Universidad Nacional Autónoma de México
Facultad de Ingeniería

Sistema de Control para Prestadores de Servicio y Becarios de la DGSCA

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERA EN COMPUTACIÓN

PRESENTAN

Aparicio Arista Reyna Elizabeth

Rosas Bernal María del Rosario

DIRECTOR DE TESIS

Ing. Carlos Alberto Román Zamitiz

CIUDAD UNIVERSITARIA, MÉXICO D.F.

Octubre, 2009

Agradecimientos

A mis padres Juanita Arista y Roberto Aparicio, porque son mi inspiración para ser mejor persona cada día, porque me han enseñado el verdadero valor de las cosas, porque a ellos debo mi formación como ser humano y como profesionista. No me alcanzará la vida para demostrarles mi gratitud, los amo.

A mi hermana Angélica, por estar cerca de mí siempre.

A Isaac, por ser una motivación más en mi vida, gracias por tu confianza.

A Rosario, gracias por los momentos compartidos. Sé que sin tu apoyo hubiera sido mucho más difícil concluir este trabajo.

A Naye, Jav y Chars que me han dejado compartir con ellos valor de la amistad.

A la UNAM, en especial a la Facultad de Ingeniería y a cada uno de mis profesores.

Elizabeth Aparicio

A mis padres y hermanos, por darme la estabilidad emocional, económica, sentimental; para poder llegar hasta este logro, se que hubo momentos en los que los descuide y que no dedique el tiempo para convivir con ustedes, pero siempre estuvieron en los momentos difíciles apoyándome y dándome el aliento necesario para continuar, se que definitivamente no hubiese podido ser realidad sin ustedes. GRACIAS

A todos mis amigos pasados y presentes; pasados por ayudarme a crecer y madurar como persona y presentes por estar siempre conmigo apoyándome en todas las circunstancias posibles, también son parte de esta alegría.

A todos mis profesores por compartir sus conocimientos, mil gracias Facultad de Ingeniería.

Y a todos aquellos, que han quedado en los recintos más escondidos de mi memoria, pero que fueron participes en cincelarme, Gracias.

Les dedico esta tesis a mis sobrinos, los quiero.

Rosario Rosas

Contenido

Índice de Figuras

1. Introducción

Los sistemas informáticos en conjunto con las tecnologías de información, han revolucionado la manera en la que operan las organizaciones desde una empresa del sector privado hasta una institución educativa. A través de su uso se logran importantes mejoras, puesto que se automatizan los procesos operativos que la mayoría de las veces, resultan muy costosos además de que dichas tecnologías brindan una plataforma de información que permite la toma de decisiones de manera más eficiente.

Hoy en día es, si no es indispensable, si es muy benéfico contar con un sistema informático, es por ello, que a partir de la identificación de la carencia de un sistema que permitiera el registro y seguimiento de las actividades de prestadores de servicio en la Dirección General de Servicios de Cómputo Académico de la UNAM (DGSCA), surge este trabajo de tesis.

En DGSCA, cada semestre se presenta la rotación de alumnos que realizan su servicio social o que ingresan a un programa de becarios los cuales presentan una solicitud de ingreso a dicha dependencia. Anteriormente toda la administración derivada de los procesos de negocio que se realiza para el ingreso y control de alumnos, se manejaba manualmente por parte de la Coordinación de Servicio Social, de manera que era un proceso poco eficiente.

Se eligió que dicho sistema tuviera una perspectiva Web debido a que, se tiene una mayor cobertura, así mismo se aprovecharon las ventajas que este tipo de sistemas ofrecen a los usuarios, tales como:

- La información es accesible desde cualquier lugar dentro de la dependencia e incluso desde el exterior.
- La información es compartida entre las partes interesadas, de manera que todas tienen acceso a la información completa o a la parte que les corresponde según su función en todo momento.

En el presente capítulo se describe de manera general el contenido de cada unos de los capítulos subsecuentes en los cuales se aborda el proceso de desarrollo de software que se realizó para resolver dicha problemática.

En el capítulo 2, se presentan las bases teóricas de orientación a objetos, patrones de diseño, bases de datos, tecnologías e ingeniería de software incluyendo el proceso de desarrollo RUP y el lenguaje de modelado UML, que se aplicaron a lo largo del desarrollo del sistema.

Una de las fases más importantes en el desarrollo de software es la de inicio puesto que en esta fase se debe tener definido el modelo de negocio, para posteriormente entender las necesidades del usuario a través de los requerimientos representados en los casos de uso y diagramas de actividades, es por ello que el capítulo 3 esta dedicado a esta fase.

En el capítulo 4 se define la fase de elaboración, en la cual se detalla el análisis de los requerimientos obtenidos, así como el diseño mediante la creación de prototipos y el planteamiento de la solución tecnológica a través de UML, incluyendo el diseño de la base de datos.

La fase de construcción es descrita en el capítulo 5, se explica cómo fueron construidos los componentes de software en base al diseño realizado, se presentan los casos de prueba efectuados junto con sus evidencias.

Al finalizar el proyecto, se presenta una carta de liberación del producto final, visualizada en el capítulo 6 a través de la fase de transición, la cual también abarca como parte de los entregables, las lecciones aprendidas que en nuestro caso forman parte de las conclusiones.

Finalmente, en el capítulo 7 se presentan las conclusiones a las que se llegó al terminar el desarrollo del sistema por parte de los involucrados en el mismo, su impacto en DGSCA así como los beneficios para los usuarios finales.

2. Bases Teóricas

A lo largo de los años, el desarrollo de software se ha vuelto cada vez más complejo, es muy común que durante el desarrollo de algún proyecto de software, éste sufra retrasos, utilice más recursos de los planeados, se modifique el alcance planeado originalmente entre otras situaciones. Las razones o causas de lo anterior pueden ser varias, entre ellas se puede mencionar: la mala estimación de los proyectos en cuanto a tiempo y recursos se refiere, la mala definición de requerimientos; puesto que generalmente un usuario no tiene claro que es lo que espera de un sistema, así como la carencia de un modelado adecuado de las características y componentes que se requieren en la aplicación.

Es por ello que hoy en día la utilización de elementos como los diagramas de UML en la fase de análisis y diseño, permiten modelar y documentar los procesos de negocios, sus interfaces, requerimientos y actividades basados en el paradigma orientado a objetos, lo cual contribuye a un desarrollo de software más eficiente y eficaz, así como a facilitar la comunicación entre desarrolladores, analistas y usuarios, mediante el uso de un lenguaje común.

2.1. Programación Orientada a Objetos (POO)

2.1.1. Conceptos Básicos

Un paradigma es un modelo o ejemplo a seguir, por una comunidad científica, es un ejemplo de los problemas que tiene que resolver y del modo como se van a dar las soluciones. Un paradigma es una manera de entender el mundo, explicarlo y manipularlo.

El paradigma Orientado a Objetos es una poderosa metodología de diseño basada en los siguientes principios:

a) Estructurar un sistema en componentes modulares que se pueden desarrollar, mantener y reutilizar por separado de manera que se pueda tener un control adecuado de la complejidad de dicho sistema.
b) Basar la estructura y comportamiento de la solución en un problema que haya sido solucionado previamente.
c) Elevar el nivel de abstracción de los componentes que serán construidos.
d) Utilizar un lenguaje común con el cliente, de manera que el negocio y la tecnología puedan ligarse y se pueda mejorar la comunicación.
e) Describir el problema de manera precisa y de forma que se evite entrar en detalles técnicos.
f) Permitir las bases para una administración efectiva del proceso de desarrollo y tener el mejor costo – beneficio posible.
g) Automatizar tareas repetitivas de diseño e implementación.
h) Facilitar los procesos iterativo – incremental, de arquitectura y de pruebas para mitigar el riesgo.
i) Responder rápidamente a los cambios de acuerdo a las necesidades de los usuarios.

Siguiendo los principios mencionados anteriormente, la programación orientada a objetos, intenta simular el mundo real a través del significado de un objeto que contiene características y funciones.

La Programación Orientada a Objetos está basada en conceptos tales como: objeto, clase y método y además cuenta con una serie de características que incluyen: herencia, abstracción, jerarquía, polimorfismo, encapsulamiento y modularidad. A continuación se describe cada uno de estos conceptos brevemente:

a) *Objeto*

Un objeto es la unidad atómica que encapsula un estado y comportamiento, este puede caracterizar una entidad física o abstracta. En el mundo cotidiano un objeto representa un elemento identificable que está constituido por ciertas características o atributos y es capaz de realizar un conjunto de acciones u operaciones. En este sentido, un objeto posee:

- Estado: El estado de un objeto lo constituyen todos los datos que encapsula en un momento determinado.
- Comportamiento: la manera en que actúa y reacciona un objeto, esto en función de sus cambios de estado y el paso de mensajes.
- Identidad: La idea es que los objetos no se definen por los valores de sus atributos en un momento determinado, un objeto tiene una existencia continua.

En la programación orientada a objetos, un objeto es una instancia de una clase. Los objetos le ofrecen al programador una herramienta para implementar: modularidad y simulación.

b) *Clase*

Una clase es un conjunto de objetos que comparten una estructura y un comportamiento común, describe un conjunto de objetos que tienen un rol o roles equivalentes en un sistema.

En otras palabras una clase es un grupo de objetos con características comunes (atributos), comportamiento común (operaciones) y relaciones comunes con otros objetos (asociaciones y agregaciones). Una clase es una definición abstracta de un objeto que:

- Enfatiza las características relevantes de los objetos.
- Define la estructura y comportamiento de un objeto.
- Sirve como "plantilla" para crear nuevos objetos.

c) *Mensaje*

A la unidad de comunicación entre los objetos se le llama mensaje. Un mensaje es en esencia, la petición de un servicio; posee un emisor, un receptor y una acción.

Para que los objetos de un sistema trabajen en conjunto, un objeto envía a otro objeto, una llamada para realizar una operación y el objeto receptor ejecutará la operación.

d) *Método*

Como se menciono anteriormente, los objetos pueden realizar operaciones, dichas operaciones se conocen como métodos. Un método es un algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje".

Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

Características

a) *Abstracción*

La abstracción es la propiedad que nos permite representar las características esenciales de un objeto, permitiéndonos omitir las propiedades y acciones de un objeto que no son de nuestro interés y dejar sólo aquellas que nos interesan para una situación en particular.

b) *Herencia*

El concepto de herencia, dentro del Paradigma Orientado a Objetos, es uno de los más significativos y de más trascendencia. Consiste en trasladar al contexto de objetos el mas puro significado de la palabra "herencia", es decir el hecho de que una clase "padre" herede características, estructuras y comportamiento a todos las clases "hijas". En sentido estricto se reutilizan las clases para definir nuevas.

La clase general o "padre" es llamada superclase y a la clase especializada o "hija" es llamada subclase. La superclase define un comportamiento (protocolo) que todas las subclases heredan. Asimismo, las subclases pueden agregar nuevos atributos y operaciones. Las subclases pueden sobre-escribir operaciones conservando la firma, es decir la manera en la que esta declarado, pero proporcionando distintas implementaciones.

c) *Jerarquía*

La jerarquía es el concepto, que en conjunto con la propiedad de abstracción, permite clasificar y ordenar a los objetos de dos maneras:

- Generalización/Especialización. Organización establecida de lo particular a lo general, conceptualmente se utiliza la frase "es un" para establecer la generalización.
- Agregación. Organización establecida de las partes hacia el "todo", conceptualmente se utiliza la frase "es parte de" para establecer la agregación.

d) *Polimorfismo*

Es la propiedad por la cual dos o más clases responden al mismo mensaje cada uno de manera diferente, lo que permite realizar la misma operación de forma diferente en cada objeto.

El polimorfismo es una característica de una clase que puede tomar varias formas:

- El comportamiento de una clase puede variar de acuerdo a las circunstancias.
- En ocasiones una operación tiene el mismo nombre en distintas clases.
- Así mismo cada subclase hereda las operaciones pero tiene la posibilidad de modificar el comportamiento de estas operaciones, pero únicamente de manera interna a la clase.

El resultado del código (o acción) que se va a ejecutar es el resultado de un enlace dinámico, el polimorfismo nos permite obtener un comportamiento y diseño coherente de nuestro sistema.

e) *Encapsulamiento*

Esta propiedad en general, hace referencia a cualquier tipo de ocultamiento de una clase.

Permite asegurar que el contenido de la información (atributos y métodos) de un objeto está oculta al mundo exterior, protegiéndola de esa forma y ocultando su implementación a los usuarios. La única manera de acceder a la información de un objeto y manipularla es a través de las operaciones. Las ventajas de la encapsulación son:

- Da seguridad a nuestros datos al protegerlos de accesos no autorizados.
- Disminuye el acoplamiento entre las clases, lo que le da flexibilidad a nuestro sistema.
- Nos permite hacer módulos de nuestro sistema, lo que facilita su mantenimiento.

El encapsulamiento protege los atributos con métodos de accesos que los definen como privados y los métodos o comportamientos que se definen como públicos y de esta manera se puede acceder a la clase ya que este método de acceso permite a otras clases ocupar sus comportamientos.

f) *Modularidad*

La modularidad es la propiedad que permite subdividir una aplicación en partes más pequeñas llamadas módulos. Cada uno de estos módulos puede ser tan independiente como sea posible de la aplicación y de las otras partes.

2.1.2. Ventajas de la POO

A continuación se mencionan las ventajas más importantes de la utilización de programación orientada a objetos:

- La aplicación de conceptos tales como: objetos, clases, herencia, polimorfismo, jerarquía, en los componentes de un sistema de información, facilita que los mismos puedan ser ampliamente reutilizables.
- La reutilización de código, así como el manejo de clases y objetos permite un desarrollo más rápido, flexible y modular de sistemas y aplicaciones.

- La POO favorece la creación de mejores estructuras de información.
- Facilita el mantenimiento y escalabilidad de una aplicación, ya que entre otras cosas permite la incorporación de otras aplicaciones o componentes externos.

Todas estas ventajas traen como consecuencia dos aspectos fundamentales: disminución de los costos de desarrollo y el incremento de la calidad del sistema. El uso y aplicación de la metodología orientada a objetos facilita el diseño e implementación de los sistemas con interfaces gráficas de usuario, así como nos permiten desarrollar sistemas inmersos de tiempo real con una mejor calidad y flexibilidad.

2.2. Patrones de Diseño

2.2.1. Conceptos Básicos

Los patrones de diseño (design patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interfaces.

Los patrones de diseño son soluciones bien documentadas que los expertos aplican para resolver nuevos problemas debido a que han sido utilizadas con éxito en el pasado. Los expertos identifican partes de un problema que son similares a otros problemas que han encontrado anteriormente, después, se basan en la solución aplicada y la generalizan para posteriormente adaptar la solución general al contexto de un problema actual.

El principio básico de los patrones de diseño es desarrollar una forma estandarizada para representar soluciones generales de problemas que se encuentran comúnmente en el desarrollo de software de manera que se pueden obtener algunos beneficios, tales como:

- Construir y proporcionar catálogos de elementos reutilizables en el diseño de software, de manera que se transmita la experiencia de forma eficiente y que facilite el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores, para que la comunicación de las soluciones a los desarrolladores sea más sencilla.
- Estandarizar el modo en que se realiza el diseño.

Los patrones de diseño representan una evolución importante en la abstracción y reutilización del software que como ya se mencionó, son características de la POO.

En este sentido, la abstracción representa la forma que tienen los desarrolladores para resolver problemas complejos dividiéndolos en otros más simples, las soluciones a los problemas más simples que se etiquetan con un nombre, pueden ser utilizadas como bloques para resolver problemas más complicados.

Por otro lado la utilización de código que ha sido anteriormente probado facilita el desarrollo de un proyecto y se consigue una mayor productividad del código.

Uno de los patrones más comúnmente utilizados es el Modelo – Vista – Controlador (MVC), el cual se explica en la siguiente sección.

2.2.2. MVC (Model – View – Controller)

Uno de los patrones que ha demostrado ser fundamental a la hora de diseñar aplicaciones Web es el Modelo-Vista-Controlador (MVC), el cual se clasifica como un patrón de sistema.

Este patrón propone la separación en distintos componentes de la interfaz de usuario (vistas), el modelo de negocio y la lógica de control. Una vista es una "fotografía" del modelo (o una parte del mismo) en un determinado momento. Un control recibe un evento disparado por el usuario a través de la interfaz, accede al modelo de manera adecuada a la acción realizada y presenta en una nueva vista el resultado de dicha acción. Por su parte, el modelo consiste en el conjunto de objetos que modelan los procesos de negocio que se realizan a través del sistema.

Este patrón es muy útil cuando se desea que los componentes sean flexibles y fáciles de mantener. Generalmente se utiliza para los casos en los que se esperan cambios en los componentes y también se espera que sean reutilizables.

Como se menciono anteriormente, el MVC divide los elementos de un sistema en tres partes, de manera que cada parte puede ser tratada de forma independiente. Para que el elemento se comporte como un todo, cada parte debe tener una interfaz hacia las otras dos, es decir, la vista es capaz de enviar mensajes al controlador y obtener información del modelo para poder realizar sus tareas.

A continuación, en la figura 2.1 se muestra el diagrama de componentes del patrón MVC.

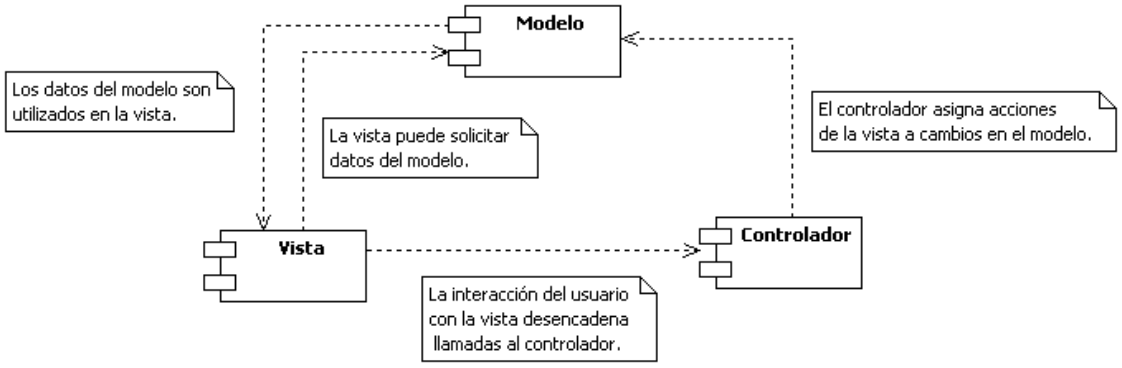


Figura 2.1 Componentes de MVC

La implementación del MVC, requiere de los siguientes componentes, representados en la figura 2.2:

- *Model* (modelo). Es el componente que contiene una o más clases e interfaces que son los responsables de mantener los datos del modelo. Se encarga de:
  - Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
  - Definir las reglas de negocio.

- *View* (vista): Las clases e interfaces de la vista proporcionan una representación de los datos en el componente del modelo. La vista puede consistir de componentes visuales, por ejemplo pantallas, aunque no necesariamente. Es responsable de:
  - Recibir datos del modelo y presentarlos al usuario.
  - Mantener un registro de su controlador asociado.

- *Controller* (controlador). Este componente administra los cambios en el modelo y las peticiones provenientes de la vista. Es responsable de:
  - Recibir los eventos de entrada (un clic, un cambio en un campo de texto).
  - Contener las reglas para la administración de eventos, es decir las reglas que tienen que ver con la funcionalidad del sistema. Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "actualizar()". Una petición al modelo puede ser "obtenerRegistroAlumno(idAlumno)".

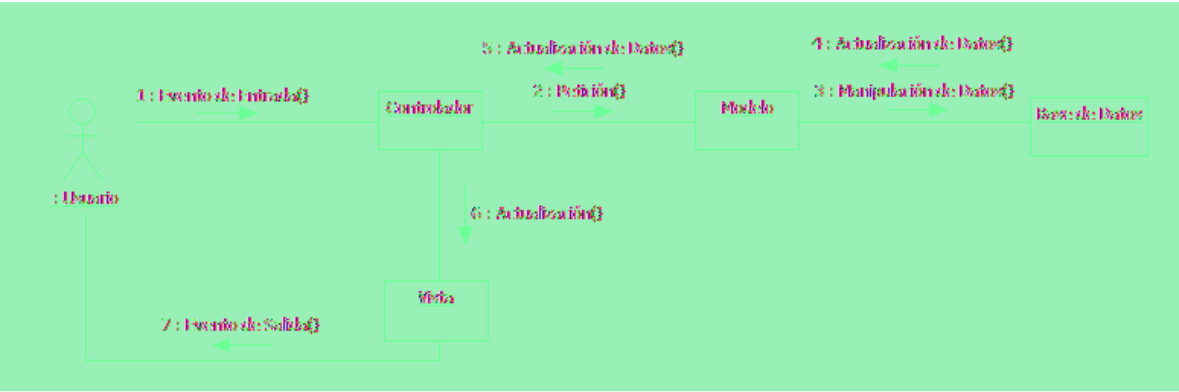


Figura 2.2 Interacción de los componentes MVC

2.2.3. Framework Struts

A continuación se explicará la estructura de Struts, que fue elegido para la implementación de la solución del problema que se plantea en este escrito, debido a que es una de las herramientas más poderosas en el desarrollo de aplicaciones Web.

Struts es un framework que implementa el patrón de arquitectura MVC en Java, así que comenzaremos por explicar que un framework es la extensión de un lenguaje mediante una o más jerarquías de clases que efectúan una funcionalidad, es decir es un conjunto de

herramientas y componentes que pueden ser usadas directamente o bien que pueden ser extendidas para crear componentes propios. Básicamente Struts permite separar fácilmente la capa de presentación de la de negocio. Struts ofrece su propio controlador el cual es un servlet llamado ActionServlet y permite hacer una integración con otras tecnologías para implementar el modelo y la vista. Para el modelo, Struts puede interactuar con tecnologías estándar de acceso a datos, como JDBC y EJB, así como con la mayoría de componentes de terceros, como Hibernate e iBatis, mientras que en el caso de la vista, Struts generalmente se utiliza con JSP.

A continuación se explica brevemente el funcionamiento de Struts:

En una aplicación basada en Struts, todas las peticiones de cliente son enviadas a un único controlador, en este caso es una clase que extiende de ActionServlet. El controlador carga un archivo de configuración al inicializarse, struts-config.xml, en él se definen tanto aspectos generales de toda la aplicación como detalles del flujo de una parte concreta de la navegación.

El controlador decidirá en función de los mapeos declarados en este archivo de configuración, a que clase Action redirigir la petición del cliente.

Si la petición del cliente viene con parámetros, Struts encapsula estos parámetros en un bean ActionForm. También se pueden utilizar los bean ActionForm para inicializar un formulario con valores, los ActionForm se apoyan de un archivo llamado validation.xml, el cual es configurable de acuerdo a los datos que se tengan en éste para que se lleven a cabo validaciones de los datos.

Así mismo, el framework Struts ofrece un conjunto de taglib para trabajar con vistas JSP, integradas con el resto del framework. También ofrece integración con el framework Validator, para validaciones fuera de código, definidas en archivos de propiedades e integración con el framework Tiles, para la composición de páginas JSP.

Soporta internacionalización del contenido a mostrar a través de archivos resource boundles, donde se pueden definir todos aquellos mensajes, etiquetas, títulos o textos susceptibles de ser internacionalizados.

Podemos separar las responsabilidades de Struts, en función de cada uno de los elementos del patrón de diseño MVC, tal como se representa en la figura 2.3.

Las responsabilidades del Controlador serán:

- Recibir las peticiones del cliente.
- Mapear acciones del usuario a actualizaciones del modelo de negocio.
- Determinar que vista en función de un resultado hay que mostrar.

Las responsabilidades del Modelo serán:

- Implementar la lógica de negocio de la aplicación.
- Encapsular el estado de la aplicación.
- Notificar cambios en las vistas.

Las responsabilidades de la Vista serán:

- Generar la interfaz gráfica de la aplicación.
- Visualizar los resultados del modelo.
- Enviar peticiones del cliente al controlador.

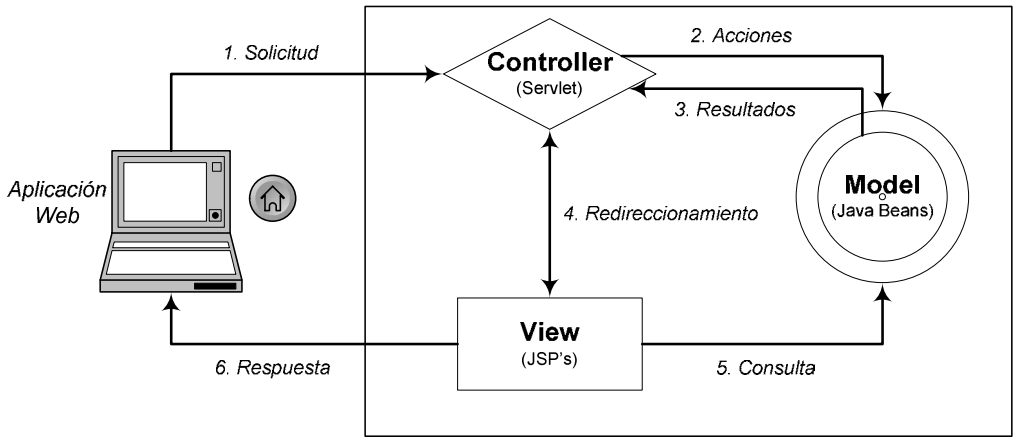


Figura 2.3 Interacción de los componentes de Struts

Es importante mencionar que Struts está disponible bajo la licencia "free-to-use-license" de la Apache Software Foundation.

2.3. Bases de Datos

Conceptos Básicos

Un sistema manejador de bases de datos (DBMS) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, generalmente se denomina base de datos y contiene información relevante para un negocio.

