System.Web.Security;
using System.Data.SqlClient;


namespace Mono.Controllers
{
    public class HomeController : Controller
    {
        private MonoDB db = new MonoDB();

        public ActionResult Index()
        {

            ViewBag.Message = "SISBECA";

            string querystr = "SELECT * FROM Tarea AS t WHERE t.IdTarea NOT IN
(SELECT a.TareaId " +
                                                 " FROM Asignacion AS a INNER
JOIN AlumnoPeriodo AS ap ON ap.IdPeriodoAlumno = a.IdAlumnoPeriodo " +
                                                 " INNER JOIN PeriodoAcademico AS
pa ON pa.IdPeriodoAcademico = ap.IdPeriodoAcademico " +
                                                 " WHERE pa.Estado=1)";

            List<Tarea> tareas = db.Tareas.SqlQuery(querystr).ToList();
                //db.Tareas.Include(x =>
x.Asignaciones.Select(xx=>xx.AlumnosPeriodos).Select(a=>a.Periodos).Where(a=>a.Est
ado==true)).ToList();

            //List<Tarea> t = db.Tareas.Include(x);


            //List<Tarea> tareas =db.

            List<PeriodoAcademico> periodo = db.Periodo_Academicos.Where(p =>
p.Estado == true).ToList();
            ViewBag.periodo = periodo;

            List<Alumno> alumnos = db.Alumnos.SqlQuery(
                "SELECT TOP 5 * FROM Alumno AS a " +
                " INNER JOIN AlumnoPeriodo AS ap ON ap.Carnet = a.Carnet " +
                " INNER JOIN PeriodoAcademico AS pa ON pa.IdPeriodoAcademico =
ap.IdPeriodoAcademico " +
                " WHERE ap.IdPeriodoAlumno  not IN (SELECT a2.IdAlumnoPeriodo " +
                " FROM Asignacion AS a2 ) AND pa.Estado=1"). ToList();
```

```
            int Cantalumno = db.Alumnos.SqlQuery(
                "SELECT  * FROM Alumno AS a " +
                 " INNER JOIN AlumnoPeriodo AS ap ON ap.Carnet = a.Carnet " +
                " INNER JOIN PeriodoAcademico AS pa ON pa.IdPeriodoAcademico =
ap.IdPeriodoAcademico " +
                " WHERE ap.IdPeriodoAlumno  not IN (SELECT a2.IdAlumnoPeriodo " +
                " FROM Asignacion AS a2 ) AND pa.Estado=1").Count();

            ViewBag.alumnonum = Cantalumno;
            ViewBag.alumnos = alumnos;

            ViewBag.periodo = periodo;

            ViewBag.tareas = tareas;
            //Alumno alum= db.AlumnosPeriodo.Include(x=>x.Alumnos).Where(a=>a.)


            return View();
        }

        [Authorize]
        public void CreateMenu()
        {
            List<Menu> MenuC = db.Menu.Include(m => m.MenuItems).ToList();

            //List<Menu> MenuC = db.Menu.
            //  SqlQuery("SELECT DISTINCT m.*, mi.* FROM UserPermisoMenu AS upm  "
+
            //              " INNER JOIN webpages_UsersInRoles AS wuir ON
wuir.RoleId = upm.RoleId  " +
            //              " INNER JOIN UserProfile AS up ON up.UserId =
wuir.UserId  " +
            //              " INNER JOIN MenuItem AS mi ON mi.id=upm.MenuItemId "
+
            //              " INNER JOIN Menu AS m ON m.Id = mi.ParentMenu " +
            //              " WHERE up.UserName=@p0 AND upm.Acceso=1", new
SqlParameter("p0", User.Identity.Name)).ToList();


            var MenuRevisar = db.UserPermisosMenu.
                SqlQuery("SELECT distinct 0 AS RoleId, upm.MenuItemId, upm.Acceso
FROM UserPermisoMenu AS upm " +
                         " INNER JOIN webpages_UsersInRoles AS wuir ON wuir.RoleId
= upm.RoleId " +
                         " INNER JOIN UserProfile AS up ON up.UserId = wuir.UserId
" +
                        " INNER JOIN MenuItem AS mi ON mi.id=upm.MenuItemId " +
                        " WHERE up.UserName=@p0 AND upm.Acceso=1", new
SqlParameter("p0",User.Identity.Name)).ToList();