El objetivo principal de un DBMS es proporcionar una forma de almacenar y recuperar información de una base de datos de manera que sea práctica y eficiente.

Los sistemas de bases de datos se diseñan para administrar grandes cantidades de información, dicha administración implica tanto la definición de estructuras para almacenar la información como la creación de mecanismos para la manipulación de dicha información.

Antes de presentarse el uso de los DBMS, se almacenaba la información en archivos, de manera que un sistema podía almacenar los registros permanentes en varios archivos y requería de diferentes programas para extraer información y añadirla a los archivos correspondientes.

Como es de suponerse, guardar la información en un sistema de procesamiento de archivos tiene una serie de inconvenientes importantes, por ejemplo:

- Redundancia e inconsistencia de los datos.
- Dificultad en el acceso a los datos.
- Aislamiento de datos.
- Problemas de integridad.
- Anomalías en el acceso concurrente.
- Problemas de seguridad.

Estas dificultades, entre otras, dieron pie al desarrollo de sistemas de bases de datos.

A continuación se explicaran los conceptos más importantes referentes a las bases de datos.

Como se mencionó anteriormente, un sistema de base de datos es un conjunto de datos interrelacionados y un conjunto de programas que permiten tener acceso a dichos datos y modificarlos. Una de las principales finalidades de la base de datos es ofrecer a los usuarios una visión abstracta de los datos de negocio, es decir, el sistema oculta ciertos detalles del modo en que se almacenan y mantienen los datos.

Para que un sistema sea útil, debe recuperar los datos eficientemente. La necesidad de eficiencia ha llevado a los diseñadores a usar estructuras de datos complejas para representar los datos y esta complejidad se maneja mediante varios niveles de abstracción para simplificar la interacción de los usuarios con el sistema, como son:

- Nivel Físico. Es el nivel más bajo de abstracción, describe como se almacenan realmente los datos, describe en detalle las estructuras de datos.

- Nivel Lógico. Es el nivel superior al nivel físico, describe que datos se almacenan en la base de datos y que relaciones existen entre ellos. En otras palabras, este nivel representa las entidades de negocio.

- Nivel de Vistas. Este es el nivel más elevado de abstracción, sólo describe parte de la base de datos. Debido a que muchos usuarios del sistema de bases de datos no necesitan toda la información y en su lugar sólo necesitan tener acceso a una parte de la base de datos, se crean vistas y este nivel de abstracción permite simplificar la interacción de los usuarios con el sistema.

Existen tres modelos para el tratamiento de la información:

a) *Modelo Jerárquico:* Permite representar relaciones de tipo uno a muchos. Es un modelo muy rígido en el que las diferentes entidades de las que está compuesta una determinada situación, se organizan en niveles múltiples de acuerdo a una estricta relación padre/hijo, de manera que un padre puede tener más de un hijo, todos ellos localizados en el mismo nivel y un hijo únicamente puede tener un padre situado en el nivel inmediatamente superior al suyo.

Este modelo tiene forma de árbol invertido en el que una rama puede tener varios hijos. Es un modelo simple y fácil de utilizar sin embargo puede presentar problemas en las operaciones de inserción, borrado o actualización que se deseen realizar.

b) *Modelo de Red*. Esta estructura abarca más que la estructura de árbol, porque un nodo hijo en la estructura de red puede tener más de un nodo padre. Este modelo permite la representación de muchos a muchos. El modelo de red evita redundancia en la información, a través de la incorporación de un tipo de registro denominado el conector. No es un modelo muy utilizado debido a que su mantenimiento es difícil ya que no es tan simple definir nuevas relaciones.

c) *Modelo Relacional*. Desde los años 80 es el modelo más utilizado, ya que permite una mayor eficacia, flexibilidad y confianza en el tratamiento de los datos. En el modelo relacional se representa el mundo real mediante tablas relacionadas entre sí por columnas comunes, su estructura principal es la relación. Cada tupla, representa una entidad que se desea almacenar en la base de datos.

   Las características de cada entidad están definidas por las columnas de las relaciones, que se llaman atributos. Las entidades con características comunes, es decir descritas por el mismo conjunto de atributos, formarán parte de la misma relación.

   En resumen el modelo relacional, se puede representar mediante un conjunto de tablas de dos dimensiones (relaciones), con tuplas (renglones) y atributos (columnas) que tienen integridad de datos, es decir los datos son precisos y consistentes.

2.4. Tecnologías

2.4.1. JAVA

Características de lenguaje

Java es una plataforma de software desarrollada por Sun Microsystems, los programas creados sobre ella, pueden ejecutarse sin cambios, en diferentes tipos de arquitecturas y dispositivos de cómputo.

Java llegó a ser muy popular cuando Sun Microsystems introdujo la especificación J2EE (Java 2 Enterprise Edition). Este modelo permite una separación entre la presentación de los datos al usuario (JSP o Applets), el modelo de datos (EJB) y el control (Servlets).

Java es un lenguaje orientado a objetos diseñado para ser multiplataforma y poder ser empleado el mismo programa en diversos sistemas operativos. Esta característica, junto con la posibilidad de emplearlo para crear applets e insertarlos en páginas HTML, o mediante servlets y páginas JSP generar código HTML dinámico, así como la capacidad de acceder a bases de datos, hace de JAVA uno de los lenguajes más utilizados en la actualidad.

Java es un lenguaje relativamente sencillo, debido a que prácticamente toda la funcionalidad se encuentra en clases que forman parte del API de Java. Es un lenguaje libre, ya que se puede utilizar el compilador y la máquina virtual de forma gratuita.

Cabe mencionar que un API (Application Programming Interface - Interfaz de Programación de Aplicaciones) es un conjunto de especificaciones de comunicación entre componentes software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla.

En resumen, las características más importantes de Java son:

- Lenguaje Orientado a Objetos. Su concepción es muy próxima a la forma de pensar humana. Implementa los conceptos de POO, tales como:

  - ✓ Herencia
  - ✓ Encapsulamiento
  - ✓ Polimorfismo
  - ✓ Abstracción
  - ✓ Reutilización
  - ✓ Jerarquía
  - ✓ Modularidad

- Distribuido y dinámico. Es un lenguaje orientado para trabajar en Red.

- Seguro. La máquina virtual, al ejecutar el código Java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los apuntadores. Facilidad para implementar la autenticación, autorización y encriptación para proteger la privacidad y asegurar la integridad de los datos

- Compilado e Interpretado. La mayoría de los lenguajes de programación se caracterizan por ser interpretados o compilados, lo que determina la manera en como serán ejecutados en una computadora.

  Java tiene la característica de ser al mismo tiempo compilado e interpretado. El compilador es el encargado de convertir el código fuente de un programa en un código intermedio llamado bytecode que es independiente de la plataforma en que se trabaje y que es ejecutado por el intérprete de Java que forma parte de la Máquina Virtual de Java. En la figura 2.4 se representa el esquema general de Java.

- Portable. El mismo código Java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual Java.

- Multitareas. Java es un lenguaje que soporta múltiples hilos o procesos o tareas, también llamados threads, es decir puede ejecutar diferentes líneas de código al mismo tiempo.

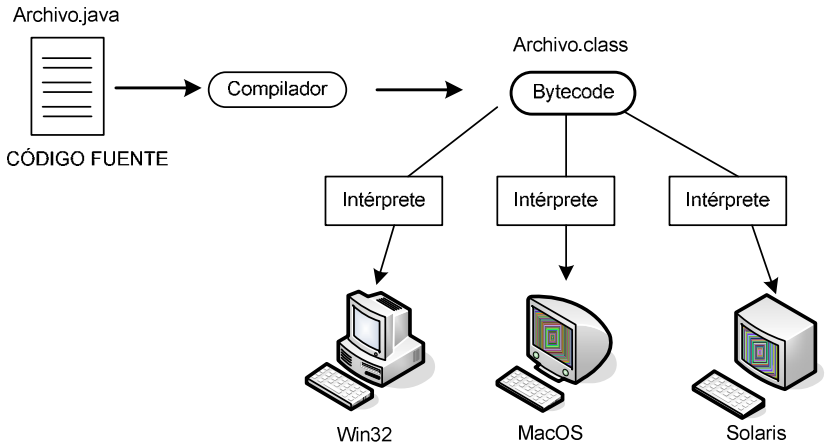


Figura 2.4 Esquema general de JAVA

La plataforma Java

Una plataforma es el ambiente de hardware o software en el cual se ejecutan los programas.

En general, la mayoría de las plataformas pueden ser descritas como una combinación de hardware y sistema operativo. Algunas de las plataformas más populares son Windows, Solaris, Linux y MacOS.

La plataforma Java difiere de las anteriores en que ésta es una plataforma basada únicamente en software que corre por encima de las plataformas basadas en hardware. La representación de dicha plataforma se puede apreciar en la figura 2.5.

La plataforma Java consta de dos componentes:

- La Máquina Virtual de Java (JVM) que permite la portabilidad en ejecución
- La Interfaz de Programación de Aplicaciones de Java (API Java), una biblioteca estándar para el lenguaje.

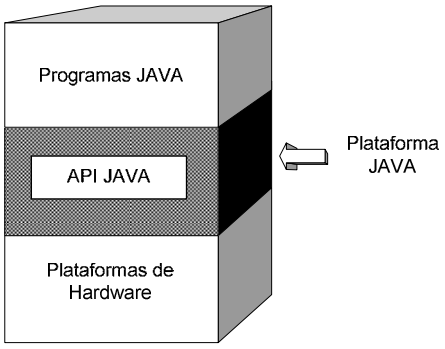


Figura 2.5 Plataforma JAVA

Tipos de programas en Java.

Los programas en Java suelen estar en una de las siguientes categorías:

- *Applets*
  Los applets son pequeños programas que se incorporan en una página Web y que por lo tanto, necesitan de un Navegador Web compatible con Java para poder ejecutarse. A menudo los applets se descargan junto con una página HTML desde un Servidor Web y se ejecutan en la máquina cliente.

- *Aplicaciones*
  Las aplicaciones son programas independientes de propósito general que normalmente se ejecutan desde la línea de comandos del sistema operativo. Con Java se puede realizar cualquier programa que normalmente se crearía con algún otro lenguaje de programación.

- *Servlets*
  Los servlets al contrario de los applets son programas que están pensados para trabajar en el lado del servidor y desarrollar aplicaciones Web que interactúen con los clientes. Los servlets son una alternativa de la programación CGI tradicional.

2.4.2. Servlet y JSP

Los servlets y JSP forman parte de J2EE (Java 2 Enterprise Edition). Su nombre significa Server-Side Entity, una entidad u objeto, que se ejecuta del lado del servidor. Un servlet es una clase que se utiliza principalmente para extender las capacidades de un servidor, en aplicaciones que son solicitadas a través de un modelo request-response. Un request es una solicitud, echa por una persona u otro sistema, a través de un protocolo TCP/IP. El response, a la inversa, es la

respuesta del servidor a esta petición, enviada a quien lo solicito por el mismo protocolo solicitado.

A través de los servlets se pueden responder a cualquier tipo de request. El protocolo más común para realizar esas peticiones es el de HTTP.

El ciclo de vida de un servlet es el siguiente:

- Un servidor carga e inicializa el servlet.
- El servlet maneja cero o más peticiones de cliente.
- El servidor elimina el servlet.

Posterior a la creación de los servlets, nacieron los Java Server Pages o JSP. Es la tecnología que nos permite combinar HTML con programación Java, para generar HTML dinámicamente.

Una página Web dinámica consiste en un lenguaje de etiquetas combinadas con algún lenguaje de programación, en la cual cuando un cliente hace un requerimiento, el contenido de una página se crea dinámicamente y se le presenta al usuario. Esta metodología se llama Server-Side Include.

Existen dos tipos de arquitecturas para construir aplicaciones basadas en JSP y servlets. Esos tipos de arquitectura son llamadas “Modelo 1 de JSP” y “Modelo 2 de JSP”.

Modelo 1

En el modelo 1, representado en la figura 2.6, básicamente los JSP remplazan a los servlets. Toda la funcionalidad se realiza en los JSP: reciben el requerimiento, crean los java beans, se conectan a la base de datos, arman la respuesta y finalmente, despliegan la vista.

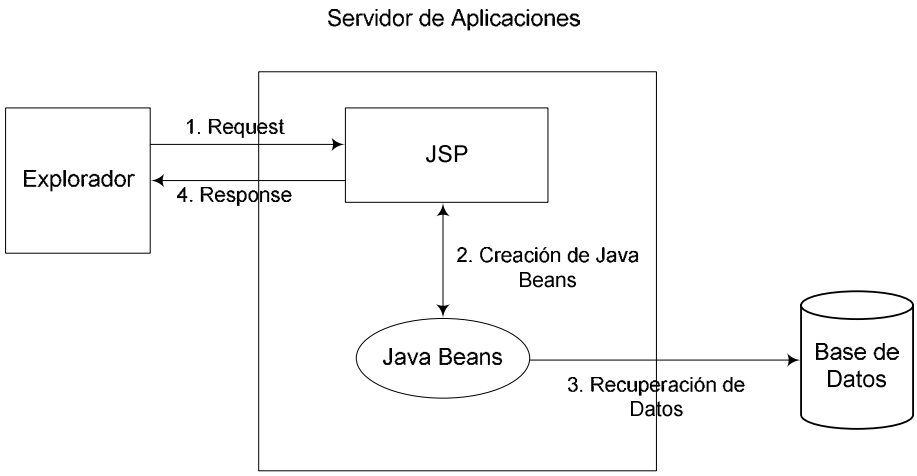


Figura 2.6 Modelo 1 de JSP

Modelo 2

Bajo esta arquitectura conviven JSP y servlets, los JSP, se encargan de la parte visual y los servlets controlan la parte lógica de la presentación y la comunicación con la capa de negocios.

En la figura 2.7 se representa el flujo que se sigue en el modelo 2 o MVC. El requerimiento de un cliente llega a un servlet que actúa como controlador. El servlet crea los java beans, se comunica con la capa de lógica de negocio, y ésta, a su vez, interactúa con base de datos. Finalmente arma la respuesta e invoca al JSP que corresponde para contestarle al cliente.

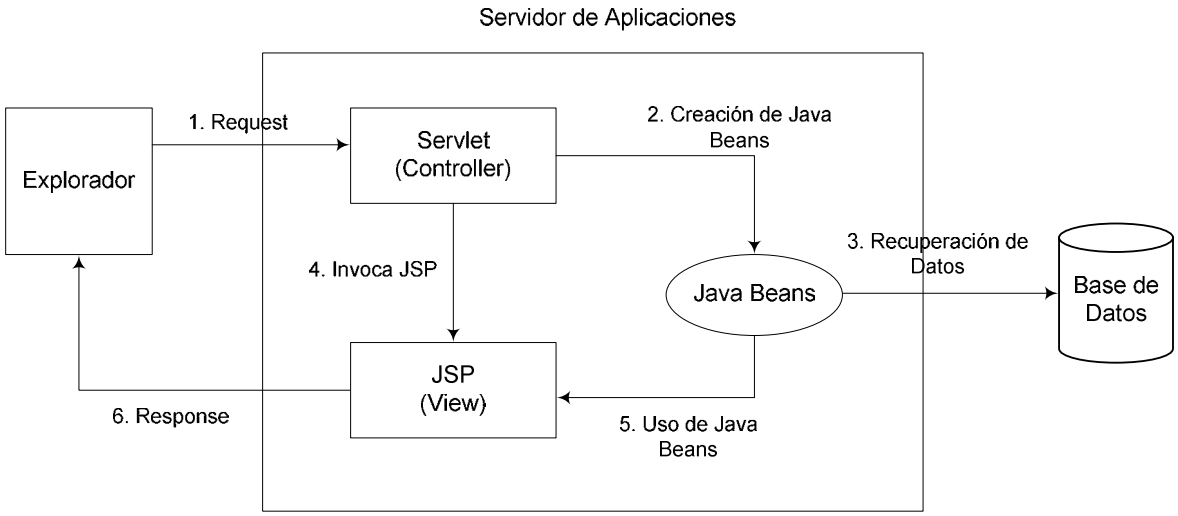


Figura 2.7 Modelo 2 de JSP

2.5. Metodología RUP

2.5.1. Conceptos Básicos

Los procesos de desarrollo son una manera de trabajar eficientemente para evitar problemas que pueden provocar que un proyecto de software termine sin éxito.

El objetivo de un proceso de desarrollo es subir la calidad del software (en todas las fases por las que pasa) por medio de un control sobre dicho proceso. Uno de los procesos de desarrollo más famosos es el Proceso Unificado de Rational (RUP por sus siglas en inglés).

RUP es uno de los procesos más generales que existen actualmente, esta pensado para adaptarse a cualquier proyecto y no únicamente de software.

Un proyecto siguiendo RUP se divide en cuatro fases:

a) Inicio (Puesta en marcha)
b) Elaboración (Definición, Análisis, Diseño)
c) Construcción (Implementación)
d) Transición (Fin del proyecto y puesta en producción)

a) **Fase de Inicio**. Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos potenciales asociados al proyecto, proponer una visión muy general de la arquitectura de software.

Durante esta fase se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y casos de uso. Se desarrolla un plan de negocio para determinar que recursos deben ser asignados al proyecto.

Los objetivos de la fase están enfocados a:

- ✓ Establecer el ámbito del proyecto y sus límites.
- ✓ Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- ✓ Mostrar al menos una arquitectura candidata para los escenarios principales.
- ✓ Estimar el costo en recursos y tiempo de todo el proyecto.
- ✓ Estimar los riesgos y las fuentes de incertidumbre.

El hito en esta fase finaliza con el establecimiento del ámbito del producto e identificación de los principales riesgos y la viabilidad del proyecto.

b) **Fase de 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.

El propósito de esta fase es plantear la arquitectura para el ciclo de vida del producto. Se construye un modelo de la arquitectura, que se desarrolla en iteraciones sucesivas hasta obtener el producto final, este prototipo debe contener los casos de uso críticos que fueron identificados en la fase de inicio. En esta fase se realiza la captura de la mayor parte de los requerimientos funcionales, manejando los riesgos que interfieran con los objetivos del sistema, acumulando la información necesaria para el plan de construcción y obteniendo suficiente información para hacer realizable el caso del negocio. Por lo tanto los esfuerzos de esta fase están enfocados a:

- ✓ Completar requerimientos.
- ✓ Crear un plan para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones, (debe incluir los costos si procede).
- ✓ Mostrar al menos una arquitectura candidata para los escenarios principales.

El hito en la fase de elaboración finaliza con la obtención de una línea base de la arquitectura del sistema, la captura de la mayoría de los requerimientos y la reducción de los riesgos importantes así como permitir la escalabilidad del equipo del proyecto durante la fase de construcción.

c) **Fase de Construcción**. El objetivo de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar los cambios de acuerdo a las evaluaciones realizadas por los usuarios además de implementar las mejoras para el proyecto.

En esta fase se debe alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Todas las características, componentes y requerimientos deben ser integrados, implementados y probados en su totalidad, obteniendo una versión aceptable del producto comúnmente llamada versión beta. Se hace énfasis en controlar las operaciones realizadas, administrando los recursos eficientemente, de tal forma que se

optimicen los costos, los calendarios y la calidad. Los objetivos de la fase se pueden resumir en:

- ✓ Minimizar los costos de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- ✓ Definir, validar y establecer la arquitectura.
- ✓ Conseguir una calidad adecuada tan rápido como sea práctico.
- ✓ Conseguir versiones funcionales (alfa, beta y otras versiones de prueba) tan rápido como sea práctico.

El hito en esta fase culmina con el desarrollo del sistema con calidad de producción y la preparación para la entrega al equipo de transición. Toda la funcionalidad debe haber sido implementada y las pruebas para el estado beta de la aplicación deben estar completadas. Si el proyecto no cumple con estos criterios de cierre, entonces la transición deberá posponerse una iteración.

d) **Fase de Transición**. El propósito de esta fase es asegurar que el software este 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.

Se debe entregar el producto funcional en manos de los usuarios finales una vez realizadas las pruebas de aceptación por un grupo especial de usuarios, para lo que se requerirá desarrollar nuevas versiones actualizadas del producto, también se requiere entrenar a los usuarios en el manejo del sistema, completar la documentación y en general realizar las tareas relacionadas con la configuración, instalación y usabilidad del producto. El objetivo de esta fase es:

- ✓ Conseguir un producto final que cumpla los requerimientos esperados.

El hito en la fase de transición corresponde a haber decidido si los objetivos se cumplieron y el comienzo de otro ciclo de desarrollo. El cliente debe haber revisado y aceptado los artefactos que le han sido entregados.

La figura 2.8, ilustra la arquitectura general de RUP, la cual tiene dos dimensiones:

- El eje horizontal representa el tiempo y muestra el ciclo de vida de los aspectos del proceso que se desarrolla.
- El eje vertical representa las actividades a desarrollar.

La gráfica muestra como el énfasis de las disciplinas varia a través del tiempo. Por ejemplo, en las primeras iteraciones se consume más tiempo en los requerimientos, mientras que en iteraciones posteriores se gasta más tiempo en la implementación.

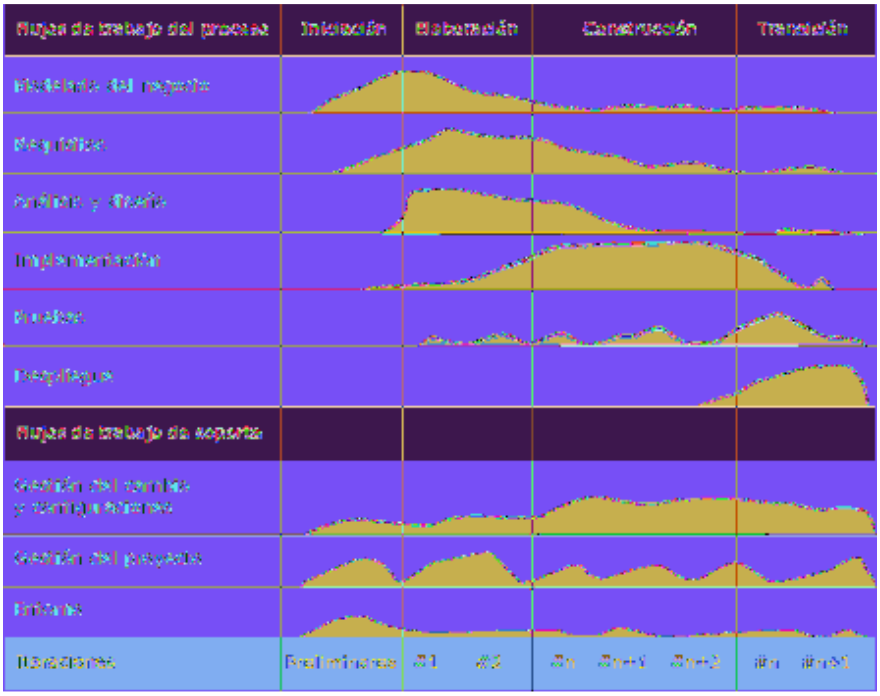


Figura 2.8 Metodología RUP

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable, según el proyecto) y dentro de cada una de ellas se seguirá un modelo de cascada para los flujos que requieren las nuevas actividades anteriormente citadas.

RUP define nueve actividades a realizar en cada fase del proyecto:

1. Modelado del negocio
2. Análisis de requerimientos
3. Análisis y diseño
4. Implementación
5. Pruebas
6. Distribución
7. Administración de configuración y cambios
8. Administración del proyecto
9. Administración del entorno.

Así mismo el proceso define una serie de roles que se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado que se espera de ellos.

- Analistas
- Desarrolladores
- Roles Generales
- Administradores
- Producción y Soporte
- Testers

RUP se basa en los casos de uso para describir lo que se espera del software y esta orientado a la arquitectura del sistema, documentándose lo mejor posible, basándose en UML como herramienta principal.

Cabe mencionar que RUP esta pensado para aplicarse en proyectos y equipos grandes, en cuanto a duración y tamaño.

Con RUP se presentarán al cliente los artefactos al final de una fase y se valoraran las precondiciones para la siguiente. Entre las ventajas que ofrece este proceso están, el nivel de organización y la documentación que se genera durante el desarrollo, puesto que se permite crear una base de conocimientos para desarrollos posteriores.

2.6. UML

2.6.1. Introducción a UML

Las iníciales UML son acrónimo de Unified Modeling Language (Lenguaje Unificado de Modelado). Es importante subrayar que UML es un lenguaje y como tal es una herramienta para expresar ideas, un lenguaje específico no limita los tipos de ideas ni la manera en que pueden describirse mediante dicho lenguaje. En otras palabras UML es un lenguaje de modelado que provee un mecanismo o notación para expresar y comunicar, mediante un modelo, los elementos necesarios para el análisis, diseño y desarrollo de un sistema de información, basados en los requerimientos que el usuario o el entorno establecen para el mismo.

UML puede usarse para describir los sistemas de información desarrollados mediante el paradigma orientado a objetos o bien sobre el proceso unificado. UML es una notación, no una metodología.

Esta notación se deriva y unifica de tres metodologías de análisis y diseño orientado a objetos, como son:

- Metodología de Grady Booch para la descripción de conjuntos de objetos y sus relaciones.
- Técnica de modelado orientada a objetos de James Rumbaugh (OMT: Object-Modeling Technique).
- Aproximación de Ivar Jacobson (OOSE: Object- Oriented Software Engineering) mediante la metodología de casos de uso.

UML es el lenguaje de modelado estándar para crear planos de software, permite:

- Especificar modelos precisos, completos y sin ambigüedad.
- Documentar arquitectura, requerimientos, pruebas, procesos de negocio, páginas Web.
- Construir código base mediante herramientas como el Rational Rose en varios lenguajes de programación.

Beneficios de UML

a) Es un estándar en la industria de construcción de software.
b) Permite modelar y documentar la arquitectura de una aplicación.
c) Define una notación expresiva y consistente.
d) Facilita la comunicación con otros.
e) Permite detectar omisiones o inconsistencias.
f) Es aplicable a sistemas sencillos y complejos.
g) Existen herramientas en el mercado para modelar y generar código a partir de UML.
h) Provee beneficios significativos para los ingenieros de software y las organizaciones al ayudarles a construir modelos rigurosos, trazables y fáciles de mantener, que soporten el ciclo de vida de desarrollo de software completo.

2.6.2. Modelos de UML y clasificación

a) Requerimientos

Lo más importante antes de comenzar cualquier análisis y/o desarrollo, es establecer los requerimientos del sistema. De manera general un requerimiento es una condición o característica que debe satisfacerse. Existen dos tipos de requerimientos:

- Funcionales: Describen lo que el sistema debe hacer, en términos de operatividad, por ejemplo en un sistema de administración de saldos de tarjetas de crédito, un requerimiento funcional podría ser: calcular intereses mensuales.

- No funcionales: Describen atributos del sistema o atributos del ambiente del sistema por ejemplo tiempo de respuesta, capacidad de almacenamiento de información, concurrencia de usuarios, entre otros.

El artefacto de UML que nos permite modelar los requerimientos funcionales son los casos de uso. De manera técnica, un caso de uso es uno de los servicios que ofrece un sistema a un actor, es una interacción entre un usuario (actor) y un sistema. Un diagrama de casos de uso modela las posibles formas en que un sistema puede ser usado.

b) Modelo Estático

El modelo estático refleja la estructura estática de las clases de objetos y sus relaciones. Se compone por los siguientes diagramas:

- Diagrama de Clases. Describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

- Diagrama de Componentes. Un diagrama de componentes representa como un sistema de software es dividido en elementos y muestra las dependencias entre

dichos elementos. Los componentes físicos incluyen archivos, librerías, módulos, archivos ejecutables o paquetes. Los diagramas de componentes pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

- Diagrama de Despliegue. Se utiliza para modelar el hardware utilizado en la implementación de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones.

c) Modelo Dinámico

- Diagrama de Secuencia. Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. Contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes pasados entre los objetos.

- Diagrama de Colaboración. Modela las interacciones entre objetos o partes en términos de mensajes en secuencia. Representan una combinación de información tomada desde el diagrama de clases, secuencia, y diagrama de casos de uso describiendo tanto la estructura estática como el comportamiento dinámico de un sistema.

- Diagrama de Estados. Permite reflejar los estados por los cuales puede pasar un objeto, entendiendo que un estado es una condición durante la vida de un objeto o una interacción durante la cual se satisface alguna condición, se realiza alguna acción o se espera algún evento.

- Diagrama de Actividades. Este diagrama representa los flujos de negocio paso a paso de y operacionales de los componentes en un sistema. Un diagrama de actividades muestra el flujo de control general. Así mismo permite diferenciar claramente que actor desarrolla cada una de las actividades por medio de la utilización de carriles.

3. Inicio

De acuerdo a la metodología RUP en esta fase se enfocan las actividades a comprender la estructura y dinámica de la organización, así como los problemas actuales para efectuar la identificación de mejoras. Para poder llevar a cabo lo anterior, la metodología plasma este análisis en los casos de uso y diagramas de actividades.

Una vez que se ha definido el proceso actual del negocio, se analizan las necesidades de éste y se define una solución, identificando las características de la solución, definiendo actores y casos de uso, describiendo requerimientos funcionales, no funcionales, se acuerda el alcance de la solución; estableciendo lo que el sistema tiene que hacer y se definen los límites del mismo. Por otro lado se establece el alcance del proyecto elaborando la estimación esfuerzo, el cronograma de trabajo y se establece el presupuesto, el cual incluye el tiempo de desarrollo para analizar la viabilidad en términos económicos y se elabora el plan integral del proyecto, que puede ir cambiando a lo largo del proyecto conforme se van cerrando las fases.

En esta fase es importante establecer los criterios técnicos de selección de soluciones, para ello se evaluaron las alternativas contra criterios técnicos lo cual permitió desarrollar una solución conceptual.

Cabe mencionar que a lo largo del proyecto se desarrollan las disciplinas de administración del proyecto, administración y configuración del cambio, así como ambiente.

El propósito de la administración del proyecto es:

- Proporcionar una estructura para administrar proyectos de software.
- Proveer prácticas para la planeación, la administración de recursos, la ejecución y el monitoreo de los proyectos.
- Proporcionar un marco para el manejo de riesgos.

Como disciplina en RUP, se centra principalmente en los aspectos importantes de un proceso iterativo como:

- El control de riesgos.
- La planificación de los ciclos de un proyecto iterativo.
- El seguimiento de los progresos de un proyecto iterativo, así como las métricas.

Las principales tareas de la disciplina son:

1. Concebir un nuevo proyecto
2. Evaluar el alcance y el riesgo del proyecto
3. Planear el proyecto
4. Monitorear y controlar el proyecto
5. Cerrar las fases
6. Cerrar el proyecto

La disciplina configuración y administración del cambio tiene como propósito proporcionar las bases para saber cómo controlar y sincronizar la evolución del conjunto de productos del trabajo (artefactos generados) que componen un sistema de software.

El propósito de la disciplina es:

- Analizar las solicitudes de cambio
- Realizar cambios
- Verificar solicitud de cambio
- Dar seguimiento a solicitudes de cambio
- Notificar solicitud de cambio

Es común que en un proyecto se hagan actualizaciones constantes de entregables por muchas personas, es por ello que tener un control sobre estos artefactos ayuda a disminuir costosas confusiones y asegura que los productos de trabajo resultantes no estén en conflicto debido a actualizaciones simultáneas, múltiples versiones o bien cuando se realizan cambios por algún miembro del equipo y éste no notifica al resto de sus compañeros.

Por otro lado la disciplina ambiente, controla los elementos que proporcionan el entorno de desarrollo de software, incluidos los procesos y herramientas.

El propósito de la disciplina de Ambiente es:

- Establecer repositorios para la información.
- Planificar configuración del proyecto.
- Definir necesidades de capacitación.

En los **Anexos A, B** y **C**, se presentan los siguientes documentos:

a) Estimación total de Esfuerzo y Costo.
b) Plan de Trabajo del Proyecto.
c) Bitácora de Riesgos.

3.1. Modelado de Negocio

El Modelado de Negocio se efectúa para estudiar el negocio para construir un sistema de información y para determinar mejor las necesidades y problemáticas a ser resueltas por dicho sistema.

Las principales actividades de la disciplina de Modelado de Negocio son:

- Entender la estructura y la dinámica de la organización para la que el sistema será desarrollado.
- Entender el problema actual en la organización objetivo e identificar potenciales mejoras.
- Asegurar que el producto será algo útil, no un obstáculo.
- Conseguir que el producto encaje de la mejor forma posible en la organización.

A continuación se describe el modelo de negocio de DGSCA.

3.1.1. Flujo de Negocio

DGSCA cuenta con programas de capacitación a becarios y acepta alumnos de servicio social en distintas sedes que se encargan de la formación o asignación de alumnos a proyectos en áreas como tecnologías de información, telecomunicaciones y sistemas, entre otras, sin embargo, la información de los alumnos adscritos es concentrada en la sede de Ciudad Universitaria, debido a esto se maneja una gran cantidad de datos, aproximadamente se reciben 200 solicitudes de ingreso cada semestre.

En este apartado se describe el proceso, el cual de manera general, inicia cuando una persona llena una solicitud de ingreso, posteriormente es aceptada, se integra a un proyecto y se le da seguimiento a sus actividades. Para un mejor entendimiento, el proceso se explica de la siguiente manera:

Solicitud de Ingreso

Cuando un alumno solicita ingresar a un nuevo servicio como: servicio social o plan de becas, llena una forma impresa que contiene:

a) Datos personales como:

- Nombre
- Dirección
- Fecha de nacimiento
- Teléfono
- Correo electrónico
- Sexo

b) Datos escolares como:

- Escuela o facultad
- Número de cuenta
- Carrera
- Promedio
- Semestre que cursa
- Porcentaje de créditos acumulados
- Horario disponible para asistir a DGSCA

c) Currículo

- Idiomas que maneja junto con el nivel que habla, lee y traduce para cada uno,
- Lenguajes de programación que maneja

- Paquetería la cual está clasificada en:
  - ✓ Procesador de palabras
  - ✓ Hojas de cálculo
  - ✓ Estadísticos
  - ✓ Bases de datos
  - ✓ Graficación
- Experiencia académica y profesional

Dicha solicitud es entregada en las oficinas de Servicio Social de DGSCA con la finalidad de que sea evaluada por el Responsable de Servicio Social.

Registro para Servicio Social

Después de que se entrega la solicitud, el encargado de Servicio Social revisa la información, de acuerdo al perfil y experiencia del solicitante, busca un área donde pueda asignar al alumno, una vez que sabe donde podría asignarlo se pone en contacto con quién sería su jefe inmediato para acordar si proceden a llamarlo. Una vez que acordaron, el encargado de servicio social, llama al alumno para entrevistarlo. Cabe mencionar que dependiendo del área se pueden aplicar algunos exámenes. Después de este proceso, el alumno es aceptado y asignado a colaborar en el departamento de su jefe inmediato en alguno de los proyectos a su cargo.

Para generar el registro del servicio, el encargado de Servicio Social completa la solicitud entregada por el alumno donde se indica:

- ✓ Tipo de servicio
- ✓ Fecha de ingreso
- ✓ Fecha de egreso
- ✓ Horario
- ✓ Departamento
- ✓ Nombre del Responsable directo con quién estará colaborando
- ✓ Firma del responsable
- ✓ Teléfono del departamento
- ✓ Observaciones respecto al alumno

En base a la información anterior se elabora una carta de aceptación para el alumno, que contiene una descripción de las actividades que deberá realizar durante su servicio de acuerdo a los programas que se tienen registrados en la Dirección General de Orientación de Servicios Educativos (DGOSE), dependiendo de la carrera.

Registro para Planes de Beca

Existe otra modalidad de ingreso a DGSCA y ésta es por medio de los diferentes planes de becas que existen tales como:

- ✓ Cómputo de alto rendimiento
- ✓ Sistemas
- ✓ Telecomunicaciones
- ✓ Docencia en cómputo

Para estos planes se emiten convocatorias, en éstas los alumnos interesados además de llenar su solicitud, entregan documentos específicos que requiere cada área según el plan. Posterior a este proceso, se realiza una etapa de selección que depende del área dónde se desee ingresar. Si los alumnos son aceptados entran a una etapa de capacitación y posteriormente se integran a un proyecto.

Becarios

Por otro lado dependiendo del tiempo que lleven colaborando en DGSCA, los alumnos pueden ser candidatos a obtener una beca económica; el nivel de beca se otorga de acuerdo al grado académico del alumno entre los que están:

- ✓ Bachillerato
- ✓ Bachillerato Técnico
- ✓ Licenciatura
- ✓ Postgrado

Para solicitar una de beca, se manejan dos convocatorias, una que va en un periodo ordinario de enero a junio y de julio a diciembre, otra en un periodo extraordinario que va de abril a junio y de octubre a diciembre, el alumno debe llenar una solicitud, con los siguientes datos:

a) Datos personales
   - ✓ Nombre
   - ✓ Dirección
   - ✓ Teléfono
   - ✓ Correo electrónico
   - ✓ Fecha de nacimiento
   - ✓ CURP

b) Datos escolares
   - ✓ Escuela o facultad
   - ✓ Número de cuenta
   - ✓ Carrera
   - ✓ Promedio
   - ✓ Semestre que cursa
   - ✓ Porcentaje de créditos acumulados

c) Antecedentes del prestador en DGSCA
   - ✓ Tiempo que lleva en servicio de apoyo o tiempo en servicio social
   - ✓ Nivel de beca anterior si es que la hubo
   - ✓ Horario en el que acude a DGSCA

d) Otras Remuneraciones (si recibe o tiene otro ingreso económico)
   - ✓ Institución o empresa
   - ✓ Horario en el que labora

Junto con la solicitud es necesario llenar y entregar la siguiente documentación establecida en un formato de Word:

1) **Plan de trabajo**, que incluye:

   a) Datos generales del prestador

      - ✓ Nombre
      - ✓ Carrera
      - ✓ Facultad o escuela
      - ✓ Semestre que cursa
      - ✓ Situación académica actual

   b) Datos de su estancia en DGSCA

      - ✓ Dirección donde colabora
      - ✓ Departamento donde está asignado
      - ✓ Nombre del responsable del proyecto

c) Datos referentes al proyecto donde participará:

   - ✓ Nombre del proyecto
   - ✓ Tiempo aproximado para el desarrollo del proyecto
   - ✓ Avance que se logrará con su participación
   - ✓ Breve descripción del proyecto
   - ✓ Objetivo del proyecto
   - ✓ Actividades a desarrollar
   - ✓ Herramientas que utilizará para el desarrollo del proyecto
   - ✓ Resultados esperados

2) **Informe de actividades,** el cual contiene:

   a) Datos generales del prestador

      - ✓ Nombre
      - ✓ Dirección donde colabora
      - ✓ Departamento donde está asignado
      - ✓ Nombre del responsable del proyecto

   b) Datos referentes a su participación en un proyecto

      - ✓ Nombre del proyecto
      - ✓ Objetivo del proyecto
      - ✓ Breve descripción del proyecto
      - ✓ Grado de avance
      - ✓ Actividades desarrolladas
      - ✓ Herramientas utilizadas y aplicadas
      - ✓ Resultados obtenidos

3) Relación de Cursos

Como parte de su formación en DGSCA, el alumno puede participar tomando o impartiendo cursos, esta información se debe anexar a la solicitud de beca, mediante un formato donde indica:

- ✓ Nombre del curso
- ✓ Duración
- ✓ Lugar donde se impartió
- ✓ Calificación obtenida

4) Evaluación

Al terminar de reunir la información, el alumno la entrega a su jefe inmediato, para que éste complete los requisitos, realizando una evaluación del desempeño del mismo, calificando los siguientes aspectos:

- ✓ Asistencia
- ✓ Actitud hacia las actividades encomendadas
- ✓ Compromiso
- ✓ Conocimientos en el área de cómputo
- ✓ Calificación global
- ✓ Comentarios

Dichos aspectos están formados por un grupo de preguntas y algunas de ellas tienen respuestas de opción múltiple.

El jefe inmediato canaliza la documentación a la Coordinación de Servicio Social, para que el encargado de servicio social los envíe a un Consejo que determinará si se otorga la beca o no.

Si el Consejo otorga la beca, informa al encargado de servicio social para que éste a su vez avise al jefe inmediato el dictamen de las becas y se le comunique al alumno.

En la figura 3.1 se muestra el diagrama de casos de uso de negocio, que representa el proceso descrito anteriormente.

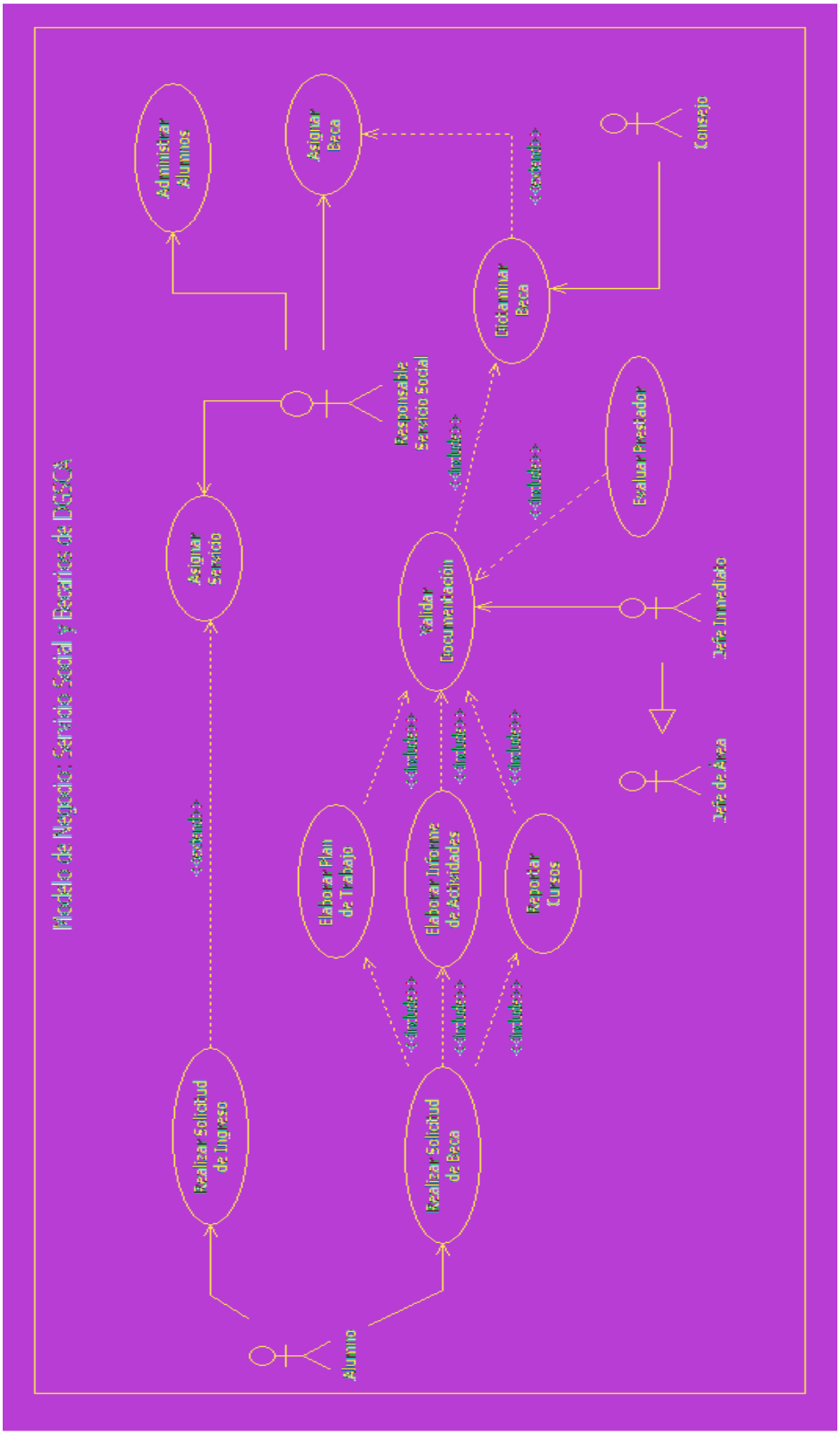


Figura 3.1 Diagrama de Casos de Uso de Negocio

3.1.2. Identificación de mejoras.

El proceso descrito anteriormente se realizaba manualmente, ya que únicamente se tenían capturados cada uno de los formatos en el procesador de palabras de Word y para su llenado se imprimían, generando una gran cantidad de papeles ya que para cada alumno se creaba un fólder con la documentación correspondiente. La información se mantenía archivada por un periodo mínimo de 5 años.

Aunado a esto, uno de los problemas a los que se enfrentaba la administración de servicio social, era que si un prestador llevaba tiempo colaborando en DGSCA y solicitaba una beca, el encargado de servicio social tenía que revisar toda su documentación para ver el avance que había logrado el alumno.

Como parte de la administración de la Coordinación de Servicio Social, constantemente se requiere realizar estadísticas que indiquen el número de alumnos que realizan servicio social en DGSCA, alumnos que tienen beca o que pertenecen a algún programa de beca o algún otro indicador que sea necesario, de manera que se tiene que revisar la documentación de cada alumno, hoja por hoja, hacer una clasificación de datos y un conteo de los mismos manualmente, para después procesarlos en Excel, implicando un procedimiento tardado.

Según datos de la Coordinación tienen que preparar los informes por lo menos con una semana de anticipación a la fecha de entrega para la Dirección General de DGSCA.

Cabe mencionar que en el modelo de negocio descrito no se tiene un seguimiento de las actividades que realizan todos los alumnos, solamente en el caso de que algún alumno sea candidato a obtener una remuneración económica se hace dicho seguimiento; por otro lado los formatos tienen información repetitiva como los datos personales, los datos escolares e información del departamento al que pertenecen.

Al efectuar el análisis del modelo de negocio se identificó que era necesaria la implementación de un sistema que automatice el proceso de negocio descrito anteriormente para agilizar los trámites, tanto para la administración de servicio social, como para los alumnos, teniendo una base de datos centralizada que cuente con una interfaz gráfica que presente los formatos adecuados para evitar la repetición de información, permitiendo llevar acabo todo el procedimiento con la finalidad de que sea mucho más cómodo registrarse y obtener la información requerida por parte de la Coordinación de Servicio Social.

3.2. Identificación de Requerimientos

Esta es la disciplina en la que se establece que es lo que tiene que hacer exactamente el sistema que se construya. En este entendido los requerimientos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen.

Los requisitos se dividen en dos grupos: Funcionales y no Funcionales.

- Los requisitos funcionales son las cosas que el sistema puede hacer, su funcionalidad. Se modelan mediante diagramas de casos de uso.
- Los requisitos no funcionales representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica.

El flujo de trabajo en esta disciplina es el siguiente:

- Analizar el problema.
- Entender las necesidades de cada uno de los Stakeholder.
- Definir el sistema.
- Administrar el alcance del sistema.
- Refinar la definición del sistema.
- Administrar los requerimientos de cambios.

3.2.1. Alcance del sistema

A partir del modelo planteado con anterioridad, surgen los siguientes requerimientos.

Se requiere que el registro de las solicitudes se lleve a cabo de manera estandarizada para cualquier tipo de servicio, de manera que el proceso que se implementará a través del sistema será el que a continuación se describe.

Inicialmente un alumno que desee integrarse a DGSCA deberá realizar una solicitud de servicio, para lo cual el sistema deberá mostrar los formularios correspondientes para ingresar datos personales; en donde los datos obligatorios serán los siguientes: nombre completo compuesto por nombre(s), apellido paterno y apellido materno; fecha de nacimiento, sexo, correo electrónico, dirección formada por calle y número, colonia, código postal y delegación. Adicionalmente, si el alumno lo desea, podrá proporcionar su RFC, CURP y algún teléfono ya sea fijo o celular.

Continuando con el registro, el alumno indicará la institución educativa a la que pertenece para poder agregar la escuela o facultad y así mismo la carrera, el sistema le permitirá al alumno seleccionar cada uno de estos datos. En el caso de los alumnos que pertenecen a la UNAM deberán proporcionar su número de cuenta, para alumnos de otras instituciones, este número será la matrícula con la que estén dados de alta en su escuela o facultad, dicho número no debe incluir caracteres como guiones o diagonales.

Como parte de los datos personales, el sistema solicitará que el alumno incluya el nombre de usuario mediante el cual podrá identificarse para acceder.

Una vez que ingresó sus datos personales y escolares, el alumno deberá indicar el servicio que solicita, éste puede ser servicio social o prebecario.

Cabe mencionar que los programas de servicio deberán estar dados de alta previamente con los siguientes datos obligatorios: nombre, descripción, modalidad; que servirá como indicador

para identificar si el servicio es interno o externo (es decir si pertenece a DGSCA o no) y tipo de servicio que deberá estar registrado con nombre, clave y estatus antes de realizar el alta de los programas de servicio.

La información del servicio deberá tener una clave de servicio, para el caso de las becas, indicará el tipo de beca y el periodo de acuerdo al año, por ejemplo si se trata de una beca interna vigente en el periodo ordinario 2009-1, la clave estará formada de la siguiente manera ORDINARIA2009-1, para el caso de los programas registrados en DGOSE se tomará la clave que proporciona dicha dependencia; cada programa tendrá una fecha de inicio y fin de registro, que indicará el periodo durante el que un alumno podrá solicitar el servicio y las fechas de inicio y fin de periodo del servicio, que es el periodo durante el cual el servicio estará activo.

En el caso de solicitar un plan de beca, la solicitud se podrá realizar únicamente dentro del plazo establecido por la convocatoria. El alumno podrá seleccionar la opción de prebecario y elegir el plan de becas en el que está interesado. Para el resto de los servicios, sólo se deben mostrar aquellos que tengan periodo de registro vigente y se podrá seleccionar sólo el tipo de servicio.

Si un alumno, desea registrarse para colaborar como becario en una dependencia gubernamental, registrará su solicitud como servicio de apoyo.

Continuando con el registro del servicio solicitado, el alumno deberá proporcionar: promedio, semestre y porcentaje de créditos acumulados al día del registro y la hora a partir de la cual podrá asistir a DGSCA.

Parte fundamental de la solicitud es el ingreso de un currículo, mediante el cual el alumno podrá hacer referencia a los lenguajes de programación y paquetería que maneja, como: procesadores de palabras, hojas de cálculo, estadísticos, bases de datos y graficación, cada uno de éstos bloques agrupa una o varias aplicaciones (temas informáticos), como pueden ser Java, C, Word, HTML, etcétera.

Para cada uno de estos bloques, un alumno tendrá la facilidad de ingresar una o más aplicaciones, indicando para cada una el nivel de manejo como bajo, medio o alto.

También deberá ingresar los idiomas que maneja junto con el nivel que habla, lee y escribe para cada uno de ellos ya sea bajo, medio o alto.

Para finalizar el currículo, el alumno deberá ingresar su experiencia académica y profesional si así lo desea, así como sus intereses personales.

Al terminar el llenado de su solicitud, el alumno quedará dado de alta en el sistema como candidato. En este punto, el sistema deberá generar un registro de servicio, el cual indicará que se trata de una solicitud del servicio. La contraseña será generada a través del sistema, dicha contraseña será única y estará formada por caracteres alfanuméricos, contendrá mayúsculas y minúsculas y le será enviada automáticamente al candidato junto con el nombre de usuario que proporcionó en la solicitud, vía e-mail.

Una vez que el candidato cuente con su usuario y contraseña de acceso, tendrá la posibilidad de modificar los datos personales entre ellos el e-mail, su dirección y sus datos escolares, el sistema no permitirá modificar el nombre de usuario que proporcionó ni la contraseña que le fue asignada.