            List<int> acceso = new List<int>();

            foreach(UserPermisoMenu upm in MenuRevisar)
            {
                acceso.Add(upm.MenuItemId);
            }
```

```
            //List<Menu> MenuC = db.Menu.Include(x => x.MenuItems.Any(p =>
acceso.Contains(p.Id))).ToList();

            //for (int i = 0; i < MenuC.Count() - 1; i++)
            //{
            //    foreach (MenuItem mi in MenuC[i].MenuItems)
            //    {
            //      if(!(MenuRevisar.Where(x=>x.MenuItemId==mi.Id).Count()>0))
            //      {
            //          MenuC[i].MenuItems.Remove(mi);
            //      }

            //        //if MenuC[i].MenuItems.Contains()
            //    }
            //}

                        //HASTA AKI KEDE falta ver como filtro el menu ;(

            string html = "";
            foreach(Menu m in MenuC){
                html += "<li> <a href='javascript:;' data-toggle='collapse' data-
target='#"+m.Id+"'><i class='"+m.glyp+"'></i> "+m.Nombre+"<i class='fa fa-fw fa-
caret-down'></i></a>";
                if (m.MenuItems.Count > 0) {
                    html += "<ul id='"+m.Id+"' class='collapse'>";
                    foreach(MenuItem mI in m.MenuItems) {
                        if (acceso.Contains(mI.Id))
                        {
                            html += "<li> <a href='/" + mI.ControllerName + "/" +
mI.ActionName + "'>" + mI.Nombre + "</a></li>";
                        }
                    }
                    html += "</ul>";
                }
                html += "</li>";
            }
            Response.Write(html);
        }

        public ActionResult About()
        {
            ViewBag.Message = "Acerca de SISBECA";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Egresado de Ingenieria en Computacion";

            return View();
        }


        protected override void Dispose(bool disposing)
        {
```

```
            db.Dispose();
            base.Dispose(disposing);
        }
    }