Por otro lado, el candidato podrá modificar el semestre, promedio, porcentaje de créditos, el horario y el servicio solicitado, siempre y cuando no haya sido asignado a un servicio. Así mismo el candidato podrá agregar o eliminar idiomas y aplicaciones y cambiar su información complementaria.

Después de haber realizado la solicitud de ingreso, la información del candidato permanecerá en una cartera con la finalidad de que este disponible para que el responsable de servicio pueda consultar los datos personales, el currículo y el servicio que esta solicitando el candidato para que posteriormente, si el candidato cuenta con el perfil requerido, éste sea notificado vía e-mail para que se presente a una entrevista.

En caso de que el alumno sea aceptado, el responsable de servicio asignará el servicio que esta solicitando el candidato, para ello consultará el servicio solicitado y actualizará la solicitud de ingreso, modificando el tipo de servicio en caso de que se desee y escogiendo un programa de servicio vigente en el que participará el prestador. En caso de que el tipo de servicio sea beca, el responsable de servicio podrá indicar si ésta fue aceptada o rechazada. Si la beca fue aceptada deberá seleccionar el nivel de beca; los cuales deberán estar disponibles con la siguiente información: descripción, monto y clave de nivel. Así mismo se podrán agregar comentarios u observaciones referentes a la beca e indicar las fechas de inicio y fin que estará el prestador en la dependencia. Cabe mencionar que para una solicitud de beca o plan de becas, se manejará un estatus previo ya que estos registros se deben realizar con cierto tiempo de anticipación y están sujetos a una aprobación, este estatus indicará que el Prestador esta en un proceso de solicitud.

En este punto, el responsable de servicio social también deberá asignarle al prestador un responsable directo quien será el encargado de supervisar sus actividades.

Es importante hacer notar que los responsables deberán estar organizados de acuerdo a la función que realizan: el responsable de servicio social será el encargado de asignar los servicios y darle seguimiento al avance del prestador. Los responsables de área serán quienes validen tanto los planes de trabajo como los informes de los prestadores que supervisan los responsables directos que tienen a su cargo. Los responsables directos son quienes trabajarán conjuntamente con los prestadores, serán los encargados de asignarles un proyecto así como de dirigir sus actividades, en primera instancia serán quienes validen el plan de trabajo y el informe realizados por un prestador y serán quienes evalúen el desempeño del mismo. Por otro lado, se agregará el rol de responsable del sistema, quien realizará las funciones de administrador del sistema, se encargará de proporcionar el mantenimiento a los catálogos, dará de alta a los responsables para que puedan acceder al sistema, registrará los servicios y administrará la vigencia de los mismos y también dará de alta los proyectos.

Los responsables se identificarán con los siguientes datos: nombre completo del responsable, formado por nombre(s), apellido paterno y apellido materno, e-mail, nombre de usuario,

centro, unidad, dirección en la que labora así como el perfil. Mediante el sistema se ingresará información referente al puesto en el que labora el responsable y además en caso de que el responsable tenga un responsable superior, se deberá seleccionar el nombre del mismo, dicho responsable debe pertenecer al mismo centro, dirección y unidad que su subordinado.

En el caso del centro, la unidad, la dirección y el puesto, deberán estar previamente registrados mediante nombre y estatus, para que se puedan seleccionar al momento del registro de los responsables. Los datos del responsable podrán ser modificados excepto el nombre de usuario y la contraseña.

Continuando con el registro de servicio del prestador, el responsable de servicio social, tendrá la facultad de modificar el semestre que cursa el prestador, el porcentaje de créditos y el promedio, en caso de que se requiera. Después de finalizado el proceso, el estatus del registro deberá cambiar de solicitud a asignado.

Por cada dirección de DGSCA se tendrán proyectos registrados con los siguientes datos: nombre del proyecto, objetivo, descripción, fecha de inicio, fecha fin y dirección, además de un responsable, quien debe pertenecer a la dirección seleccionada. Un proyecto podrá ser permanente, indicándose al darlo de alta y el sistema no deberá solicitar la fecha de fin.

Después de que se ha registrado el prestador, el responsable directo le asignará un proyecto, si un prestador esta como prebecario no se le asignará ningún proyecto ya que en esta etapa solamente participan en una capacitación.

Una vez que el prestador tiene asignado un proyecto, el alumno generará un plan de trabajo para éste, por lo que deberá ingresar la siguiente información: resultados a obtener, avance que logrará, herramientas a utilizar y actividades a realizar. El responsable directo, aprobará el plan de trabajo, para ello podrá consultar toda la información en forma de reporte incluyendo los datos generales del prestador y la información general del proyecto, cabe mencionar que el plan de trabajo podrá ser modificado por el prestador antes de que sea aprobado.

Al finalizar el periodo de servicio, el alumno elaborará un informe de trabajo indicando el avance logrado en el proyecto, los resultados obtenidos, las actividades que realizó y las herramientas que utilizó; para las actividades y herramientas, se deberán mostrar las actividades y herramientas que ingreso en el plan de trabajo, de manera que el prestador indicará cuales realizó o utilizó y si efectúo otras actividades y utilizó otras herramientas podrá ingresarlas. El informe será aprobado por el responsable directo. Si es necesario, el prestador podrá modificar la información proporcionada, al igual que el plan de trabajo una vez aprobado no podrá ser modificado.

Si el plan de trabajo o el informe no han sido aprobados primero por el responsable directo no podrán ser aprobados por el responsable de área. Debido a que puede cambiar el responsable directo de un prestador, es necesario guardar el nombre del responsable directo en ambas aprobaciones, en el plan y en el informe.

El prestador podrá agregar los cursos que haya tomado o impartido en DGSCA. Para realizar dicho registro proporcionará los siguientes datos: nombre del curso, el cual podrá ser seleccionado de la lista de cursos registrados previamente en el sistema, fecha de inicio de curso y fecha de fin de curso, lugar donde se impartió y calificación obtenida.

Los cursos estarán registrados con la siguiente información: nombre del curso, estatus para indicar si esta activo o no, tipo de curso para hacer referencia a si es un curso de DGSCA o es un curso externo que identifica al curso.

Al finalizar el periodo de servicio, el responsable directo, evaluará el desempeño del prestador, para este fin, el sistema contará con el formato de evaluación donde se califican los siguientes rubros: asistencia, actitud hacia las actividades encomendadas, compromiso, conocimientos en el área de cómputo, se proporcionará una calificación global al desempeño del Prestador y se podrán agregar comentarios.

Cada rubro de la evaluación estará formado por una serie de preguntas con algunas respuestas opcionales y otras abiertas.

Cuando un prestador termine su servicio, si el responsable directo decide que el prestador puede ser candidato a obtener una beca económica le dará permiso de ingresar un nuevo registro para solicitar la beca, este registro se hará de acuerdo a la convocatoria de becas que se manejan en DGSCA u otras dependencias gubernamentales por lo que el responsable de servicio social deberá crear un nuevo registro de periodo de beca indicando: periodo, tipo de beca, inicio beca y fin beca; para este caso, se habilitará la opción beca en tipo de servicio, de manera que prestador podrá seleccionarla.

Una vez que el alumno tenga permiso de llenar una solicitud de beca, ingresará la siguiente información: promedio, porcentaje de créditos, semestre actual, tipo de servicio, responsable directo y tipo de beca. En caso de renovación de beca deberá indicarse en la solicitud.

Cuando se abre una convocatoria para una beca externa, por ejemplo de tipo gubernamental, se debe seguir el mismo proceso para solicitar un servicio, en caso de que el alumno no pertenezca a la dependencia, deberá registrar su solicitud con tipo de servicio de apoyo.

Después de una serie de entrevistas y evaluaciones si el interesado es aceptado para obtener la beca, el responsable de servicio social hará la asignación de está. Para este tipo de becas no es necesario tener antecedentes en DGSCA, pues esta convocatoria se abre al público en general. Si el alumno ya es prestador, el responsable directo deberá darle permiso para solicitar un nuevo servicio como se mencionó anteriormente y el prestador podrá seleccionar el servicio como becario.

Después de registrar la solicitud de beca, se repite el proceso a partir de asignar servicio.

3.2.2. Modelado de Casos de Uso

Para una mejor organización y entendimiento de los requerimientos se llevó a cabo la agrupación de los mismos como se muestra en la figura 3.2, según la funcionalidad a realizarse:

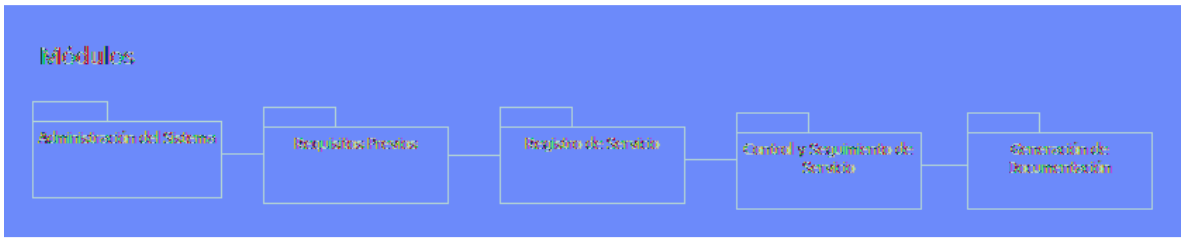


Figura 3.2 Módulos del Sistema Control de Prestadores

a) Administración del Sistema, que incluye el mantenimiento de los catálogos para el sistema control de prestadores de DGSCA.

b) Requisitos Previos, que abarca: administrar responsables, administrar proyectos y administrar servicios.

c) Registro de Servicio, que incluye: registrar candidatos, administrar prestadores cuya funcionalidad a agrupar es:
   - Solicitar servicio.
   - Asignar servicio.
   - Asignar prestadores a proyectos.
   - Evaluar prestadores.
   - Registrar plan de trabajo.
   - Registrar informe de trabajo.
   - Registrar cursos.

d) Control y Seguimiento de Servicio, en cuyo módulo, se describe la funcionalidad para las consultas y modificaciones de los registros de candidatos y prestadores.

e) Generación de Documentación, el cual provee la generación de históricos de alumnos y estadísticas.

Así mismo se definieron claramente los actores involucrados:

- Responsable de servicio social
- Responsable de área
- Responsable directo
- Candidato
- Prestador
- Administrador del sistema

A continuación se enlistan los casos de uso identificados:

Módulo	Id	Caso de Uso
Administración del Sistema	CU-0010	Administrar Catálogo
	CU-0020	Dar de Alta Catálogo
Requisitos Previos	CU-0030	Administrar Responsable
	CU-0040	Dar de Alta Responsable
	CU-0050	Administrar Proyecto
	CU-0060	Dar de Alta Proyecto
	CU-0070	Administrar Programa de Servicio
	CU-0080	Dar de Alta Programa de Servicio
Registro de Servicio	CU-0090	Registrar Solicitud
	CU-0100	Generar Contraseña
	CU-0110	Enviar e-mail
	CU-0120	Asignar Servicio
	CU-0130	Renovar Servicio
	CU-0140	Asignar Proyecto
	CU-0150	Agregar Plan de Trabajo
	CU-0160	Agregar Informe de Trabajo
	CU-0170	Registrar Cursos
	CU-0180	Evaluar Prestador
	CU-0190	Consultar Registro de Servicio
	CU-0200	Consultar Alumno
	CU-0210	Habilitar Solicitud de Servicio
Control y Seguimiento de Servicio	CU-0190	Consultar Registro de Servicio
	CU-0200	Consultar Alumno
	CU-0220	Administrar Datos Personales
	CU-0230	Administrar Currículo
	CU-0240	Administrar Servicio
	CU-0250	Administrar Plan de Trabajo
	CU-0260	Administrar Informe de Trabajo
	CU-0270	Administrar Cursos
	CU-0280	Administrar Evaluación
Generación de Documentación	CU-0290	Generar Histórico de Alumno
	CU-0300	Generar Estadísticas

Tabla 3.1 Casos de Uso del Sistema Control de Prestadores

En el **Anexo D** se presentan los diagramas de casos de uso referidos.

Cabe mencionar que a partir de esta fase y para mejor comprensión del proceso de desarrollo de software se tomará un caso de estudio que será explicado a lo largo del presente trabajo.

Partiendo de que el caso ejemplo será el proceso de Registro de un Candidato, se incluye diagrama de casos de uso correspondiente, en la figura 3.3.

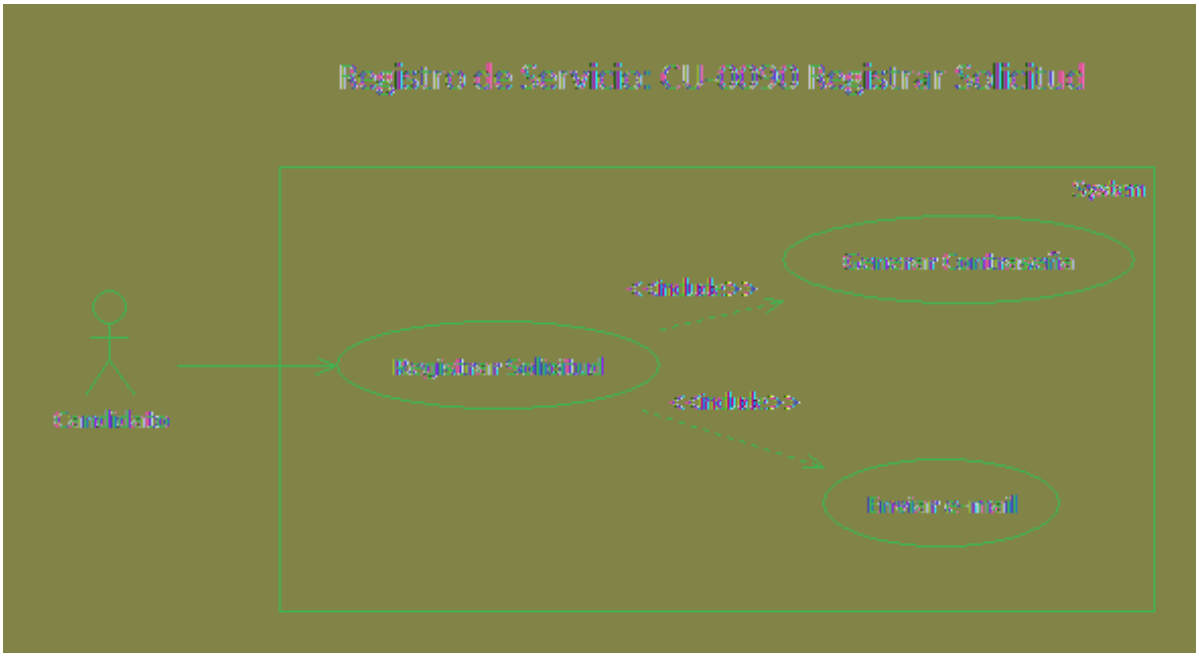


Figura 3.3 Diagrama de Casos de Uso: Registrar Solicitud

Una vez que se ha identificado el caso de uso, se procede a realizar la especificación del mismo. Durante el proceso de desarrollo de software, dichas especificaciones pueden ir adquiriendo complejidad en la medida en que se van afinando los detalles tanto de los requerimientos como de la solución tecnológica.

A continuación se incluye la especificación completa para el caso de uso CU-0090 Registrar Solicitud, esta especificación muestra mayor detalle y esta estructurada de manera que se permite entender los objetivos, las tareas y requisitos, para ello se tienen que identificar al menos, los siguientes aspectos:

1. *Objetivo*. En esta sección se describe que funcionalidad se pretende alcanzar mediante la realización del caso de uso.
2. *Actores involucrados*. Se listan los actores que participan en la ejecución del caso de uso.
3. *Precondiciones*. Establecen lo que siempre debe cumplirse antes de comenzar el flujo principal del caso de uso.
4. *Flujo Principal*. Describe el camino de éxito para lograr el objetivo del caso de uso. Puede incluir: interacción entre actores y validaciones a cargo del sistema.
5. *Flujos Alternos*. Se deben incluir todos los escenarios o bifurcaciones que no están contempladas dentro del camino de éxito.
6. *Flujos de Excepción*. Se describen las acciones a realizarse en caso de error en el caso de uso.
7. *Post-condiciones*. Establecen que debe cumplirse cuando el caso de uso se termina con éxito.

Así mismo se pueden incluir secciones como:

1. *Reglas de negocio*. En esta sección se incluyen las reglas que deben cumplirse y que han sido requeridas por el modelo de negocio.
2. *Requerimientos no funcionales*. Si algún caso de uso incluye una restricción, un requerimiento no funcional o alguna cualidad especifica como rendimiento, facilidad de uso o diseño, se deben incluir dentro de la especificación.
3. *Notas para la implementación*. Si se requiere que técnicamente se implemente algo de manera específica, se debe incluir en el caso de uso.

Generales del Caso de Uso			
Nombre Caso Uso	CU-0090 Registrar Solicitud		
Creación	Aparicio Arista Reyna Elizabeth Rosas Bernal María del Rosario	**Fecha**	Agosto – 2008

Objetivo

Permitir a los interesados en formar parte de DGSCA, llenar una solicitud de ingreso la cual incluye Datos Personales, Domicilio y Datos Escolares; así como un breve Currículo con información sobre los idiomas que maneja, lenguajes de programación y paquetes, por último le permite agregar información complementaria respecto a su experiencia tanto académica como laboral y le permite adicionar una descripción de sus intereses personales y profesionales.

Nivel del Caso de Uso	Prioridad	Complejidad
Usuario	Alta	Alta

Actores involucrados

- **Candidato**.- Es la persona interesada en formar parte de DGSCA, ya sea como apoyo o servicio social. Tiene acceso a realizar la solicitud de servicio.
- **Sistema**.- Es el encargado de registrar la solicitud del Candidato, así como de almacenar su currículo.

Precondiciones

1. Acceder a la página de inicio para llenar una solicitud.
2. En el Sistema debe haber información en catálogos de los siguientes rubros: Nivel Académico, Escuela o Facultad, Carrera, Herramientas Informáticas e Idiomas.
3. En el Sistema deben estar habilitados programas de servicio, es decir debe haber periodos de registro activos.

Post-condiciones

1. Se almacenarán los datos personales, escolares, currículo e intereses proporcionados por el Candidato.
2. Se envía nombre de usuario y contraseña de acceso al sistema para el Candidato que realizó su solicitud.
3. El Responsable de Servicio Social podrá consultar la información proporcionada por el Candidato.

Escenario Principal

Paso	Acción
1.	El Candidato solicita registrarse en el sistema.
2.	El Sistema verifica si se tienen programas de servicio activos.
3.	En caso de que se tengan programas de servicio activos el sistema muestra la pantalla con los campos para ingresar los datos personales. En caso contrario, ***ver escenario de excepción EX01.***
4.	El Candidato ingresa los datos personales: Nombre, Apellido Paterno, Apellido Materno, RFC, CURP, Fecha de Nacimiento, Sexo y Correo Electrónico. ***Ver regla de negocio RN-001.***
5.	El Candidato proporciona un Nombre de Usuario con el cual podrá acceder al sistema. ***Ver regla de negocio RN-002.***
6.	El Candidato ingresa los datos correspondientes a su domicilio, como son: Calle y número, Colonia, Delegación, Código Postal, Teléfono fijo y Teléfono celular. ***Ver regla de negocio RN-003.***
7.	El Candidato selecciona la institución académica en la que estudia ***Ver regla de negocio RN-004.***
8.	El Sistema despliega los nombres de escuelas y facultades que pertenecen a la institución seleccionada por el Candidato.
9.	El Candidato selecciona la Escuela/Facultad dónde está inscrito. ***Ver regla de negocio RN-004.***
10.	El Sistema despliega las carreras dadas de alta en la Escuela/Facultad, seleccionada por el Candidato.
11.	El Candidato selecciona la carrera que cursa. ***Ver regla de negocio RN-004.***
12.	El Candidato ingresa el número de cuenta o matrícula. ***Ver regla de negocio RN-005.***
13.	El Candidato indica que ha finalizado de capturar su información personal.
14.	El Sistema verifica que se hayan ingresado los campos requeridos de Datos Personales, Domicilio y Datos Escolares.
15.	En caso de que los datos estén correctos, el sistema despliega la pantalla de información del servicio. En caso contrario, ***ver escenario alterno EA01.***
16.	El Candidato proporciona información complementaria para realizar la solicitud de servicio, mediante los campos: Semestre, Promedio, Porcentaje en créditos, Horario disponible y Tipo de servicio solicitado. ***Ver regla de negocio RN-006.***
17.	El Candidato selecciona el tipo de servicio que desea realizar.
18.	El Sistema verifica el tipo de servicio que seleccionó el Candidato.
19.	Si el tipo de servicio es prebecario, el Sistema busca los programas de becas que están activos y se los muestra el Candidato. *En caso contrario continuar con el paso 21.*
20.	El Candidato selecciona el programa de servicio en el que desea participar.
21.	El Candidato indica que ha finalizado de ingresar la información complementaria para la solicitud de servicio.
22.	El Sistema verifica que se hayan ingresado los campos requeridos de Solicitud de Servicio. ***Ver regla de negocio RN-007.***
23.	En caso de que se hayan ingresado correctamente los datos del servicio, el sistema muestra información de los bloques de herramientas registrados en el sistema. En caso contrario, ***ver escenario alterno EA02.***

24.	El Candidato selecciona el bloque que desea agregar.
25.	El Sistema despliega las herramientas informáticas almacenadas en la base de datos, para el bloque seleccionado.
26.	El Candidato selecciona las herramientas informáticas que maneja, indicando el nivel de dominio para cada una, como: alto, medio o bajo. ***Ver regla de negocio RN-008.***
27.	El Candidato indica que ha finalizado de ingresar las herramientas informáticas que maneja.
28.	El Sistema muestra la información de los idiomas que el Candidato puede seleccionar.
29.	El Candidato selecciona los idiomas que domina e indica el nivel de manejo para cada uno como: bajo, medio, avanzado. ***Ver regla de negocio RN-009.***
30.	El Candidato indica que ha finalizado de ingresar los idiomas que maneja.
31.	El Sistema verifica si el Candidato selecciono idiomas, que se haya ingresado el nivel de manejo.
32.	En caso de que se haya ingresado correctamente la información de los idiomas, el Sistema muestra la pantalla para ingresar información complementaria. En caso contario ***ver escenario alterno EA03***.
33.	El Candidato ingresa información acerca de su experiencia académica, experiencia profesional e intereses. ***Ver regla de negocio RN-010.***
34.	El Candidato indica que la información ingresada es verídica.
35.	El Sistema verifica que el Candidato haya seleccionado que la información que proporcionó es verídica y que haya indicado sus intereses. En caso contrario, ***ver escenario alterno EA04.***
36.	El Sistema genera una contraseña para el Candidato. ***Ver caso de uso CU-0100 Generar Contraseña***.
37.	El Sistema guarda toda la información proporcionada por el Candidato en la base de datos.
38.	Si se guardo correctamente la información el Sistema envía vía e-mail el nombre de usuario y la contraseña que le permitirán al Candidato acceder al sistema. ***Ver caso de uso CU-0110 Enviar e-mail.*** En caso contrario ***ver escenario de excepción EX02.***
39.	El Sistema despliega la información ingresada por el Candidato.
40.	Fin de Caso de Uso.

Escenarios Alternos

EA01 – Información incorrecta o incompleta.	
Paso	**Acción**
1.	Si la información que se ingreso es incorrecta, el sistema despliega el siguiente mensaje: "**El *campo* contiene caracteres no permitidos"**, indicando el nombre del campo incorrecto.
2.	Si no se ingresaron los campos requeridos, el sistema despliega el siguiente mensaje: "**El *campo* es requerido** ", indicando el nombre del campo que es requerido.
3.	Regresar al paso 3 del escenario principal.

EA02 – Información incorrecta o incompleta.	
Paso	**Acción**
1.	Si la información que se ingreso es incorrecta, el sistema despliega el siguiente mensaje: "**El *campo* contiene caracteres no permitidos"**, indicando el nombre del campo incorrecto.

2.	Si no se ingresaron los campos requeridos, el sistema despliega el siguiente mensaje: "**El *campo* es requerido** ", indicando el nombre del campo que es requerido.
3.	Regresar al paso 15 del escenario principal.

EA03 No se ingreso el nivel de manejo de idioma.	
Paso	**Acción**
1.	Si no se ingreso el nivel de manejo de idioma, el Sistema despliega el siguiente mensaje: **"Indicar el nivel que se tiene para leer, hablar o escribir"**, según sea el caso.
2.	Regresar al paso 28 del escenario principal.

EA04– Sin confirmación de veracidad.	
Paso	**Acción**
1.	Si el Candidato no ha marcado la casilla donde se indica que la información proporcionada es verídica, el sistema desplegará el siguiente mensaje:**"Falta aceptar que la información enviada es verídica"**.
2.	Regresar al paso 32 del escenario principal.

Excepciones

EX01 No existen programas de servicio activos.	
Paso	**Acción**
1.	Si no existen programas de servicio activos, el sistema despliega una pantalla indicándole al Candidato que por el momento no se tienen servicios activos, mediante el siguiente mensaje: "***Estimado Alumno, lo sentimos por el momento no contamos con programas de servicio activos para registrar solicitudes, favor de intentarlo en otra ocasión***."
2.	Fin de Caso de Uso.

EX02– No se almacenó la información.	
Paso	**Acción**
1.	Si la información ingresada por el Candidato no se almacenó en la Base de Datos, el sistema despliega el siguiente mensaje:**"Error al guardar la información."**
2.	Fin de Caso de Uso.