}
```

## Fuente de Controlador de Catálogo de Periodos Académicos

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Transactions;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class PeriodoAcademicoController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /PeriodoAcademico/

        public ActionResult Index()
        {
            return View(db.Periodo_Academicos.ToList());
        }

        //
        // GET: /PeriodoAcademico/Details/5

        public ActionResult Details(int id = 0)
        {
            PeriodoAcademico periodoacademico = db.Periodo_Academicos.Find(id);
            if (periodoacademico == null)
            {
                return HttpNotFound();
            }
            return View(periodoacademico);
        }

        //
        // GET: /PeriodoAcademico/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /PeriodoAcademico/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(PeriodoAcademico periodoacademico)
        {
            if (ModelState.IsValid)
            {
                using (TransactionScope scope = new TransactionScope())
```

```
                {
                    try
                    {
                        var periodos = db.Periodo_Academicos;

                        foreach (PeriodoAcademico p in periodos)
                        {
                            p.Estado = false;
                            db.Entry(p).State = EntityState.Modified;


                            Auditoria aud = new Auditoria();
                            aud.Referencia = "id: " +
p.IdPeriodoAcademico.ToString();
                            aud.Accion = "Actualizacion automatica (estado) por
alta";
                            aud.Objeto = "PeriodoAcademico";
                            aud.Equipo = "pedro"; //User.Identity.Name;
                            aud.Fecha = DateTime.Now;

                            db.Auditorias.Add(aud);

                        }

                        db.Periodo_Academicos.Add(periodoacademico);

                        Auditoria aper = new Auditoria();
                        aper.Referencia = periodoacademico.Ciclo;
                        aper.Accion = "Alta periodo" +
periodoacademico.Nombre_Unico;
                        aper.Objeto = "PeriodoAcademico";
                        aper.Equipo = "pedro"; //User.Identity.Name;
                        aper.Fecha = DateTime.Now;
                        db.Auditorias.Add(aper);

                        db.SaveChanges();
                        scope.Complete();
                        return RedirectToAction("Index");
                    }
                    catch (Exception ex)
                    {
                        ModelState.AddModelError("", ex.Message);
                    }
                }


            }

            return View(periodoacademico);
        }

        //
        // GET: /PeriodoAcademico/Edit/5

        public ActionResult Edit(int id = 0)
        {
            PeriodoAcademico periodoacademico = db.Periodo_Academicos.Find(id);
            if (periodoacademico == null)
```

```
            {
                return HttpNotFound();
            }
            return View(periodoacademico);
        }

        //
        // POST: /PeriodoAcademico/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(PeriodoAcademico periodoacademico)
        {
            if (ModelState.IsValid)
            {
                using (TransactionScope scope= new TransactionScope())
                {
                    try
                    {
                        if (periodoacademico.Estado == true)

                        {
                            var periodos =
db.Periodo_Academicos.Where(p=>p.IdPeriodoAcademico !=
periodoacademico.IdPeriodoAcademico);

                            foreach (PeriodoAcademico p in periodos)
                            {
                                p.Estado = false;
                                db.Entry(p).State = EntityState.Modified;


                                Auditoria aud = new Auditoria();
                                aud.Referencia = "id: " +
p.IdPeriodoAcademico.ToString();
                                aud.Accion = "Actualizacion automatica (estado)";
                                aud.Objeto = "PeriodoAcademico";
                                aud.Equipo = "pedro"; //User.Identity.Name;
                                aud.Fecha = DateTime.Now;

                                db.Auditorias.Add(aud);

                            }
                            db.SaveChanges();

                            //db.Periodo_Academicos.SqlQuery("update
PeriodoAcademico set Estado=0 where IdPeriodoAcademico<>" +
periodoacademico.IdPeriodoAcademico);

                        }
                        db.Entry(periodoacademico).State = EntityState.Modified;
                        db.SaveChanges();
                        scope.Complete();
                        return RedirectToAction("Index");
                    }
                    catch(Exception ex)
                    {
                        ModelState.AddModelError("",ex.Message);
```

```
                    }

                }


            }
            return View(periodoacademico);
        }

        //
        // GET: /PeriodoAcademico/Delete/5

        public ActionResult Delete(int id = 0)
        {
            PeriodoAcademico periodoacademico = db.Periodo_Academicos.Find(id);
            if (periodoacademico == null)
            {
                return HttpNotFound();
            }
            return View(periodoacademico);
        }

        //
        // POST: /PeriodoAcademico/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            PeriodoAcademico periodoacademico = db.Periodo_Academicos.Find(id);
            db.Periodo_Academicos.Remove(periodoacademico);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Catálogo de Recintos

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class RecintoController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Recinto/

        public ActionResult Index()
        {

            return View(db.Recintos.ToList());
        }

        //
        // GET: /Recinto/Details/5

        public ActionResult Details(int id = 0)
        {
            Recinto recinto = db.Recintos.Find(id);
            if (recinto == null)
            {
                return HttpNotFound();
            }
            return View(recinto);
        }

        //
        // GET: /Recinto/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Recinto/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Recinto recinto)
        {
            if (ModelState.IsValid)
            {
                recinto.Estado = true;
```

```
                db.Recintos.Add(recinto);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(recinto);
        }

        //
        // GET: /Recinto/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Recinto recinto = db.Recintos.Find(id);
            if (recinto == null)
            {
                return HttpNotFound();
            }
            return View(recinto);
        }

        //
        // POST: /Recinto/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Recinto recinto)
        {
            if (ModelState.IsValid)
            {
                db.Entry(recinto).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(recinto);
        }

        //
        // GET: /Recinto/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Recinto recinto = db.Recintos.Find(id);
            if (recinto == null)
            {
                return HttpNotFound();
            }
            return View(recinto);
        }

        //
        // POST: /Recinto/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Recinto recinto = db.Recintos.Find(id);
```

```
            db.Recintos.Remove(recinto);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Formularios de Reportes

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

using System.Web.UI.WebControls;
using Microsoft.Reporting.WebForms;
using System.IO;

namespace Mono.Controllers
{
    public class ReportesController : Controller
    {
        //
        // GET: /Reportes/

        public ActionResult Index()
        {
            return View();
        }

        private void SetLocalReport()
        {
            ReportViewer reportViewer = new ReportViewer();
            reportViewer.ProcessingMode = ProcessingMode.Local;
            reportViewer.SizeToReportContent = true;
            reportViewer.Width = Unit.Percentage(100);
            reportViewer.Height = Unit.Percentage(100);

            List<Alumno> alumnos = new List<Alumno>();
            using (MonoDB dr = new MonoDB())
            {
                alumnos = dr.Alumnos.ToList();
            }
            ReportDataSource rd = new ReportDataSource("DataSet1", alumnos);

            //FillDataSet();
            reportViewer.LocalReport.ReportPath =
Request.MapPath(Request.ApplicationPath) + @"Reportes\rep_estudiantes.rdlc";
            reportViewer.LocalReport.DataSources.Add(rd);
            //reportViewer.LocalReport.SetParameters(GetParametersLocal());

            ViewBag.ReportViewer = reportViewer;
        }

        public ActionResult LocalReportExample()
        {
            SetLocalReport();

            return View();
        }

        public ActionResult RptCatEstudiante(string id)
```

```
        {

            LocalReport lr = new LocalReport();

            string path = Path.Combine(Server.MapPath("~/Reportes"),
"rep_estudiantes.rdlc");
            if (System.IO.File.Exists(path))
            {
                lr.ReportPath = path;
            }
            else
            {
                return View("Index");
            }
            List<Alumno> alumnos = new List<Alumno>();
            using (MonoDB dr = new MonoDB())
            {
                alumnos = dr.Alumnos.ToList();
            }
            ReportDataSource rd = new ReportDataSource("SIPSERBEDataSet",
alumnos);
            lr.DataSources.Add(rd);
            string reportType = id;
            string mimeType;
            string encoding;
            string fileNameExtension;

            string deviceInfo =

           "<DeviceInfo>" +
           "  <OutputFormat>" + id + "</OutputFormat>" +
           "  <PageWidth>8.5in</PageWidth>" +
           "  <PageHeight>11in</PageHeight>" +
           "  <MarginTop>0.5in</MarginTop>" +
           "  <MarginLeft>1in</MarginLeft>" +
           "  <MarginRight>1in</MarginRight>" +
           "  <MarginBottom>0.5in</MarginBottom>" +
           "</DeviceInfo>";

            Warning[] warnings;
            string[] streams;
            byte[] renderedBytes;

            renderedBytes = lr.Render(
                reportType,
                deviceInfo,
                out mimeType,
                out encoding,
                out fileNameExtension,
                out streams,
                out warnings);

            return File(renderedBytes, mimeType);

        }

    }
}
```

## Fuente de Controlador de Catálogo de Tareas

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class TareaController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /Tarea/

        public ActionResult Index()
        {
            var tareas = db.Tareas.Include(t => t.TipoTareas).Include(t =>
t.Areas).Include(t => t.Beneficiarios);
            return View(tareas.ToList());
        }

        //
        // GET: /Tarea/Details/5

        public ActionResult Details(int id = 0)
        {
            Tarea tarea = db.Tareas.Find(id);
            if (tarea == null)
            {
                return HttpNotFound();
            }
            return View(tarea);
        }

        //
        // GET: /Tarea/Create

        public ActionResult Create()
        {
            ViewBag.TipoTareaId = new SelectList(db.TiposTareas, "IdTipoTarea",
"Descripcion");
            ViewBag.IdArea = new SelectList(db.Areas, "IdArea", "Nombre");
            ViewBag.IdBeneficiario = new SelectList(db.Beneficiarios,
"IdBeneficiario", "Nombre");
            ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones, "IdEvaluacion",
"Descripcion");
            return View();
        }

        //
        // POST: /Tarea/Create
```

```
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Tarea tarea)
        {
            if (ModelState.IsValid)
            {
                db.Tareas.Add(tarea);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.TipoTareaId = new SelectList(db.TiposTareas, "IdTipoTarea",
"Descripcion", tarea.TipoTareaId);
            ViewBag.IdArea = new SelectList(db.Areas, "IdArea", "Nombre",
tarea.IdArea);
            ViewBag.IdBeneficiario = new SelectList(db.Beneficiarios,
"IdBeneficiario", "Nombre", tarea.IdBeneficiario);
            //ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", tarea.IdEvaluacion);
            return View(tarea);
        }

        //
        // GET: /Tarea/Edit/5

        public ActionResult Edit(int id = 0)
        {
            Tarea tarea = db.Tareas.Find(id);
            if (tarea == null)
            {
                return HttpNotFound();
            }
            ViewBag.TipoTareaId = new SelectList(db.TiposTareas, "IdTipoTarea",
"Descripcion", tarea.TipoTareaId);
            ViewBag.IdArea = new SelectList(db.Areas, "IdArea", "Nombre",
tarea.IdArea);
            ViewBag.IdBeneficiario = new SelectList(db.Beneficiarios,
"IdBeneficiario", "Nombre", tarea.IdBeneficiario);
            //ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", tarea.IdEvaluacion);
            return View(tarea);
        }