Reglas de Negocio

Id	Regla de Negocio
RN-001	Los datos personales obligatorios son: • Nombre • Apellido Paterno

	• Apellido Materno • Fecha de Nacimiento • Sexo • Correo Electrónico
RN-002	El nombre de usuario debe estar formado por seis caracteres, puede incluir mayúsculas, minúsculas, guión bajo y guión medio, así como números.
RN-003	Los datos obligatorios para el domicilio son: • Calle y número • Colonia • Código postal • Delegación
RN-004	Todos los datos escolares son obligatorios, el candidato debe seleccionar institución, escuela/facultad y carrera.
RN-005	En el caso de alumnos que no estén inscritos en la UNAM el número de cuenta será la matrícula con la que estén dados de alta en su escuela o facultad, dicho número no debe incluir caracteres como guiones o diagonales
RN-006	En el caso del Horario disponible, el candidato deberá indicar la hora a partir de la cual puede asistir a DGSCA a realizar su servicio; el sistema calculará la hora fin de tal manera que el horario complete las 4 horas reglamentarias.
RN-007	Todos los datos escolares complementarios son requeridos.
RN-008	No es obligatorio ingresar herramientas informáticas. En caso de requerirse, el candidato puede agregar una o más herramientas informáticas.
RN-009	No es obligatorio ingresar idiomas. En caso de requerirse, el Candidato puede agregar uno o más idiomas y será obligatorio indicar el nivel de manejo.
RN-010	Es opcional ingresar información acerca de experiencia académica, profesional. Es obligatorio ingresar información para intereses.

Requerimientos No-Funcionales

Id	Requerimiento No-Funcional
RNF-001	

Notas para implementación
1.

Las especificaciones para los casos de uso CU-0100 y CU-0110, se encuentran en el **Anexo E**.

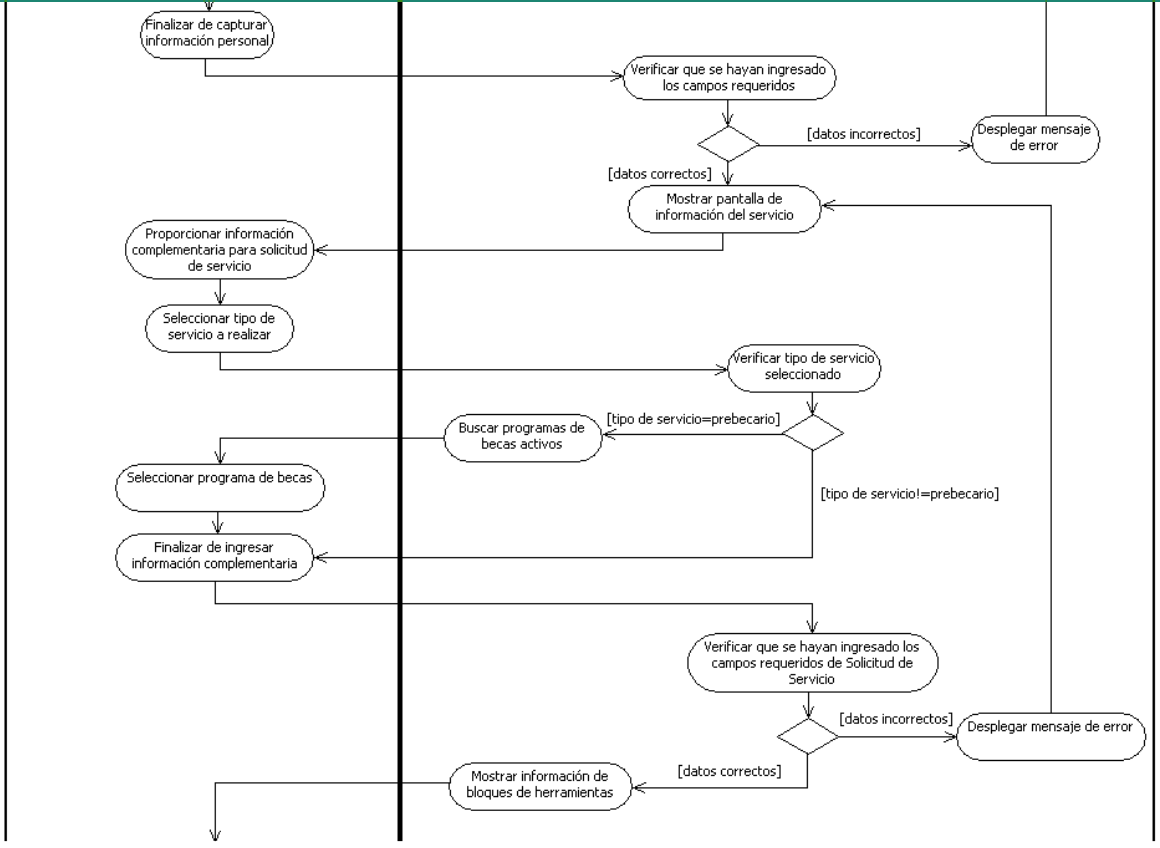
3.2.3. Diagrama de Actividades.

Por otro lado, se realizó el siguiente diagrama de actividades, de manera que nos permita visualizar la interacción de los diferentes actores y el flujo de las acciones que se realizan para alcanzar el objetivo del caso de uso en cuestión. Al igual que las especificaciones, durante el ciclo de desarrollo, el detalle de estos diagramas se va complementando.

El diagrama que se presenta en la figura 3.4, muestra la comunicación entre el candidato y el sistema, al momento de registrar una solicitud para ingresar a algún programa de servicio de DGSCA.

En el **Anexo F**, se incluyen los diagramas de actividades correspondientes a los casos de uso CU-0100 y CU-0110.


Registrar Solicitud
Candidato
Sistema
Solicitar registro en el sistema
Verificar si se tienen programas de servicio activos
[no existen servicios activos]
Desplegar pantalla indicando que no se tienen servicios activos
[existen servicios activos]
Mostrar pantalla con campos para ingresar los datos personales
Ingresar datos personales
Proporcionar nombre de usuario para acceder al sistema
Ingresar domicilio
Seleccionar institución académica en la que estudia
Desplegar nombres de escuelas y facultades
Seleccionar Escuela/Facultad
Desplegar carreras dadas de alta en la Escuela/Facultad
Seleccionar carrera
Ingresar número de cuenta o matrícula


Finalizar de capturar información personal
Verificar que se hayan ingresado los campos requeridos
[datos incorrectos]
Desplegar mensaje de error
[datos correctos]
Mostrar pantalla de información del servicio
Proporcionar información complementaria para solicitud de servicio
Seleccionar tipo de servicio a realizar
Verificar tipo de servicio seleccionado
[tipo de servicio=prebecario]
Buscar programas de becas activos
Seleccionar programa de becas
[tipo de servicio!=prebecario]
Finalizar de ingresar información complementaria
Verificar que se hayan ingresado los campos requeridos de Solicitud de Servicio
[datos incorrectos]
Desplegar mensaje de error
[datos correctos]
Mostrar información de bloques de herramientas

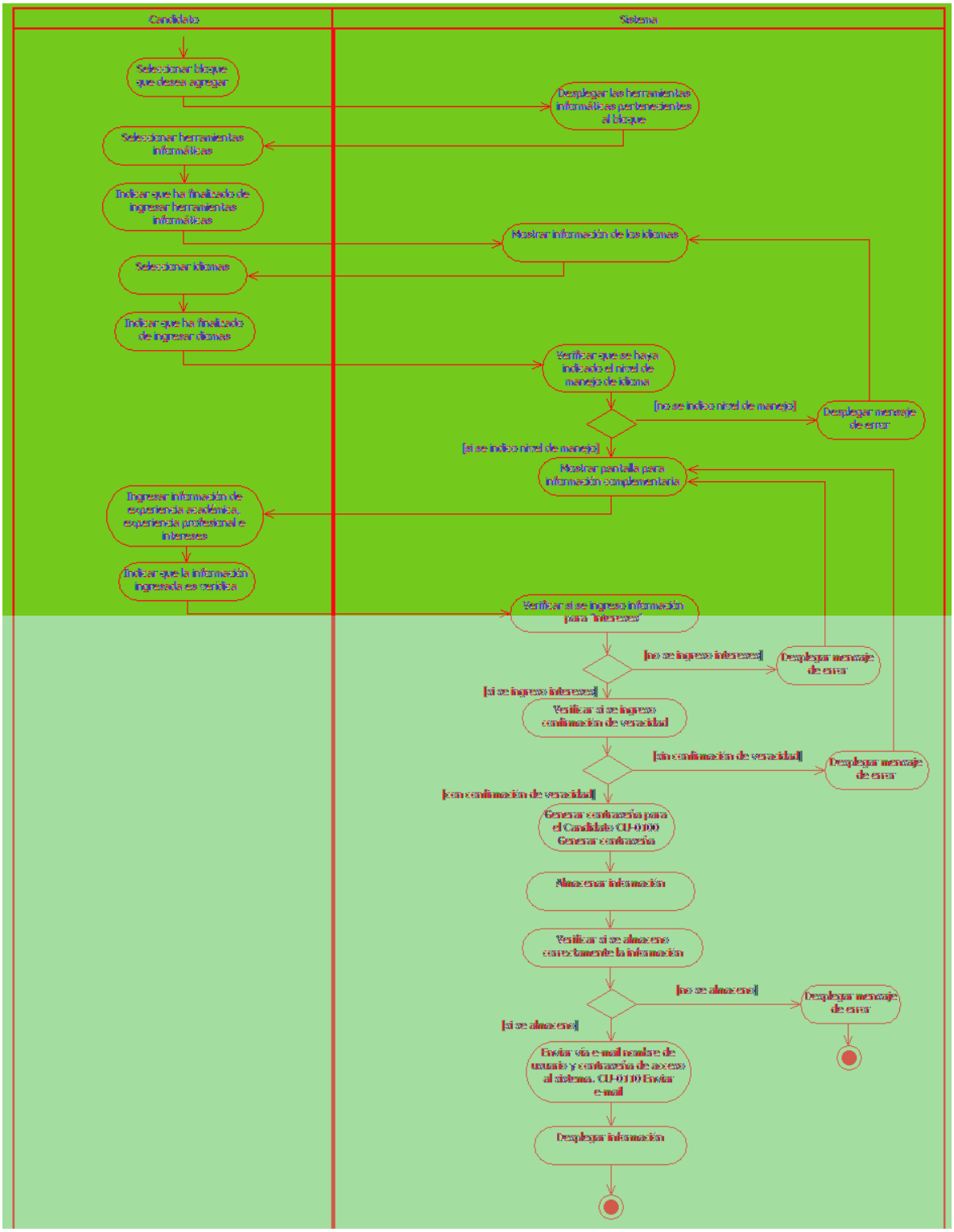


Figura 3.1 Diagrama de Actividad: Registrar Solicitud

3.2.4. Fuera de Alcance

Como parte de la identificación de requerimientos es indispensable listar que no hará el sistema, de manera que no se generen falsas expectativas a los usuarios. En el caso del sistema de Control de Prestadores:

a) No tendrá la funcionalidad de generar cartas de aceptación de servicio social que se mencionan en el modelo de negocio.

b) El sistema no administrará información específica para los planes de beca, es decir la documentación que se requiera en particular para ingresar a dichos planes.

c) En la solicitud de beca original, se incluía un campo denominado “otras remuneraciones”, el cual fue descartado debido a que no es un dato significativo para la Coordinación de Servicio Social.

En el siguiente capítulo se detallará el análisis y diseño del caso de estudio.

4. Elaboración

De acuerdo a RUP, en esta fase, tanto la funcionalidad como el dominio del problema se estudian en profundidad. Se define una arquitectura básica sobre la que se asentará la fase de construcción. Las tareas a realizarse durante esta fase son:

- Administrar la interfaz técnica.
- Detallar Requerimientos.
- Detallar Casos de Uso de Sistema.
- Detallar Requerimientos Funcionales.
- Detallar Requerimientos No Funcionales.
- Diseñar Componentes.
- Diseñar la Interfaz de Usuario.
- Revisar el Diseño de la Interfaz de Usuario.
- Aprobar el Diseño de la Interfaz de Usuario.
- Diseñar la Base de Datos.
- Administrar de cambios.
- Administrar el proyecto.

4.1. Interfaz de Usuario: Prototipos

Parte esencial del entendimiento entre el usuario y el personal de sistemas, es la generación de prototipos que le brinden al usuario un acercamiento a lo que será el producto terminado.

RUP da la libertad de construir prototipos como una estrategia para disminuir riesgos, ya que al poder mostrar al usuario una aproximación de lo que será el sistema, se evitan falsas expectativas, es por ello que una vez definido el flujo de los procesos a implementar, se diseñan los prototipos, en este caso se realizó una validación con el usuario de los siguientes modelos de pantallas que corresponden al caso de estudio.

En la figura 4.1, se puede apreciar cada uno de los campos que se solicita ingrese el candidato para realizar su registro.

Dicha pantalla esta divida por secciones para que sea más fácil el entendimiento; en la primera sección se solicitan los datos personales, posteriormente el domicilio y para finalizar se solicita información académica, tal como se ve en el diseño, esta parte será implementada mediante combos que le permitan al candidato seleccionar una opción para cada uno de los datos solicitados, tales como institución educativa, escuela o facultad y carrera.

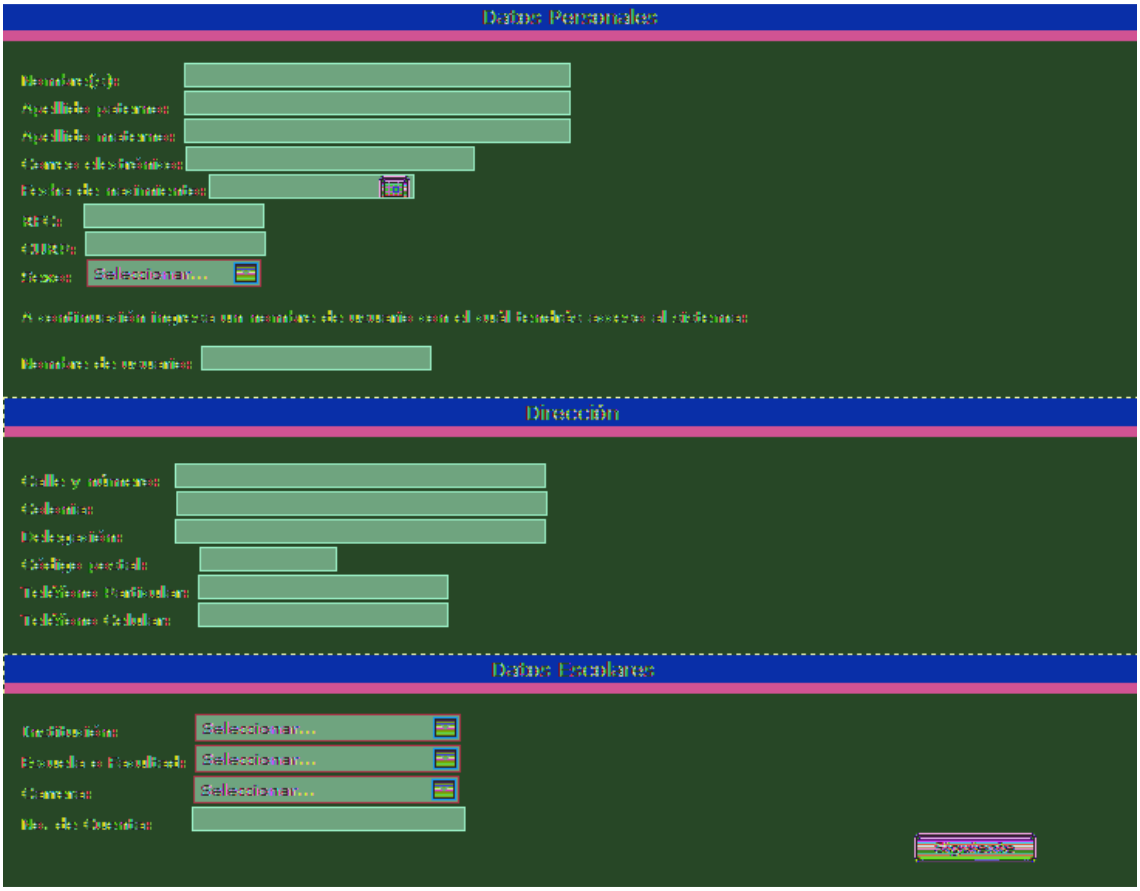


Figura 4.1 Prototipo: Ingresar Datos Personales

En la figura 4.2, se observa la pantalla que permitirá que se ingresen datos específicos en cuanto al desempeño académico además de que se realice la solicitud de servicio al que se desea ingresar indicando datos como el horario y el tipo de servicio a realizar.

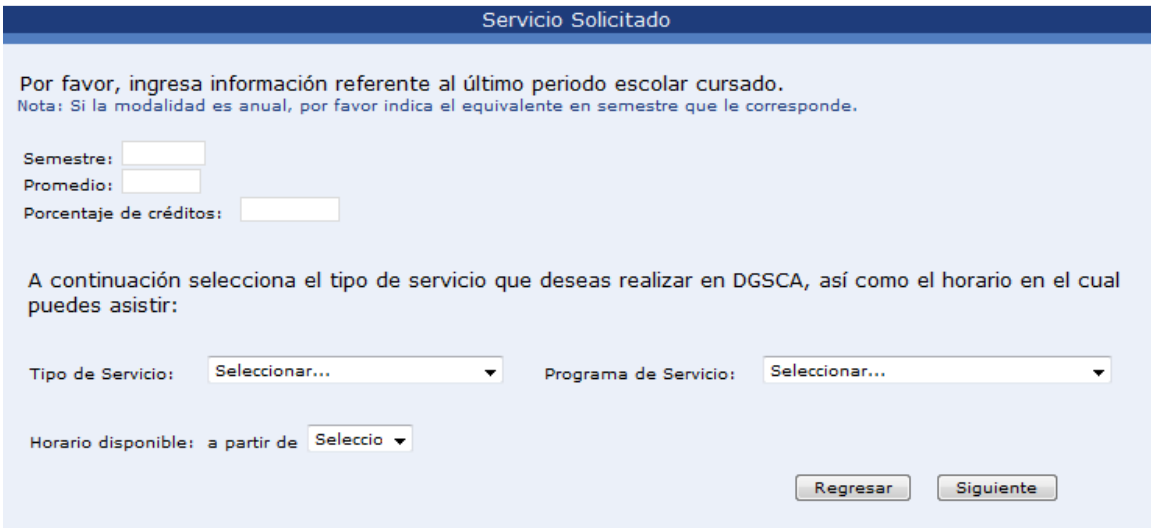


Figura 4.2 Prototipo: Ingresar Servicio

Posteriormente, parte importante de la solicitud es el currículum del candidato, para ello se presentan los siguientes prototipos:

El primero de ellos muestra en la figura 4.3 y detalla la forma en la que se ingresarán los conocimientos de herramientas de tecnologías de la información, con ayuda de listas desplegables y radio botones que permitirán seleccionar el bloque de conocimiento que se desea registrar.

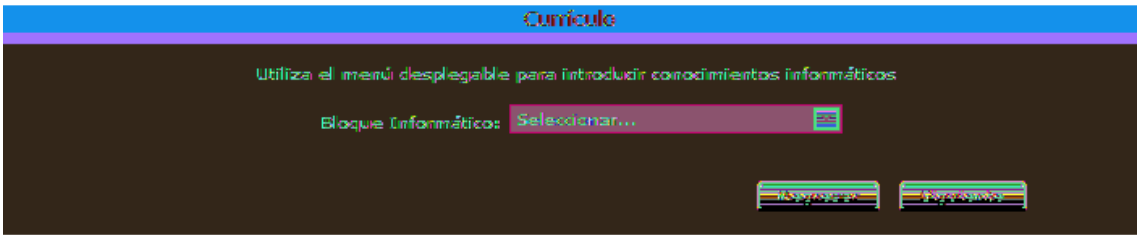


Figura 4.3 Prototipo: Ingresar Currículo

Una vez seleccionado el bloque se mostrarán las áreas temáticas encontradas para seleccionar el nivel de manejo de aquellas que desee reportar el candidato, esto se puede apreciar en la figura 4.4.

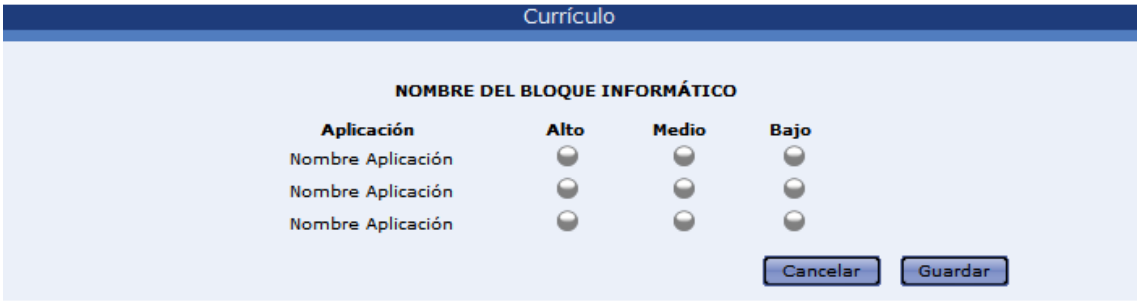


Figura 4.4 Prototipo: Ingresar Currículo - Selección

Conforme se vayan ingresando las aplicaciones, se irán mostrando en la parte superior de la pantalla, como se muestra en la figura 4.5.

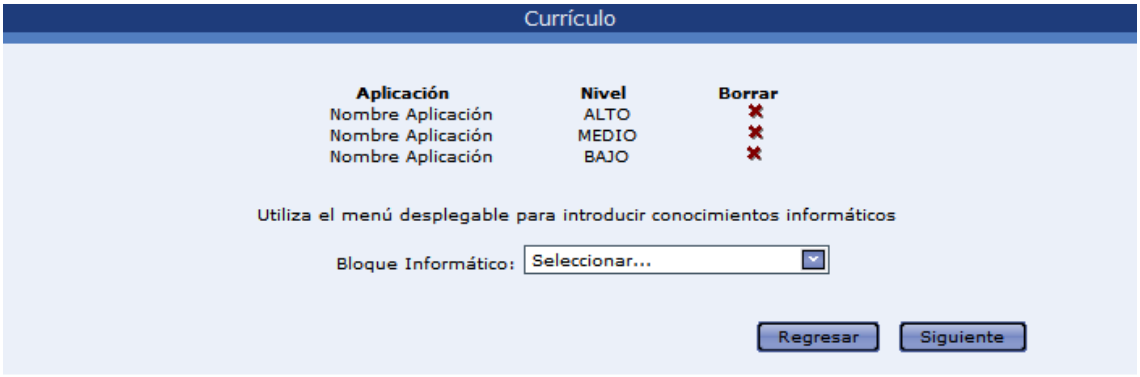


Figura 4.5 Prototipo: Ingresar Currículo - Guardar

El siguiente paso sería ingresar los idiomas que maneja así como el nivel de dominio de cada uno de ellos., tal como se muestra en la figura 4.6.

Figura 4.6 Prototipo: Ingresar Idiomas

Al igual que en el caso de las áreas temáticas, conforme se van ingresando los idiomas se muestran en la parte superior de la pantalla y se cargan nuevamente las listas desplegables para agregar más idiomas, como se puede ver en la figura 4.7.

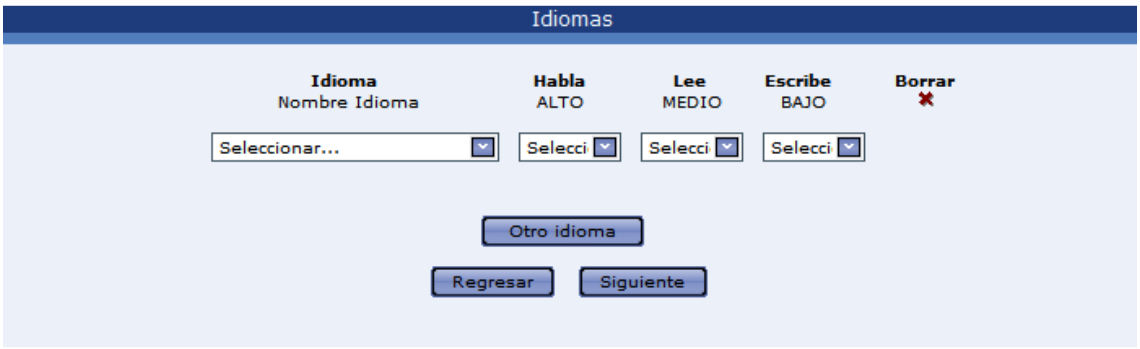


Figura 4.7 Prototipo: Ingresar Currículo – Guardar

Para finalizar con el registro del candidato, se requiere que se ingresen datos referentes a la experiencia y los intereses del mismo, para ello se presento el diseño de la figura 4.8

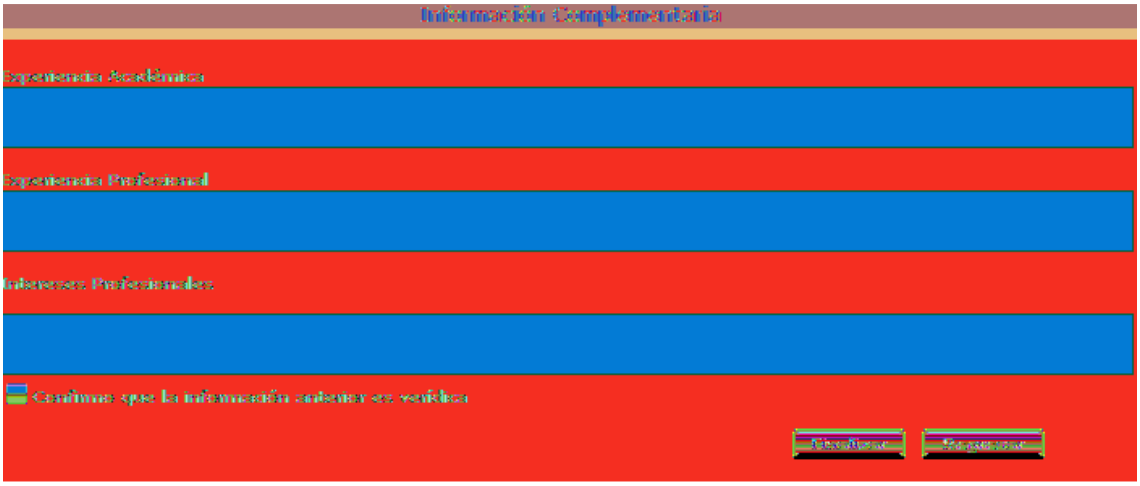


Figura 4.8 Prototipo: Ingresar Información Complementaria

De acuerdo a las reglas de negocio se permitirá la modificación de ciertos campos, una vez que el candidato haya recibido su contraseña para acceder al sistema.