        //
        // POST: /Tarea/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Tarea tarea)
        {
            if (ModelState.IsValid)
            {
                db.Entry(tarea).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
```

```
            ViewBag.TipoTareaId = new SelectList(db.TiposTareas, "IdTipoTarea",
"Descripcion", tarea.TipoTareaId);
            ViewBag.IdArea = new SelectList(db.Areas, "IdArea", "Nombre",
tarea.IdArea);
            ViewBag.IdBeneficiario = new SelectList(db.Beneficiarios,
"IdBeneficiario", "Nombre", tarea.IdBeneficiario);
            //ViewBag.IdEvaluacion = new SelectList(db.Evaluaciones,
"IdEvaluacion", "Descripcion", tarea.IdEvaluacion);
            return View(tarea);
        }

        //
        // GET: /Tarea/Delete/5

        public ActionResult Delete(int id = 0)
        {
            Tarea tarea = db.Tareas.Find(id);
            if (tarea == null)
            {
                return HttpNotFound();
            }
            return View(tarea);
        }

        //
        // POST: /Tarea/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Tarea tarea = db.Tareas.Find(id);
            db.Tareas.Remove(tarea);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Catálogo Tipo de Becas

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class TipoBecaController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /TipoBeca/

        public ActionResult Index()
        {
            return View(db.TiposBecas.ToList());
        }

        //
        // GET: /TipoBeca/Details/5

        public ActionResult Details(int id = 0)
        {
            TipoBeca tipobeca = db.TiposBecas.Find(id);
            if (tipobeca == null)
            {
                return HttpNotFound();
            }
            return View(tipobeca);
        }

        //
        // GET: /TipoBeca/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /TipoBeca/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(TipoBeca tipobeca)
        {
            if (ModelState.IsValid)
            {
                db.TiposBecas.Add(tipobeca);
                db.SaveChanges();
```

```
                return RedirectToAction("Index");
            }

            return View(tipobeca);
        }

        //
        // GET: /TipoBeca/Edit/5

        public ActionResult Edit(int id = 0)
        {
            TipoBeca tipobeca = db.TiposBecas.Find(id);
            if (tipobeca == null)
            {
                return HttpNotFound();
            }
            return View(tipobeca);
        }

        //
        // POST: /TipoBeca/Edit/5

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(TipoBeca tipobeca)
        {
            if (ModelState.IsValid)
            {
                db.Entry(tipobeca).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(tipobeca);
        }

        //
        // GET: /TipoBeca/Delete/5

        public ActionResult Delete(int id = 0)
        {
            TipoBeca tipobeca = db.TiposBecas.Find(id);
            if (tipobeca == null)
            {
                return HttpNotFound();
            }
            return View(tipobeca);
        }

        //
        // POST: /TipoBeca/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            TipoBeca tipobeca = db.TiposBecas.Find(id);
            db.TiposBecas.Remove(tipobeca);
            db.SaveChanges();
```

```
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente de Controlador de Tipo de Tarea

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Mono.Models;

namespace Mono.Controllers
{
    public class TipoTareaController : Controller
    {
        private MonoDB db = new MonoDB();

        //
        // GET: /TipoTarea/

        public ActionResult Index()
        {
            return View(db.TiposTareas.ToList());
        }

        //
        // GET: /TipoTarea/Details/5

        public ActionResult Details(int id = 0)
        {
            TipoTarea tipotarea = db.TiposTareas.Find(id);
            if (tipotarea == null)
            {
                return HttpNotFound();
            }
            return View(tipotarea);
        }

        //
        // GET: /TipoTarea/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /TipoTarea/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(TipoTarea tipotarea)
        {
            if (ModelState.IsValid)
            {
                db.TiposTareas.Add(tipotarea);
                db.SaveChanges();
```

```
            return RedirectToAction("Index");
        }

        return View(tipotarea);
    }

    //
    // GET: /TipoTarea/Edit/5

    public ActionResult Edit(int id = 0)
    {
        TipoTarea tipotarea = db.TiposTareas.Find(id);
        if (tipotarea == null)
        {
            return HttpNotFound();
        }
        return View(tipotarea);
    }

    //
    // POST: /TipoTarea/Edit/5

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit(TipoTarea tipotarea)
    {
        if (ModelState.IsValid)
        {
            db.Entry(tipotarea).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        return View(tipotarea);
    }

    //
    // GET: /TipoTarea/Delete/5

    public ActionResult Delete(int id = 0)
    {
        TipoTarea tipotarea = db.TiposTareas.Find(id);
        if (tipotarea == null)
        {
            return HttpNotFound();
        }
        return View(tipotarea);
    }

    //
    // POST: /TipoTarea/Delete/5

    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(int id)
    {
        TipoTarea tipotarea = db.TiposTareas.Find(id);
        db.TiposTareas.Remove(tipotarea);
        db.SaveChanges();
```

```
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

## Fuente Jquery de archivo common.js

```
; (function ($, window, undifined) {
    $.fn.resizeScreen = function (capa) {
        return this.bind('resize load', function () {
            $(capa).css({ 'min-height': window.innerHeight - 50 });
        });
    }

    $.fn.clearInput = function () {
        return this.on('keypress', function (event) {
            var keycode = (event.keyCode ? event.keyCode : event.which);
            if (keycode == '13') {
                event.preventDefault();
                $(this).val('');
            }
        });
    }

    $.fn.createMenu = function () {
        return this.on("load", function () {
            $.ajax({
                type: "POST",
                dataType: "html",
                url: "/Home/CreateMenu",
                success: function (data) {
                    $(".menuiz").append(data);
                }
            });
        });
    }

    $.fn.saveMultipleAperiodo = function () {
        return this.on('click', function (event) {
            event.preventDefault();
            var reg;

            $('#tbalumno tbody tr').each(function () {

                var token = $('input[name="__RequestVerificationToken"]').val();

                var fe = new Date($.now());

                var reg = {
                    Carnet: $(this).attr('id'),
                    IdPeriodoAcademico: $(this).attr('data-code'),
                    IdCarrera: $(this).attr('data-carrera'),
                    IdTipoBeca: $(this).attr('data-beca'),
                    __RequestVerificationToken: token
                };
                var headers = {};

                headers['__RequestVerificationToken'] = token;

                $.ajax({
                    url: '/Aperiodo/create',
                    headers: headers,
```

```
                    type: 'POST',
                    dataType: 'json',
                    data: reg,
                    success: function () {

                    }

                });
            });
        });
    }
})(jQuery, window);

$(function () {


    var createAutocomplete = function () {
        var $input = $(this);

        var options = {
            source: $input.attr("data-mvc-autocomplete"),
            select: function (event, ui) {
                $('#tbalumno tbody').append('<tr id=' + ui.item.label.substring(0,
10) + ' data-code=' + $("#lperiodo option:selected").val()
                    + ' data-carrera=' + $("#lcarrera option:selected").val() + '
data-beca=' + $("#lbeca option:selected").val() + '><td>'
                    + ui.item.label.substring(0, 10) + '</td><td>' +
ui.item.label.substring(10, 50) +
                    '</td><td>' + $("#lperiodo option:selected").text() +
'</td><td>' + $("#lcarrera option:selected").text()
                    + '</td><td>' + $("#lbeca option:selected").text() +
'</td></tr>')
            }
        }
        $input.autocomplete(options);
    };


    var aperiodoAutocompleteA = function () {
        var $input = $(this);

        var options = {
            source: $input.attr("data-mvc-autocompleteA"),
            select: function (event, ui) {
                //alert(ui.item.id);
                $('#alperiodo').val(ui.item.id)


            }
        }

        $input.autocomplete(options);
    };

    var asignacionAutocomplete = function () {
        var $input = $(this);
        //data - mvc - completaasig
        var options = {
```

```
            source: $input.attr("data-mvc-autocompleteasig"),
            select: function (event, ui) {
                //alert(ui.item.id);
                $('#IdAlumnoPeriodo').val(ui.item.id)

            }
        }

        $input.autocomplete(options);
    };


    $("input[data-mvc-autocompleteasig]").each(asignacionAutocomplete);
    $("input[data-mvc-autocomplete]").each(createAutocomplete);
    $("input[data-mvc-autocompleteA]").each(aperiodoAutocompleteA);
    //anchor link
    //$("a[data-carrito-agregar='true']").click(agregarProducto);
});
```