4.2. Análisis y Diseño

El objeto de esta disciplina de trabajo es traducir los requisitos a una especificación que describa como implementar el sistema, es decir, trasladar los requerimientos funcionales a conceptos de software. Las principales actividades de la etapa de análisis y diseño son:

- Transformar los requerimientos en un diseño del sistema a crear.
- Definir una arquitectura para el sistema.
- Adaptar el diseño para que funcione en el ambiente de implementación y así obtener un performance adecuado.

El análisis consiste en obtener una visión del sistema que se preocupa de ver QUÉ hace, de modo que sólo se interesa por verificar los requisitos funcionales.

Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos. Así mismo el diseño permite la implementación del sistema libre de ambigüedades.

4.2.1. Diagrama de Clases

Al contar con el visto bueno de los diseños de pantallas se procedió a realizar el análisis técnico para poder realizar la implementación del sistema.

La finalidad del diagrama de clases es representar todas las entidades involucradas así como las relaciones entre ellas. En un primer acercamiento, se genera un diagrama de clases básico, en el que se representan las clases con su respectivo estereotipo y las relaciones identificadas. Para comenzar definiremos los estereotipos que pueden tener las clases:

- **Clases Boundary**: Manejan comunicaciones entre el entorno del sistema y éste, suelen proporcionar la interfaz con un usuario o con otro sistema, por lo tanto, modelan las interfaces éste.

- **Clases Control**: Modelan el comportamiento específico de uno o varios casos de uso. Se trata de clases que coordinan los eventos necesarios para llevar a cabo el comportamiento que se especifica en el caso de uso, representan su dinámica.

- **Clases Entity**: Modelan información de larga vida y su comportamiento asociado. Este tipo de clase suele reflejar elementos del mundo real o bien, elementos necesarios para realizar tareas internas al sistema. También se denominan clase dominio, ya que suelen tratar con abstracciones de entidades del mundo real.

En la figura 4.9, se representan las clases identificadas por medio del diagrama de clases.

Como se puede apreciar se incluyen las clases Boundary (clases de la vista), las clases Control que procesan la información de acuerdo a las reglas de negocio descritas en el caso de uso y finalmente las clases Entity que se encargan de llevar la información hacia la Base de Datos.

Figura 4.9 Diagrama de Clases: Registrar Solicitud

Una vez que se ha definido el flujo y que el diseño se ha detallado, se procede a visualizar las clases adicionales que se requieren implementar y como será la comunicación entre ellas. En el **Anexo G**, se muestra el diagrama de clases resultado del diseño.

4.2.2. Diagramas de Secuencia

En UML, se tienen dos tipos de diagramas de interacción, ambos pueden ser utilizados para representar el intercambio de mensajes entre objetos, dichos diagramas son:

- Diagramas de Colaboración.
- Diagramas de Secuencia.

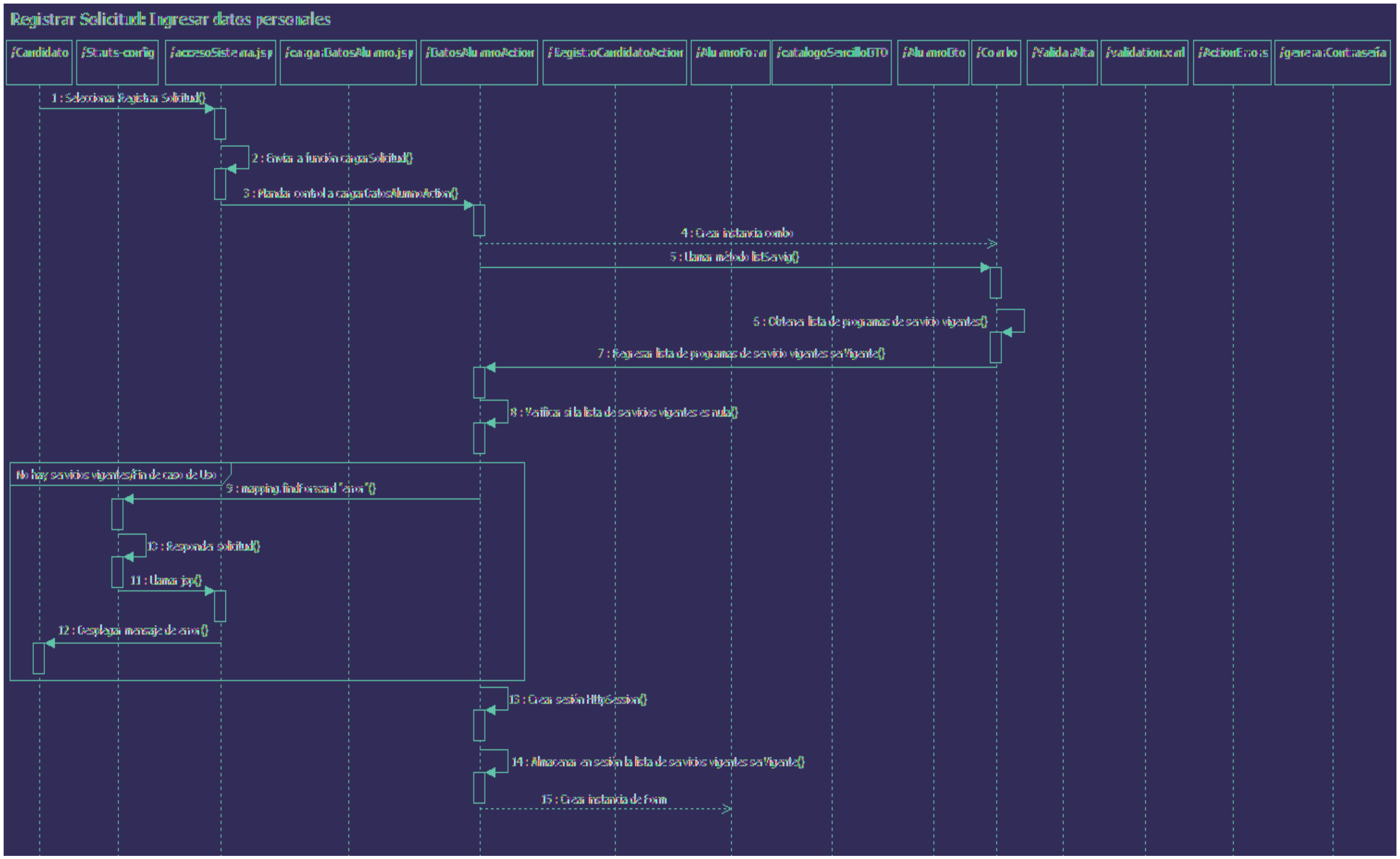
Los diagramas de colaboración, ilustran las interacciones entre objetos, mediante un diagrama de red. Por otro lado, los diagramas de secuencia, se extienden hacia la derecha y muestran claramente la secuencia u orden en el tiempo en el que se intercambian los mensajes entre objetos.

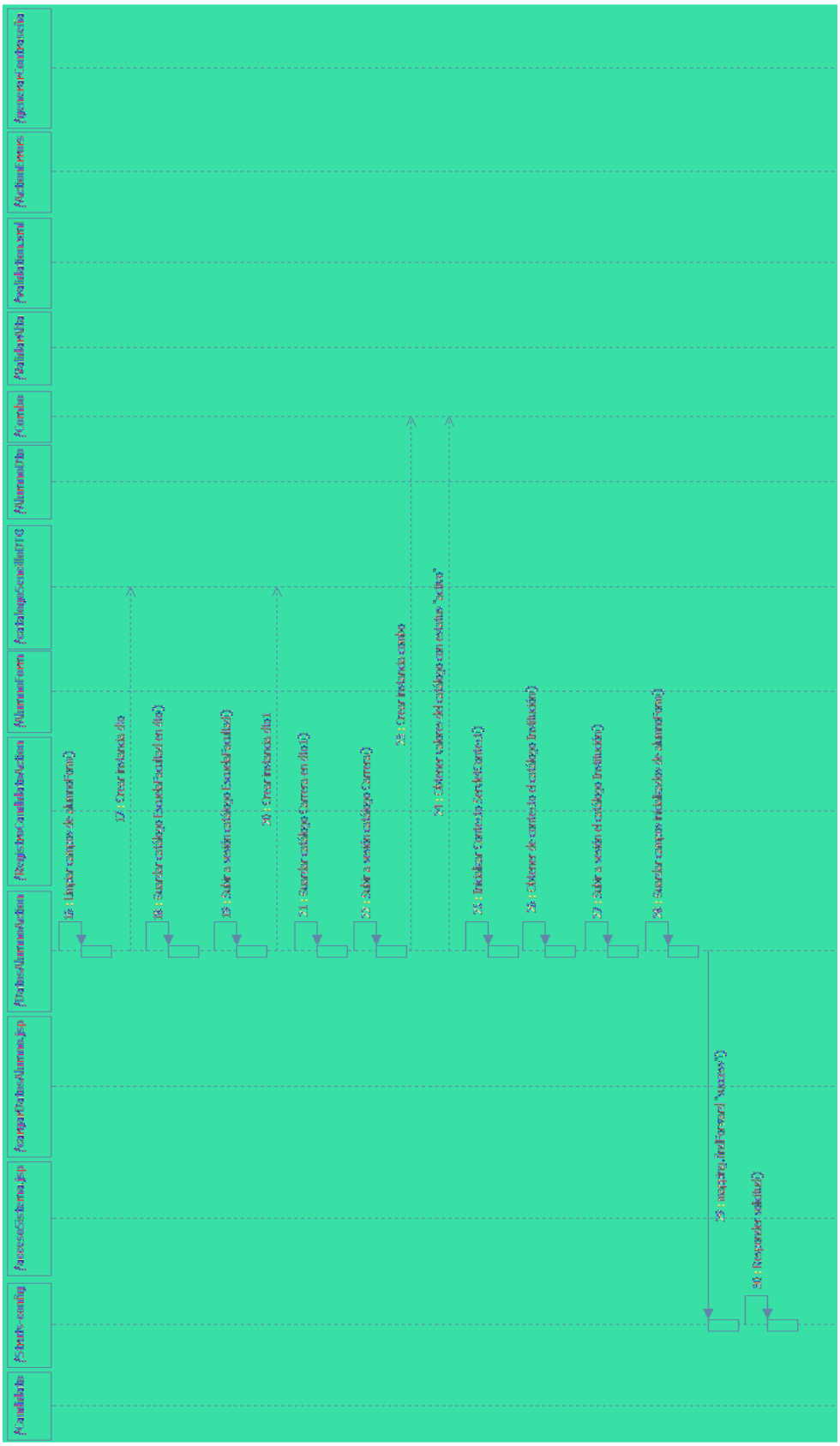
Para el caso de estudio se generaron los siguientes diagramas de secuencia:

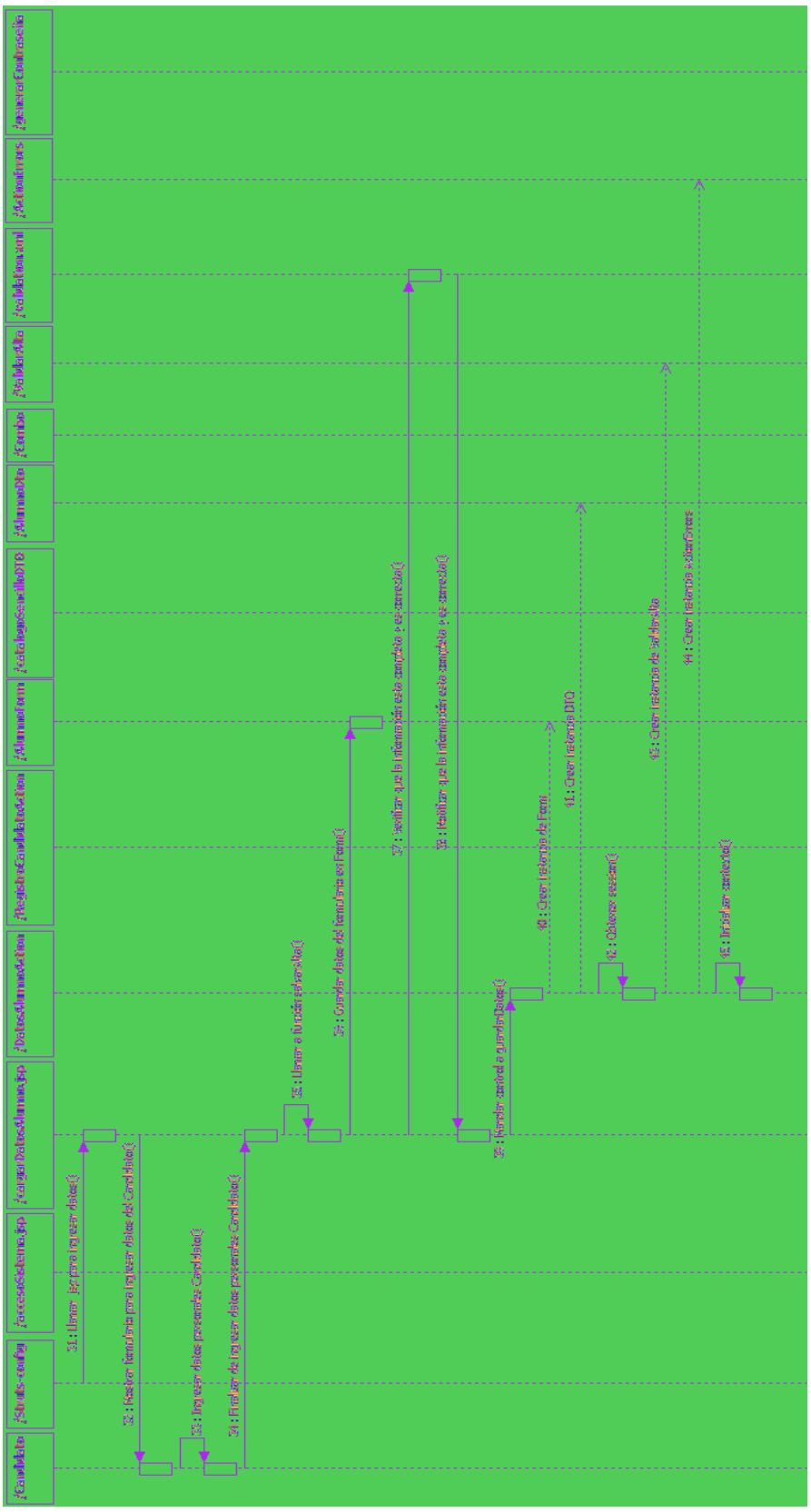
- Ingresar datos personales
- Ingresar servicio
- Ingresar currículo
- Ingresar idiomas
- Ingresar complemento

En cada uno de estos diagramas se detalla el diseño de la solución, en los diagramas se puede apreciar claramente cual será el flujo de los mensajes entre los objetos que se han delineado.

En la figura 4.10, se muestra el diagrama de secuencia correspondiente a ingresar datos personales, el resto se incluyen en el **Anexo H**.


Registrar Solicitud: Ingresar datos personales
/Candidato
/Struts-config
/accesoSistema.jsp
/cargarDatosAlumno.jsp
/DatosAlumnoAction
/RegistroCandidatoAction
/AlumnoForm
/catalogoSencilloDTO
/AlumnoDto
/Combo
/ValidarAlta
/validation.xml
/ActionErrors
/generarContraseña
1 : Seleccionar Registrar Solicitud()
2 : Enviar a función cargarSolicitud()
3 : Mandar control a cargarDatosAlumnoAction()
4 : Crear instancia combo
5 : Llamar método listServig()
6 : Obtener lista de programas de servicio vigentes()
7 : Regresar lista de programas de servicio vigentes serVigente()
8 : Verificar si la lista de servicios vigentes es nula()
No hay servicios vigentes/Fin de caso de Uso
9 : mapping.findForward "error"()
10 : Responder solicitud()
11 : Llamar jsp()
12 : Desplegar mensaje de error()
13 : Crear sesión HttpSession()
14 : Almacenar en sesión la lista de servicios vigentes serVigente()
15 : Crear instancia de Form





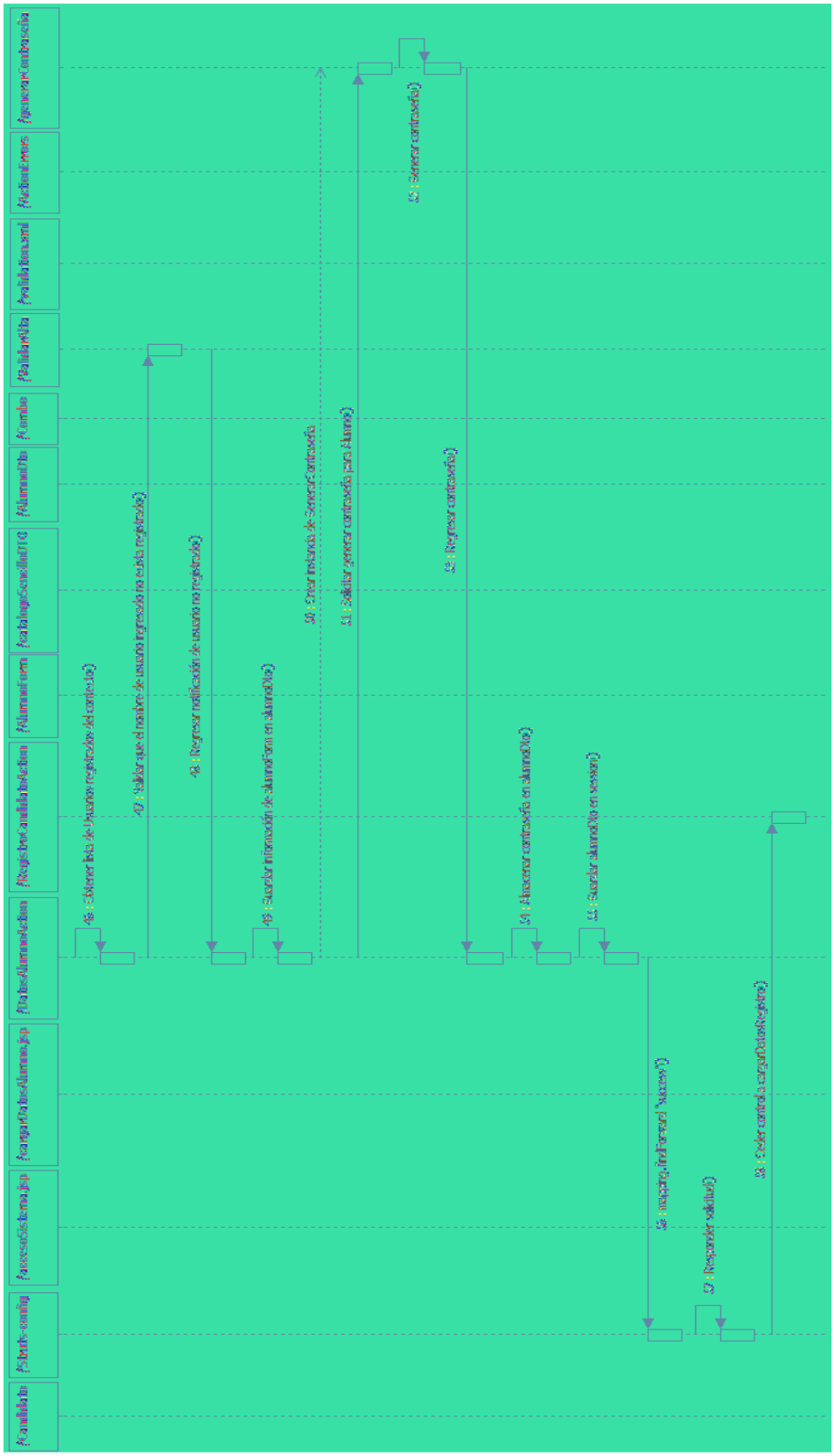


Figura 4.10 Diagrama de Secuencia: Ingresar Datos Personales

4.2.3. Diagrama de Componentes

La mejor manera de tener visualizada la organización de los componentes del sistema es por medio del diagrama de componentes, pues nos presenta un panorama completo de los módulos, paquetes y piezas que contiene cada uno.

En el diagrama de componentes que se presenta en la figura 4.11, se puede apreciar como esta estructurado el módulo de registro de candidato.

Figura 4.11 Diagrama de Componentes

4.3. Diseño de la Base de Datos

El buen diseño de la base de datos radica en la funcionalidad que ésta soportará para el sistema. El modelo entidad – relación, ofrece una manera clara de identificar las entidades involucradas y como están relacionadas. Posteriormente se puede representar el diseño, en términos de una base de datos relacional.

En general el diseño de la base de datos es una tarea compleja, que abarca desde la identificación de las entidades, el diseño de los programas que realizarán el acceso a los datos, hasta la seguridad para controlar dicho acceso, esta sección se enfocará a describir los conceptos básicos que se emplearon para diseñar la base de datos Control de Prestadores.

4.3.1. Modelo entidad – relación

El modelo entidad – relación (E-R), se basa en la percepción del mundo real que consiste en un conjunto de entidades y las relaciones que pueden darse entre esas entidades. Una entidad es identificable mediante atributos, por ejemplo la entidad alumno, puede identificarse mediante los atributos, nombre, edad, RFC, CURP, entre otros, además de un identificador único que permita distinguirla de posibles entidades con los mismos atributos.

La estructura lógica general de la base de datos se puede expresar gráficamente mediante un diagrama E-R, conformado por los siguientes elementos:

- Rectángulos: Representan las entidades.
- Elipses: Representan los atributos.
    - Elipses dobles: Atributos multivaluados.
    - Elipses discontinuas: Atributos derivados.
- Rombos: Representan las relaciones entre las entidades.
- Líneas: Para unir entidades con relaciones y entidades con atributos.

Cabe mencionar que pueden existir diferentes tipos de atributos:

- *Atributos simples y compuestos*: Los atributos compuestos son aquellos que pueden subdividirse, por ejemplo el atributo nombre puede estar compuesto por nombre de pila, apellido paterno y apellido materno.
- *Atributos monovaluados y multivaluados*. Aquellos atributos que tienen un único valor para una entidad en concreto son monovaluados. Los atributos que tienen un conjunto de valores para una entidad concreta son multivaluados.
- *Atributos derivados*. El valor para este tipo de atributos, se puede obtener a partir del valor de otros atributos. El valor de estos atributos no se almacena, se calcula cada que es necesario.

Cardinalidad

La cardinalidad, expresa el número de entidades a las que otra entidad se puede asociar mediante un conjunto de relaciones. Las cardinalidades que pueden aplicar son las siguientes:

- Uno a uno
- Uno a varios
- Varios a uno
- Varios a varios

Del Modelo E-R a tablas

1. Todas las entidades se convierten en tablas.
2. Todas las propiedades o atributos se convierten en columnas.
   - Atributos clave se convierten en PK.
   - Atributos compuestos se descomponen en columna por cada atributo.
   - Atributos derivados se convierten en default.
   - Atributos multivaluados se convierten en otra entidad con una relación M a M con su entidad original.
3. Los vínculos o relaciones 1 a M pasan la PK del lado de 1 al lado de M como FK.
4. Los vínculos o relaciones 1 a 1 se tratan como vínculos 1 a M dependiendo de la entidad más importante.
5. Los vínculos M a M se convierten en una tercera tabla formada con las PK de las entidades participantes y como FK. Su llave primaria PK será la suma de las FK.
6. Para tipos y subtipos los subtipos heredan la PK del súper tipo la cual será la FK del subtipo.

En el **Anexo I** se presenta el diagrama E-R correspondiente.

4.3.2. Modelo Relacional

El modelo relacional utiliza un conjunto de tablas para representar datos además de las relaciones entre ellos. Una base de datos relacional consiste en un conjunto de tablas en las cuales cada fila representa una relación entre un conjunto de valores.

Los elementos que componen el modelo relacional son los siguientes:

1. *Entidad*: Una entidad puede ser una persona, objeto, lugar o evento identificado en forma única.
2. *Relación*. Dentro del modelo relacional, una tabla es una relación y es un conjunto de tuplas (registros) y atributos (campos).
3. Cada columna contiene valores relativos al mismo atributo y cada valor de una columna de la tabla debe ser simple, es decir debe tener un solo valor.
   - Las tuplas están en desorden.
   - Los atributos están en desorden.
   - Las tuplas no se repiten.
4. *Atributo*. Son las características de una entidad.
5. *Tupla*. Es un conjunto de valores que componen un renglón de la relación. Es el equivalente a una instancia de un registro.
6. *Grado de una tupla*. Es el número de atributos que tiene una tupla.
7. *Cardinalidad*. Es el número de tuplas de una relación.
8. *Dominio*. Es el conjunto de valores válidos para un atributo.

Aunado a los conceptos anteriores, existe uno que es primordial y es el de llave. En el modelo relacional podemos tener diferentes tipos de llaves, que se describen brevemente a continuación:

a) *Llave Primaria (PK).* Es el atributo que identifica de manera única a un registro o renglón de una tabla. Las condiciones para que un campo sea una llave primaria son:
   - La llave primaria debe tener un valor único para cada renglón de una tabla.
   - No puede contener valores nulos.
   - No se pueden realizar cambios sobre los valores de la llave primaria.
   - Puede estar formada por más de una columna, es decir puede ser una llave compuesta.

b) *Llave foránea (FK).* Una llave foránea es la llave primaria de una tabla, pero al mismo tiempo forma parte de otra tabla como atributo.

Reglas de Cood

En 1984, el creador del modelo relacional, Cood, creo las 12 reglas de Cood, éstas tienen como objetivo establecer los lineamientos a los que deberían apegarse los sistemas manejadores de datos que sean relacionales. Las reglas son:

1. ***Regla de la información.*** Toda información contenida en una base de datos relacional será representada a nivel lógico con valores dentro de una tabla.
2. ***Regla del acceso garantizado.*** Todos los valores de una tabla, deben poder ser accedidos mediante la combinación del nombre de la tabla, el nombre del atributo y el valor de la llave primaria.
3. ***Regla del tratamiento sistemático del valor nulo.*** Los valores nulos, deben ser soportados por el DBMS para representar información faltante o inaplicable, independientemente del tipo de dato.
4. ***Regla del catálogo dinámico basado en línea***. El diccionario de datos debe tener la misma estructura, estar contenido en tablas, y se va a usar de igual forma que las tablas de usuario.
5. ***Regla del sub-lenguaje de datos completo.*** Un solo lenguaje es usado para comunicarse con el DBMS y el usuario. Esto significa que debe haber un lenguaje con una sintaxis bien definida que pueda ser usado para administrar completamente la base de datos, por ejemplo SQL.
6. ***Regla de la actualización de vistas.*** Las vistas deben poder ser actualizadas por el sistema.
7. ***Regla de inserción, actualización y eliminación de alto nivel.*** El manejador de base de datos debe ser capaz de permitir que en una sola operación se pueda afectar a más de un registro de una o más tablas.
8. ***Regla de la independencia física de datos.*** Significa, que las bases de datos deben contemplar una independencia física de los datos al medio de almacenamiento, en otras palabras, al cambiarse el medio de almacenamiento no se debe afectar la estructura lógica de la base de datos.

9. ***Regla de la independencia lógica de datos.*** Las bases de datos deben contemplar la independencia lógica de los datos, es decir cuando un elemento de una tabla, sea actualizado, no se debe afectar la información ya existente en la base de datos.
10. ***Regla de la independencia de integridad.*** Todas las restricciones de integridad deben estar definidas desde la base de datos, mediante el sub-lenguaje de datos.
11. ***Regla de la independencia de distribución.*** El sistema debe contemplar el hecho de poder particionar una base de datos en diferentes dispositivos de almacenamiento físico, permitiendo que el usuario la vea como una sola.
12. ***Regla de la no subversión.*** Si un sistema relacional tiene un lenguaje de bajo nivel, es decir que afecte a un solo registro a la vez, este lenguaje no deberá ser usado para violar las reglas de integridad y restricciones establecidas en un lenguaje de alto nivel.

Normalización.

La normalización es una descomposición de información sin pérdida, que se aplica a las bases de datos para evitar problemas de redundancia y así mejorar el rendimiento de la base de datos. Se implementa mediante las formas normales e implica dividir los atributos de una relación en dos subconjuntos sin que se pierda la información de la relación original.

Formas normales

- *Primera Forma Normal (1FN)*. Se dice que una entidad R esta en primera forma normal, si los dominios de todos los atributos de R son atómicos, es decir una relación está en primera forma normal cuando a cada atributo le corresponde un solo valor.

- *Segunda Forma Normal (2FN)*. Una relación esta en segunda forma normal, si esta en 1FN y se han eliminado las dependencias funcionales. En otras palabras, una relación está en 2FN si esta en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal.

- *Tercera Forma Normal (3FN).* Una relación esta en tercera forma normal si y solo si está en 2FN y los atributos no clave no tienen dependencias transitivas.

Una dependencia transitiva es aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave.

Desnormalización.

Existen ocasiones en las que la normalización no es lo más conveniente, debido a que el costo de la obtención de información a partir de los datos relacionados es considerablemente más grande que el costo de su mantenimiento. Este costo considera la frecuencia de manipulación de datos y operaciones de consulta, el tiempo de estas operaciones y el tiempo en que se quiera obtener la respuesta.

Un ejemplo de beneficio de la desnormalización podría ser el siguiente: En muchas aplicaciones se utilizan tablas que contienen códigos, para proveer descripciones en las

consultas. Se podría elegir mantener la descripción en el registro padre en vez del código, para evitar estas uniones durante las consultas.

En la figura 4.12, se ilustra el modelo lógico de la base de datos del Sistema Control de Prestadores.

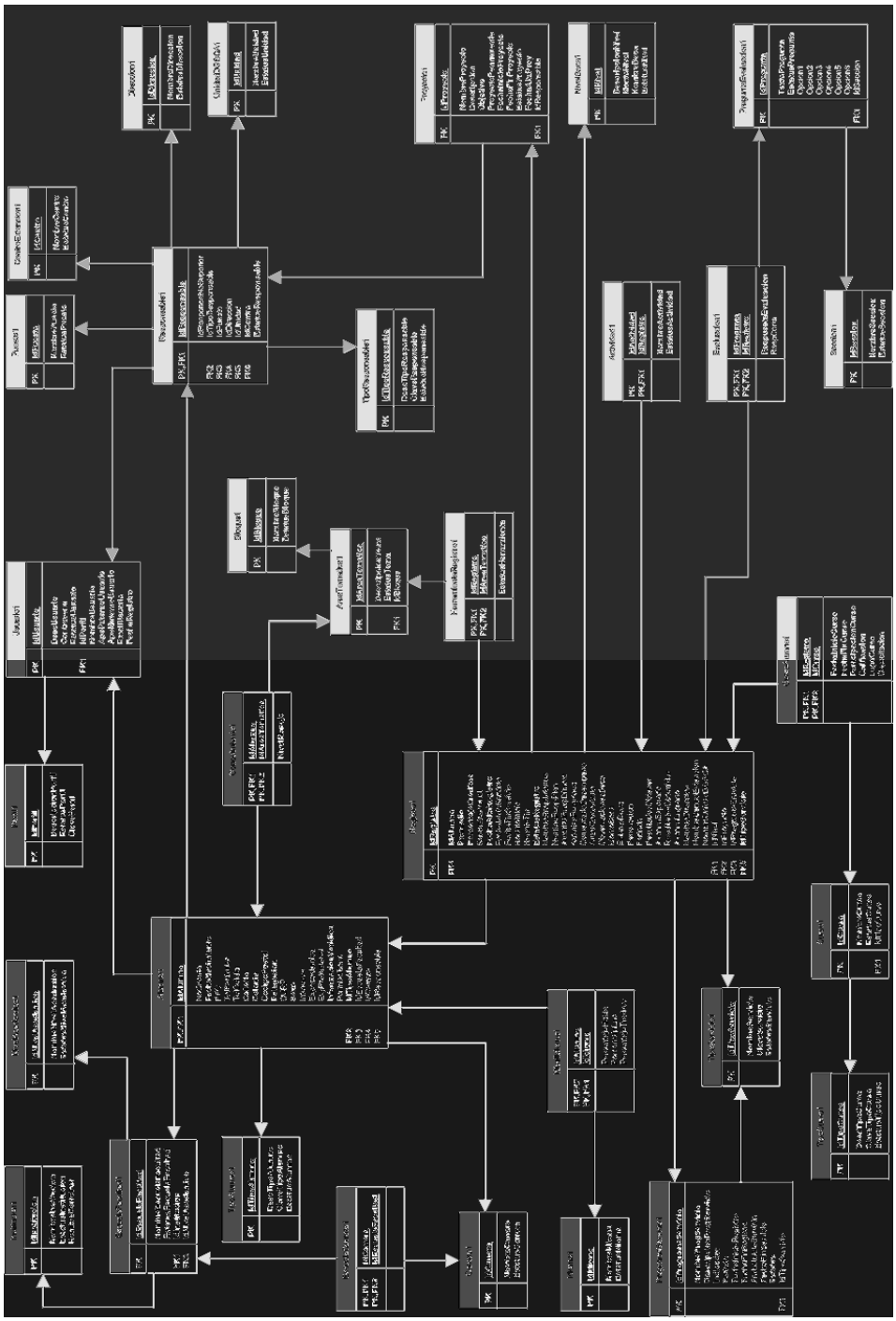


Figura 4.12 Modelo Lógico

En el **Anexo J**, se incluye el modelo físico y el diccionario de datos se encuentra en el **Anexo K**.

Cabe mencionar que en el alcance de este proyecto no se contempla un alto grado de seguridad del sistema, ya que será implementado a futuro como una ampliación a este proyecto. Por el momento el sistema únicamente contará con la funcionalidad para autenticar usuarios de acuerdo a un perfil de manera que se les permita el acceso únicamente a la funcionalidad definida por reglas de negocio. Así mismo se hará uso de sesiones para garantizar la integridad de los datos de un usuario.

Respecto a la seguridad de la base de datos y del sitio en general, ésta será responsabilidad del departamento de administración de servidores de DGSCA.

5. Construcción

En esta fase se profundiza en el diseño de los componentes y de manera iterativa se van añadiendo las funcionalidades al software a medida que se construyen y prueban, permitiendo a la vez ir incorporando cambios.

La fase de construcción se conforma de tres disciplinas RUP que son:

- Implementación
- Pruebas
- Despliegue

Al final de esta fase, todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto, así como la documentación necesaria para que sea entregada a los usuarios.

5.1. Implementación

Esta disciplina se encarga de pasar de los resultados de la fase de diseño a desarrollar el sistema en términos de componentes tales como archivos fuente, binarios, ejecutables scripts, entre otros. Los objetivos que se plantean para esta fase son:

- Implementar las clases encontradas dentro del diseño.
- Asignar los componentes ejecutables a los nodos del diagrama de despliegue.
- Probar los componentes individualmente e integrarlos en uno o más ejecutables.
- Integrar los componentes en el sistema siguiendo un enfoque incremental.

En este apartado, se realizará la explicación de la implementación de la funcionalidad para almacenar los datos personales del alumno utilizando algunos fragmentos de código.

Las piezas involucradas son las siguientes:

1. AlumnoForm.java
2. AlumnoDto.java
3. DatosAlumnoAction.java
4. SalvarCandidatoAction.java
5. struts-config.xml
6. cargarDatosAlumno.jsp
7. validation.xml
8. GenerarContrasena.java
9. AlumnoDao.java
10. Conexion.java

Inicialmente se debe construir un form y un dto, que serán los encargados de transportar los datos a través de las diferentes capas. En el caso del form, éste es el bean encargado de llevar los datos desde la vista hacia el controlador.

```
/* AlumnoForm.java*/
/*Esta clase contiene los métodos set() y get() que se utilizan para trasladar la
información desde la vista hacia el controlador*/

package dgsca.unam.registro.form;

import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.validator.ValidatorForm;

public class AlumnoForm extends ValidatorForm {

    private String nombreAlumno;
    private String apePatAlumno;
    private String apeMatAlumno;
    private int idUsuario;
    private String nombreUsuario;
    private String contrasena;
    private int idPerfilAlumno;
    private String clavePerfilAlumno;

    public String getNombreAlumno() {
        return nombreAlumno; }

    public void setNombreAlumno(String nombreAlumno) {
        this.nombreAlumno = nombreAlumno;}

    public String getApePatAlumno() {
        return apePatAlumno;}

    public void setApePatAlumno(String apePatAlumno) {
        this.apePatAlumno = apePatAlumno;}

    public String getApeMatAlumno() {
        return apeMatAlumno;}

    public void setApeMatAlumno(String apeMatAlumno) {
        this.apeMatAlumno = apeMatAlumno;}

    public int getIdUsuario() {
        return idUsuario;}

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;}

    public String getNombreUsuario() {
        return nombreUsuario;}

    public void setNombreUsuario(String nombreUsuario) {
        this.nombreUsuario = nombreUsuario;}

    public String getContrasena() {
        return contrasena;}

    public void setContrasena(String contrasena) {
        this.contrasena = contrasena;}

    public int getIdPerfilAlumno() {
        return idPerfilAlumno;}
```

```
    public void setIdPerfilAlumno(int idPerfilAlumno) {
        this.idPerfilAlumno = idPerfilAlumno;}

    public String getClavePerfilAlumno() {
        return clavePerfilAlumno;}

    public void setClavePerfilAlumno(String clavePerfilAlumno) {
        this.clavePerfilAlumno = clavePerfilAlumno;}

    public String getValidationKey(ActionMapping mapping, HttpServletRequest request) {
        String parameterName = mapping.getParameter();
        String parameterValue = request.getParameter(parameterName);
        String AlumnoForm = mapping.getName();
        return AlumnoForm + "-" + parameterValue;}
}
```

En el caso del ***dto***, será el bean que pase los datos del controlador hacia el modelo, tiene la misma estructura que el ***form***.

```
/*AlumnoDto.java*/

/*La clase AlumnoDto, contiene los métodos set() y get() que se utilizan para trasladar la
información desde el controlador hacia la base de datos*/

package dgsca.unam.alumno.dto;
import java.io.Serializable;

public class AlumnoDto implements Serializable {

    /** Creates a new instance of AlumnoDto */
    public AlumnoDto() {
    }

    private String nombreAlumno;
    private String apePatAlumno;
    private String apeMatAlumno;
    private String nombreCompleto;
    private int idUsuario;
    private String nombreUsuario;
    private String contrasena;
    private int idPerfilAlumno;
    private String clavePerfilAlumno;
    private int idAlumno;

    public String getNombreAlumno() {
        return nombreAlumno;}

    public void setNombreAlumno(String nombreAlumno) {
        this.nombreAlumno = nombreAlumno;}

    public String getApePatAlumno() {
        return apePatAlumno;}

    public void setApePatAlumno(String apePatAlumno) {
        this.apePatAlumno = apePatAlumno;}

    public String getApeMatAlumno() {
        return apeMatAlumno;}

    public void setApeMatAlumno(String apeMatAlumno) {
        this.apeMatAlumno = apeMatAlumno;}

    public int getIdUsuario() {
        return idUsuario;}

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;}
```

```
    public String getNombreUsuario() {
        return nombreUsuario;}

    public void setNombreUsuario(String nombreUsuario) {
        this.nombreUsuario = nombreUsuario;}

    public String getContrasena() {
        return contrasena;}

    public void setContrasena(String contrasena) {
        this.contrasena = contrasena;}

    public int getIdPerfilAlumno() {
        return idPerfilAlumno;}

    public void setIdPerfilAlumno(int idPerfilAlumno) {
        this.idPerfilAlumno = idPerfilAlumno;}

    public String getClavePerfilAlumno() {
        return clavePerfilAlumno;}

    public void setClavePerfilAlumno(String clavePerfilAlumno) {
        this.clavePerfilAlumno = clavePerfilAlumno;}

    public int getIdAlumno() {
        return idAlumno;}

    public void setIdAlumno(int idAlumno) {
        this.idAlumno = idAlumno;}
}
```

Cuando desde el explorador llamamos a uno de los métodos, en este caso al de Ingresar Solicitud, Struts inicia el controlador, quien posteriormente enviará la petición hacia la vista correspondiente. En el siguiente código el método invocado será ***cargarDatosAlumno,*** este método inicializa los campos y hace el ***forward*** a la vista.

```
/*DatosAlumnoAction.java*/
/*Esta clase permite inicializar el formulario que se usará para ingresar los datos
personales, tiene los métodos para obtener la información del formulario y almacenarla en la
base de datos, así mismo tiene los métodos para consultar la información de un alumno */

package dgsca.unam.admonSolicitudes.action;
import dgsca.unam.admonSistema.dto.CatalogoSencilloDto;
import dgsca.unam.alumno.dao.AlumnoDao;
import dgsca.unam.alumno.dao.RegistroDao;
import dgsca.unam.alumno.dto.AlumnoDto;
import dgsca.unam.alumno.dto.CurriculoDto;
import dgsca.unam.alumno.dto.RegistroDto;
import dgsca.unam.registro.form.AlumnoForm;
import dgsca.unam.usuario.dao.UsuarioDao;
import dgsca.unam.util.Combo;
import dgsca.unam.util.GenerarCont;
import dgsca.unam.util.ValidarAlta;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.actions.DispatchAction;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionForward;
```

```
public class DatosAlumnoAction extends DispatchAction {

    private static String result = "success";

        public ActionForward cargarDatosAlumno(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        result = "success";
        AlumnoForm alumnoForm= (AlumnoForm)form;

        Combo combo = new Combo();
        ServletContext servletContext = null;
        HttpSession session =request.getSession(true);
        servletContext = this.getServlet().getServletContext();
        ActionErrors errores = new ActionErrors();
        int serv=1;
        /*Se verifica si existen servicios activos
        Primero se recupera del contexto los servicios que se tienen dados de alta.
        En caso de que existan servicios, a través del objeto combo, se acceso al método
listServig() para ver si se tienen servicios vigentes para el día que se está registrando el
alumno. Si no existen servicios se regresa mensaje en el AcctionMessage(error) para indicar
que por el momento no existen servicios disponibles */

        List listServicios =(List)servletContext.getAttribute("servicioGeneral");
        if(listServicios != null){
             List progServicio=combo.listServig(listServicios);
             if(progServicio.size()==0){
                 serv=0;
             }
             else{
                  session.setAttribute("serVigente",progServicio);
             }
        }
        else{
            serv=0;
        }
        if(serv==0){
            errores.add(ActionErrors.GLOBAL_MESSAGE,new
ActionMessage("errors.SinServicio"));
            saveErrors(session, errores);
        }

        /*Si existen servicios se limpian los datos del form para que se muestre el
formulario al alumno */
        else{
            alumnoForm.setNombreAlumno("");
            alumnoForm.setApePatAlumno("");
            alumnoForm.setApeMatAlumno("");
            alumnoForm.setNombreAlumno("");
            alumnoForm.setCalleAlumno("");
            alumnoForm.setClavePerfilAlumno("");
            alumnoForm.setCodPostalAlumno("");
            alumnoForm.setCurpAlumno("");
            alumnoForm.setDelegacionAlumno("");
            alumnoForm.setDescCarreraAlumno("");
            alumnoForm.setEmailAlumno("");
            alumnoForm.setFechaNacAlumno("");
            alumnoForm.setInteresesAlumno("");
            alumnoForm.setRfcAlumno("");
            alumnoForm.setNombreUsuario(null);
            alumnoForm.setTelCelAlumno("");
            alumnoForm.setTelPartAlumno("");
            alumnoForm.setSexoAlumno("");
            alumnoForm.setColoniaAlumno("");
            alumnoForm.setIdInstitucion(0);
            alumnoForm.setNoCuentaAlumno("");

            /* Debido a que Escuela Facultad depende de Institución, se guarda el
comboEsculaFacultad un arreglo con un id uno y un nombre EscuelaFacultad */
```

```
            List lista = new ArrayList();
            CatalogoSencilloDto dto = null;
            dto = new CatalogoSencilloDto();
            dto.setIdentificador(1);
            dto.setNombre("EscuelaFacultadad");
            dto.setIdCompuesto(1);
            lista.add(dto);
            session.setAttribute("comboEscuelaFacultad",lista);

            /* Debido a que Carrera depende de Escuela Facultad, se guarda el comboCarrera
un arreglo con un id uno y un nombre Carrera */
            List lista1 = new ArrayList();
            CatalogoSencilloDto dto1 = null;
            dto1 = new CatalogoSencilloDto();
            dto1.setIdentificador(1);
            dto1.setNombre("Carrera");
            dto1.setIdCompuesto(1);
            lista1.add(dto1);
            session.setAttribute("comboCarrera",lista1);
            /*Se guarda en sesión las Instituciones que estan activas, primero se recupera
del contexto las instituciones que estan dadas de alta y posteriormene a traves del objeto
combo con el metodo listaCombo() se recuperan las instituciones que estan activas*/
            List lis1=(List)servletContext.getAttribute("institucion");
            List lis2=combo.listaCombo(lis1);
            session.setAttribute("comboInstitucion",lis1);

            alumnoForm.setFuncion("ALTA");
        }

        return mapping.findForward(result);
    }
    /*Método para cargar Escuela Facultad de acuerdo a la Institución seleccionada y Carrera
de acuerdo a la Escuela Facultad*/
    public ActionForward cambiarCombo(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        result = "success";
        AlumnoForm alumnoForm = (AlumnoForm)form;
        HttpSession session = request.getSession(true);
        Combo combo = new Combo();
        ServletContext servletContext = null;
        servletContext = this.getServlet().getServletContext();

        /* Si se guarda en funcion2 cambioEscuelaFac, se recupera del contexto las escuelas
y facultades después se toma del form el id de la Institución y a través del objeto combo
con el método listaComboDep se recuperan las escuelas facultades que pertenecen a la
Institución seleccionada y se guarda en sesión*/
        if(alumnoForm.getFuncion2().equals("CambioEscuelaFac")){
            int idInst;
            List lista1=(List)servletContext.getAttribute("escuelaFacultad");
            idInst=alumnoForm.getIdInstitucion();
            List  lista =combo.listaComboDep(lista1,idInst);
            session.setAttribute("comboEscuelaFacultad",lista);
            alumnoForm.setIdCarrera(0);
        }

         /* Si se guarda en funcion2 CambioCarrera, se recupera del contexto las carreras,
después se toma del form el id de la Escuela/Facultad y a través del objeto combo con el
método listaComboDep se recuperan las carreras que pertenecen a la escuelas/facultade
seleccionada*/
        if(alumnoForm.getFuncion2().equals("CambioCarrera")){
            int idEscuelaFac;
            List lista1=(List)servletContext.getAttribute("carreraEscuela");
            idEscuelaFac=alumnoForm.getIdEscuelaFacAlumno();
            List  lista =combo.listaComboDep(lista1,idEscuelaFac);
            session.setAttribute("comboCarrera",lista);
        }
```

```
        if(alumnoForm.getFuncion().equals("ALTA")){
            result="success";
        }

        return mapping.findForward(result);
    }

     public ActionForward guardarDatos(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        result = "success";
        AlumnoForm alumnoForm=(AlumnoForm)form;
        int val;
        AlumnoDto dto =new AlumnoDto();
        HttpSession session= request.getSession(true);
        ValidarAlta valAlta = new ValidarAlta();
        ActionErrors errores = new ActionErrors();
        Combo combo = new Combo();
        /*Se verifica si el usuario que ingreso el alumno no esta dado de alta para otro
usuario. Se recupera del contexto los usuarios y a través del objeto valAlta con el método
validaCatSentillo se verifica q no este dado de alta*/
        ServletContext servletContext = null;
        servletContext = this.getServlet().getServletContext();
        List lista1=(List)servletContext.getAttribute("usuarios");
        if(lista1!=null){
             val = valAlta.validaCatSencillo(lista1,alumnoForm.getNombreUsuario());
        }
        else val=0;
//Si está dado de alta, se regresa mensaje de error para que el alumno cambie el usuario
        if(val==1){
            errores.add(ActionErrors.GLOBAL_MESSAGE,new
ActionMessage("errors.usuarioRepetido"));
            saveErrors(session, errores);
        }
        //Si no está dado de alta, se guarda la información del alumno en sesión.
        else {
            dto.setNombreAlumno(alumnoForm.getNombreAlumno().trim().toUpperCase());
            dto.setApePatAlumno(alumnoForm.getApePatAlumno().trim().toUpperCase());
            dto.setApeMatAlumno(alumnoForm.getApeMatAlumno().trim().toUpperCase());
            dto.setEmailAlumno(alumnoForm.getEmailAlumno().trim());
            dto.setFechaNacAlumno(alumnoForm.getFechaNacAlumno());
            dto.setRfcAlumno(alumnoForm.getRfcAlumno().trim().toUpperCase());
            dto.setCurpAlumno(alumnoForm.getCurpAlumno().trim().toUpperCase());
            dto.setSexoAlumno(alumnoForm.getSexoAlumno());
            dto.setNombreUsuario(alumnoForm.getNombreUsuario().trim());
            dto.setCalleAlumno(alumnoForm.getCalleAlumno().trim().toUpperCase());
            dto.setColoniaAlumno(alumnoForm.getColoniaAlumno().trim().toUpperCase());
            dto.setCodPostalAlumno(alumnoForm.getCodPostalAlumno().trim());
            dto.setDelegacionAlumno(alumnoForm.getDelegacionAlumno().trim().toUpperCase());
            dto.setTelCelAlumno(alumnoForm.getTelCelAlumno().trim());
            dto.setTelPartAlumno(alumnoForm.getTelPartAlumno().trim());
            dto.setIdInstitucion(alumnoForm.getIdInstitucion());
            dto.setIdEscuelaFacAlumno(alumnoForm.getIdEscuelaFacAlumno());
            dto.setIdCarrera(alumnoForm.getIdCarrera());
            dto.setNoCuentaAlumno(alumnoForm.getNoCuentaAlumno().trim());

             if(alumnoForm.getIdInstitucion() != 0){
                List lista2=(List)session.getAttribute("comboInstitucion");
                String institucion=combo.nombreSimple(lista2,alumnoForm.getIdInstitucion());
                dto.setDescInstitucion(institucion);
            }
            else{
                dto.setDescInstitucion(" ");}
            if(alumnoForm.getIdEscuelaFacAlumno() != 0){
                List lista3=(List)servletContext.getAttribute("escuelaFacultad");
                String
escuela=combo.nombreSimple(lista3,alumnoForm.getIdEscuelaFacAlumno());
                dto.setDescEscuelaFacAlumno(escuela);
            }
```

```
            else{
                 dto.setDescEscuelaFacAlumno(" ");
            }
            if(alumnoForm.getIdCarrera() != 0){
                List lista4=(List)servletContext.getAttribute("carrera");
                String carrera=combo.nombreSimple(lista4,alumnoForm.getIdCarrera());
                dto.setDescCarreraAlumno(carrera);
            }
            else{
                dto.setDescCarreraAlumno(" ");
            }

            //Generar contraseña
            GenerarCont pass=new GenerarCont();
            String password=pass.OptenerCont();
            dto.setContrasena(password);

            //Guardar datos en sesión
            session.setAttribute("alumnoDet",dto);
            result="siguiente";

        }

        return mapping.findForward(result);

    }

     public ActionForward mostrarDatos(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        result = "success";
        AlumnoForm alumnoForm=(AlumnoForm)form;
        AlumnoDto dto =new AlumnoDto();

        HttpSession session = request.getSession(true);
        dto=(AlumnoDto)session.getAttribute("alumnoDet");

        alumnoForm.setNombreAlumno(dto.getNombreAlumno());
        alumnoForm.setNombreAlumno(dto.getApePatAlumno());
        alumnoForm.setApeMatAlumno(dto.getApeMatAlumno());
        alumnoForm.setNombreUsuario(dto.getNombreUsuario());
        alumnoForm.setFuncion("ALTA");

        return mapping.findForward(result);
    }

    public ActionForward detDatosPersonales(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

            result = "consulta";
            AlumnoForm alumnoForm=(AlumnoForm)form;
            AlumnoDto dto=new AlumnoDto();
            HttpSession session =request.getSession(true);
            dto=(AlumnoDto)session.getAttribute("alumnoDet");

            alumnoForm.setNombreAlumno(dto.getNombreAlumno());
            alumnoForm.setApePatAlumno(dto.getApePatAlumno());
            alumnoForm.setApeMatAlumno(dto.getApeMatAlumno());
            alumnoForm.setNombreUsuario(dto.getNombreUsuario());
            alumnoForm.setFuncion("INICIO");
            return mapping.findForward(result);
    }
}
```

Como se puede apreciar el result trae el valor "***success***" dicho valor esta definido en el ***struts-config***.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>

<form-beans>
   <form-bean name="alumnoForm" type="dgsca.unam.registro.form.AlumnoForm"/>
</form-beans>

    <global-exceptions>

    </global-exceptions>

    <global-forwards>
        <forward name="welcome"  path="/Welcome.do"/>
    </global-forwards>

    <action-mappings>

<!-- == Action para Administración de Alumnos ==-->

        <action name="alumnoForm" path="/dispath/datosAlumno" scope="session"
type="dgsca.unam.admonSolicitudes.action.DatosAlumnoAction"   validate="true"
input="/admonAlumno/datosPersonales/cargarDatosAlumno.jsp" parameter="method">
            <forward contextRelative="true" name="success"
path="/admonAlumno/datosPersonales/cargarDatosAlumno.jsp"/>
            <forward contextRelative="true" name="siguiente"
path="/dispat/registroCandidato.do?method=cargarDatosRegistro"/>
            <forward contextRelative="true" name="consulta"
path="/admonAlumno/datosPersonales/consDatosPersonales.jsp"/>
            <forward contextRelative="true" name="modificacion"
path="/dispath/datosAlumno.do?method=detDatosPersonales"/>
            <forward contextRelative="true" name="complemento"
path="/admonAlumno/curriculo/curriculoIdiomas.jsp"/>
            <forward contextRelative="true" name="continuar"
path="/dispat/salvarCandidato.do?method=cargarInfo"/>
            <forward contextRelative="true" name="consIdioma"
path="/admonAlumno/curriculo/consIdiomas.jsp"/>
            <forward contextRelative="true" name="failure" path="/admonSistema/error.jsp"/>
        </action>

        <action path="/Welcome" forward="/welcomeStruts.jsp"/>
    </action-mappings>

    <controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>

    <message-resources parameter="com/myapp/struts/ApplicationResource"/>

</struts-config>
```

De manera que la vista que tendremos es ***cargarDatosAlumno.jsp***

```
<%@page contentType="text/html"%>
<%@page pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic"%>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html:html>

<head>

    <link rel="stylesheet" href="../js/dhtmlgoodies_calendar.css" media="screen"></link>
```

```
    <script type="text/javascript" src="<html:rewrite
page="/js/dhtmlgoodies_calendar.js"/>"></script>
    <link href="../css/estilo.css" rel="stylesheet" type="text/css"/>
    <title>Datos Personales</title>

</head>
<script language="javascript">
    <!--
    function salvarAlta(){
        document.forms[0].action = "<html:rewrite
action="/dispath/datosAlumno.do?method=guardarDatos"/>";
        document.forms[0].submit();
        document.forms[0].className ='espera';
    }

    function regresarDatos(){
        document.forms[0].action = "<html:rewrite
action="/dispath/datosAlumno.do?method=detDatosPersonales"/>";
        document.forms[0].submit();
    }

    //-->
</script>

<% String perfilUsuario = (String)request.getSession().getAttribute("perfilUsuario"); if
(perfilUsuario == null) perfilUsuario ="-"; %>
    <body bgcolor="#F0F0F0">
        <table width="700"  class="tabla" cellspacing="0" align="center">
            <html:form action="/dispath/datosAlumno.do?method=guardarDatos">
                <html:hidden name="alumnoForm" property="funcion"/>
                <html:hidden name="alumnoForm" property="funcion2"/>

                <tr valign="baseline" class="celda">
                    <td height="10" colspan="3"><div align="center" class="titulo">Datos
Personales</div></td>
                </tr>
                <tr valign="baseline" class="celda2">
                    <td height="5" colspan="3"></td>
                </tr>
                <tr valign="baseline">
                    <td height="10" colspan="3"></td>
                </tr>
                <tr class="cuerpo">

<td>  Nombre(s):         <html:text
property="nombreAlumno" maxlength="150" size="50"/></td>
                </tr>
                <tr class="cuerpo">
                    <td>  Apellido Paterno: <html:text
property="apePatAlumno" maxlength="150" size="50"/></td>
                </tr>
                <tr class="cuerpo">
                    <td>  Apellido Materno: <html:text
property="apeMatAlumno" maxlength="150" size="50"/></td>
                </tr>
                <!--INI 20090210 Validación para cuando sea candidato-->
                <% if( perfilUsuario.equals("-")) { %>
                <tr>
                </tr>
                <tr><td class="cuerpo">  A continuación ingresa un nombre de
usuario para tener acceso al sistema </td></tr>
                <tr>
                    <tr class="cuerpo">
                        <td>  Nombre de Usuario: <html:text
property="nombreUsuario" maxlength="150" size="15"/></td>
                    </tr>
                </tr>
                <% } %>
                 <!--FIN-->

                <tr><td> </td></tr>
```

```
                <!--INI 20090210 Mostrar solo cuando sea candidato-->
                <% if( perfilUsuario.equals("-")) { %>
                <tr>
                    <td align="right" colspan="3">
                        <input type="button" name="guardar" value="Siguiente"
class="BUTTON3" ONCLICK="salvarAlta();"/>
                    </td>
                </tr>
                <% } %>
                <!--FIN-->

                 <tr>
                    <td align="right" colspan="3">
                        <input type="button" name="back" value="Cancelar" class="BUTTON3"
ONCLICK="regresarDatos();"/>
                    </td>
                </tr>

                <% } %>
            </html:form>
        </table>

        <table align="center">
            <html:messages id="error">
                <tr>
                    <td colspan="3">
                        <li class="error"><bean:write name="error"/></li>
                    </td>
                </tr>
            </html:messages>
        </table>
    </body>
    </html:html>
```

Este ***jsp***, muestra los campos que se deben ingresar y los botones de siguiente y cancelar, dependiendo de botón seleccionado, se pasará el control a un método del ***dispatch***. Al tomar los datos mediante el form, se llama al método ***getValidationKey*** quien hará uso del ***ValidatorForm*** para verificar que los datos que se hayan ingresado estén correctos en cuanto a sintaxis.

El ValidationForm, permite crear reglas que validen si un campo puede permitir espacios vacios, si debe ser llenado únicamente con letras o con números, también permite definir reglas para validar fechas.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE form-validation PUBLIC
          "-//Apache Software Foundation//DTD Commons Validator Rules Configuration
1.1.3//EN"
          "http://jakarta.apache.org/commons/dtds/validator_1_1_3.dtd">

<form-validation>

<formset>
<!—Validaciones para AlumnoForm, se valida sintáxis y datos obligatorios-->
    <form name="alumnoForm-guardarDatos">
        <field property="nombreAlumno" depends="required,mask">
            <arg0 key="Nombre de Alumno" resource="false"/>
            <var><var-name>mask</var-name><var-value>^[a-zA-Z áéíóúÁÉÍÓÚñÑ]*$</var-
value></var>
        </field>
        <field property="apePatAlumno" depends="required,mask">
            <arg0 key="Apellido Paterno" resource="false"/>
```

```
            <var><var-name>mask</var-name><var-value>^[a-zA-Z áéíóúÁÉÍÓÚñÑ]*$</var-
value></var>
        </field>
        <field property="apeMatAlumno" depends="required,mask">
            <arg0 key="Apellido Materno" resource="false"/>
            <var><var-name>mask</var-name> <var-value>^[a-zA-Z áéíóúÁÉÍÓÚñÑ]*$</var-
value></var>
        </field>
        <field property="nombreUsuario" depends="required,mask">
            <arg0 key="Nombre de Usuario" resource="false"/>
            <var><var-name>mask</var-name> <var-value>^[a-z A-Z 0-9 áéíóúÁÉÍÓÚñÑ \.]*$</var-
value> </var>
        </field>
     </form>
</formset>

    </form-validation>
```

Al regresar el control al dispacth, mediante la petición de guardar, se observa que se toman los datos del form y se pasan al dto y el forward tiene el valor de “siguiente”.

```
public ActionForward guardarDatos(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        result = "success";
        AlumnoForm alumnoForm=(AlumnoForm)form;
        int val;
        AlumnoDto dto =new AlumnoDto();
        HttpSession session= request.getSession(true);
        ValidarAlta valAlta = new ValidarAlta();
        ActionErrors errores = new ActionErrors();
        Combo combo = new Combo();

        ServletContext servletContext = null;
        servletContext = this.getServlet().getServletContext();
        List lista1=(List)servletContext.getAttribute("usuarios");
        if(lista1!=null){
             val = valAlta.validaCatSencillo(lista1,alumnoForm.getNombreUsuario());
        }
        else val=0;

        if(val==1){
            errores.add(ActionErrors.GLOBAL_MESSAGE,new
ActionMessage("errors.usuarioRepetido"));
            saveErrors(session, errores);
        }
        else {
            dto.setNombreAlumno(alumnoForm.getNombreAlumno().trim().toUpperCase());
            dto.setApePatAlumno(alumnoForm.getApePatAlumno().trim().toUpperCase());
            dto.setApeMatAlumno(alumnoForm.getApeMatAlumno().trim().toUpperCase());
            dto.setNombreUsuario(alumnoForm.getNombreUsuario().trim());

            //Generar contraseña
            GenerarCont pass=new GenerarCont();
            String password=pass.OptenerCont();
            dto.setContrasena(password);

            //Guardar datos en sesión
            session.setAttribute("alumnoDet",dto);
            result="siguiente"; }

        return mapping.findForward(result);

     }
```

Desde el método guardarDatos se invoca al método GenerarCont, que se muestra a continuación.

```
/*GenerarCont.java*/

package dgsca.unam.util;

public class GenerarCont {

    /** Creates a new instance of GenerarCont */
    public GenerarCont() {
    }

     //Método para generar contraseña, a partir de aplicar el método random*/
    public String OptenerCont(){
        int charsCont = 6;
        int i =0;
        char characterToShow='-';
        String password="";
        String elegibleChars = "ABCDEFGHJKLMPQRSTUVWXY23456789abcdefhjkmnpqrstuvwxy";
        char[] chars = elegibleChars.toCharArray();
        StringBuffer finalString = new StringBuffer();
        while( i < charsCont )
        {
            double randomValue = Math.random();
            int randomIndex = (int) Math.round(randomValue * (chars.length - 1));
            characterToShow = chars[randomIndex];
            password=password+characterToShow;
            i=i+1;
        }
        return password;
    }
}
```

Al finalizar de ingresar los datos del Candidato, se procede a almacenarlos en la base de datos.

```
/*
 * SalvarCandidatoAction.java
 *
*/

package dgsca.unam.admonSolicitudes.action;

import dgsca.unam.admonSistema.dao.CatalogosDAO;
import dgsca.unam.alumno.dao.AlumnoDao;
import dgsca.unam.alumno.dto.AlumnoDto;
import dgsca.unam.registro.form.AlumnoForm;
import dgsca.unam.util.Combo;
import dgsca.unam.util.Resultado;
import java.util.Iterator;
import java.util.List;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.actions.DispatchAction;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionForward;

public class SalvarCandidatoAction extends DispatchAction {

    /* forward name="success" path="" */
   private static String result = "success";

     public ActionForward guardarInfo(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        result = "success";
```

```
        int ideAlumno;
        int ideRegistro;
        int indicador;
        AlumnoForm alumnoForm=(AlumnoForm)form;
        ActionErrors errores = new ActionErrors();
        HttpSession session= request.getSession(true);
        ServletContext servletContext = null;
        servletContext = this.getServlet().getServletContext();

        AlumnoDto alumnoDto =new AlumnoDto();
        AlumnoDao dao= AlumnoDao.getInstance();
        alumnoDto=(AlumnoDto)session.getAttribute("alumnoDet");
        session.setAttribute("alumnoDet",alumnoDto);

        //Se insertan los datos del alumno y se regresa el id de alumno.
        ideAlumno=dao.insertAlumno(alumnoDto);
        if(ideAlumno!=0){
            regDto.setIdAlumno(ideAlumno);

            //Se actualiza el contexto de los usuarios.
            Resultado res= new Resultado();
            CatalogosDAO catDao = CatalogosDAO.getInstance();
            res=catDao.selectUsuario();
            List list1=res.getLista();
            servletContext.setAttribute("usuarios",list1);

            //Se inserta el registro y se regresa el id del registro.
            ideRegistro=dao.insertRegistro(regDto);
            session.setAttribute("alumnoDet",alumnoDto);

    public ActionForward finRegistro(ActionMapping mapping, ActionForm  form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        result = "fin";
        HttpSession session =request.getSession(true);
        session.invalidate();
        return mapping.findForward(result);
    }
}
```

Mediante AlumnoDao se hace la petición a la conexión a la base de datos y se aplica en método para insertar la información.

```
/*
 * AlumnoDao.java
 *
*/

package dgsca.unam.alumno.dao;

import dgsca.unam.admonSistema.dto.CatalogoSencilloDto;
import dgsca.unam.alumno.dto.AlumnoDto;
import dgsca.unam.util.Conexion;
import dgsca.unam.util.Resultado;
import java.io.IOException;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import javax.naming.NamingException;

public class AlumnoDao {

        //Crear instancia única del objeto de acceso a datos
        private static AlumnoDao instance;
```

```
        //Fuente de datos a SQL Server
        private Conexion sqlDataSource;

    public AlumnoDao() {
        sqlDataSource=new Conexion();
    }

    /*Regresa la instancia unica del objeto de acceso a datos*/
    public static AlumnoDao getInstance() {
        if(instance == null) {
            instance = new AlumnoDao();
        }
        return instance;
    }

  //****************************** Datos alumno ********************************* //
    public int insertAlumno(final AlumnoDto alumnoDto) throws NamingException,
ClassNotFoundException, IOException {
        int indicador =0;

        Connection conn = null;
        CallableStatement cstmt = null;
        int idAlumno=0;
/*A partir de los datos que el controlador almacenó en el dto, se arma la cadena que se
pasará al store procedure para insetar los datos del alumno*/
        try {
            conn = sqlDataSource.getConnection();
            conn.setAutoCommit(false);
            cstmt = conn.prepareCall("Exec ins_Alumno ?,?,?,?,?,?,?,?,?,?");
            cstmt.setString(1,alumnoDto.getNombreAlumno());
            cstmt.setString(2,alumnoDto.getApePatAlumno());
            cstmt.setString(3,alumnoDto.getApeMatAlumno());
            cstmt.setString(20,alumnoDto.getNombreUsuario());
            cstmt.setString(21,alumnoDto.getContrasena());
            indicador= cstmt.executeUpdate();
            conn.commit();

            cstmt = conn.prepareCall("Exec qry_Alumno ?");
            cstmt.setString(1,alumnoDto.getNombreUsuario());
            final ResultSet rs = cstmt.executeQuery();

            if(rs.next()){
                idAlumno=rs.getInt(1);
            }
        } catch (java.sql.SQLException e){
            System.out.println("Error de SQL al insertar Alumno");
            e.printStackTrace();
        }
        finally {
            try{
                if( conn != null )
                conn.close();
            } catch(Exception e) {
                System.out.println("Error en sqldesde DAo: "+ e.getMessage());
            }
        }
        return (int)idAlumno;
    }
}
```

La clase Conexión, es una clase genérica para realizar la comunicación con la base de datos.

```
/*
 * Conexion.java
*/

package dgsca.unam.util;
import javax.sql.*;

import java.sql.*;
```

```
import java.sql.Connection;
import java.io.*;
import javax.naming.*;
import java.sql.SQLException;

public class Conexion {

    Connection con;

        /** Creates a new instance of Conexion */
    public  Conexion() {
  }
    public java.sql.Connection getConnection() throws NamingException,
ClassNotFoundException,java.io.IOException{
        System.out.println("En la clase conexion");
    try {
      Context ctx = new InitialContext();
      if(ctx != null ){
                System.out.println("antes de datasource");
        DataSource ds  = (DataSource)ctx.lookup ("java:comp/env/jdbc/SYSPRES");
        System.out.println("despúes de datasource");
        if( ds != null){
            con = ds.getConnection();
        }
      }

    }
    catch(java.sql.SQLException sqle) {
     sqle.printStackTrace();
    }
    return con;
    }
  }
```

5.2. Pruebas

Al hablar del concepto de pruebas podemos referirnos a aquellas comprobaciones hechas a todos los elementos que se producen durante la construcción, para ver que cumplen con los requisitos y con los estándares de calidad definidos para el proyecto.

El objetivo de esta actividad es verificar el correcto funcionamiento de la estructura del sistema que se va formando con los módulos implementados, esto para poder asegurar la integración de las piezas, detectar errores oportunamente y en caso de ser necesario iniciar el trabajo de adecuación.

Un sistema es aceptable cuando:

- Hace lo que se acordó que debía hacer en las especificaciones.
- No hace lo que no debe hacer.

Como es sabido existen diferentes tipos de pruebas para validar el correcto funcionamiento de un sistema, entre ellas destacan:

- *Pruebas Unitarias.* Permiten asegurar el funcionamiento de un módulo del sistema, por separado. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas.

Las pruebas unitarias presentan las siguientes desventajas:

- ✓ No descubren todos los defectos del código.
- ✓ No permiten determinar problemas de integración de componentes.
- ✓ No se pueden determinar fácilmente, todas las posibles entradas de datos.

- *Pruebas Integrales.* Sirven para comprobar el funcionamiento del sistema, es decir, de todos los módulos que lo componen. Este tipo de pruebas también se conocen como pruebas de sistema.

- *Pruebas UAT.* Son las pruebas de aceptación del usuario, éstas se realizan una vez que se hace la entrega del sistema implementado por completo y son el último tipo de pruebas a realizarse para que el cliente certifique que realmente se le entrega lo que solicitó.

5.2.1. Casos de Prueba

Para tener el control adecuado de las pruebas, se deben diseñar los casos de prueba a manera de guía para no perder de vista ningún punto importante a considerarse dentro de las validaciones al sistema. Los casos de prueba, deben estar basados en los casos de uso, ya que en estos se plasma tanto el flujo principal como las excepciones que pudieran presentarse durante la ejecución.

A continuación se incluye el caso de prueba generado para el caso de uso CU-0010 Registrar Solicitud.

MATRIZ FUNCIONAL

ID Escenario	Caso de uso	Descripción corta del escenario
1	Registrar Solicitud	El candidato solicita registrar su información en el sistema.
2	EX01	Error, no existen periodos de servicio activos.
3	EA01	Error en la Información de datos personales (incorrecta o requerida).
4	EA02	Error en la Información de solicitud de servicio (incorrecta o requerida).
5	EA03	Error, se selecciono un idioma, no se ingreso el nivel de manejo de idioma.
6	EA04	Sin confirmación de veracidad.
7	EX02	No se almaceno la información.

Tabla 5.1 índice de casos de prueba

ID Escenario	Paso	Acción del usuario	Resultados esperados
1	1	Pulsa botón "Registrar"	Muestra la pantalla: --Datos Personales-- Nombre (s): [ ] Apellido Paterno: [ ] Apellido Materno: [ ] e-mail: [ ] Fecha de nacimiento: [ ] RFC: [ ] CURP: [ ] Sexo: [ ] Nombre Usuario: [ ] --Domicilio Particular-- Calle y Número:[ ] Delegación: [ ] Código Postal: [ ] Teléfono Particular: [ ] Teléfono Celular: [ ] --Datos Escolares-- Combo: Institución Combo: Escuela o Facultad Combo: Carrera Número de cuenta: [ ]
1	2	Ingresa información --*Datos Personales*-- Nombre (s): [Elsa] Apellido Paterno: [Aparicio] Apellido Materno: [Martinez] e-mail: [elyaparicio@gmail.com] Fecha de nacimiento: [10/08/1984] RFC: [AMAE841008] CURP: [AMAE841008] Sexo: [Femenino] Nombre Usuario: [apael01] --*Domicilio Particular*-- Calle y Número:[Av. Revolución #1510] Colonia: [San Angel] Delegación: [Álvaro Obregón] Código Postal: [010300] Teléfono Particular: [52599694] Teléfono Celular: [ ] --*Datos Escolares*-- Institución: UNAM Escuela o Facultad: FACULTAD DE INGENIERÍA Carrera: INGENIERÍA EN COMPUTACIÓN Número de cuenta: [990138660] Pulsa botón "Siguiente"	El sistema muestra la pantalla de Servicio: Semestre:[ ] Promedio:[ ] Porcentaje de créditos:[ ] Combo: Horario disponible Combo: Tipo de Servicio

1	3	Ingresa información Semestre:[9] Promedio:[9.0] Porcentaje de créditos:[87.0] Selecciona Horario disponible:16:00 Tipo de Servicio: SERVICIO SOCIAL	Muestra en pantalla Currículo: Combo: Bloque informativo
1	4	Selecciona Bloque Informativo: BASES DE DATOS	Muestra en pantalla : Lista: Aplicación radio botón: Alto radio botón: Medio radio botón: Bajo
1	5	Selecciona Alto en Sql Server Pulsa botón "Guardar"	Muestra pantalla: **Aplicación Nivel Borrar** Sql Server ALTO x Lista: Bloque Informativo
1	**6**	Pulsa botón "Siguiente"	Muestra pantalla de Idiomas Lista: Idiomas Lista: Habla Lista: Lee Lista: Escribe
1	7	Selecciona Idioma: INGLES Habla: MEDIO Lee: ALTO Escribe: ALTO Pulsa botón "Siguiente"	Muestra pantalla de Información complementaria Intereses Personales:[ ] Experiencia Académica:[ ] Experiencia Profesional:[ ] Confirmar que la información es verídica
1	8	Ingresa: Intereses Personales:[Análisis y Diseño de sistemas mediante RUP y UML] Experiencia Académica:[Proyectos escolares en Java]	Mostrar pantalla con la información ingresada: --Datos Personales-- **Nombre (s)**: Elsa **Apellido Paterno**: Aparicio **Apellido Materno**: Martinez **e-mail**: elyaparicio@gmail.com **Fecha de nacimiento**: 10/08/1984 **RFC**: AMAE841008 **CURP**: AMAE841008 **Sexo**: Femenino **Nombre Usuario:** apael01 **Password:** 4kWvXh --Domicilio Particular-- **Calle y Número**: Av. Revolución #1510 **Colonia**: San Angel **Delegación**: [Álvaro Obregón] **Código Postal**: 010300 **Teléfono Particular**: 52599694 **Teléfono Celular:**

1	8	Experiencia Profesional:[Ninguna] Confirma que la información es verídica Pulsar botón "Finalizar"	--Datos Escolares-- **Institución**: UNAM **Escuela o Facultad**: FACULTAD DE INGENIERÍA **Carrera**: INGENIERÍA EN COMPUTACIÓN **Número de cuenta**: [990138660]
2	1	Pulsar botón "Registrar"	Muestra mensaje de error: [***Estimado Alumno, lo sentimos por el momento no contamos con programas de servicio activos para registrar solicitudes, favor de intentarlo en otra ocasión***]
3	1	Ingresa información **--Datos Personales--** Nombre (s): [Elsa] Apellido Paterno: [Aparicio] Apellido Materno: [Martinez] e-mail: [ ] Fecha de nacimiento: [10/08/1984] RFC: [AMAE841008] CURP: [ ] Sexo: [ ] Nombre Usuario: [apael01] **--Domicilio Particular--** Calle y Número:[Av. Revolución #1510] Colonia: [San Angel] Delegación: [Álvaro Obregón] Código Postal: [010300] Teléfono Particular: [52599694] Teléfono Celular: [ ] **--Datos Escolares--** Institución: UNAM Escuela o Facultad: FACULTAD DE INGENIERÍA Carrera: INGENIERÍA EN COMPUTACIÓN Número de cuenta: [990138660] Pulsa botón "Siguiente"	Muestra mensaje de error: [**e-mail es requerido**] [**Sexo es requerido**]
4	1	Ingresa información Semestre:[9] Promedio:[ ] Porcentaje de créditos:[87.0] Selecciona Horario disponible: Tipo de Servicio: SERVICIO SOCIAL	Muestra mensaje de error: [**Promedio es requerido**] [**Hora de inicio es requerido**]

5	1	Selecciona Idioma: INGLES Habla: Lee: Escribe: Pulsa botón "Siguiente"	Muestra mensaje de error: [**Indicar el nivel que se tiene para leer**] [**Indicar el nivel que se tiene para Escribir**] [**Indicar el nivel que se tiene para Hablar**]
6	1	Ingresa: Intereses Personales:[Análisis y Diseño de sistemas mediante RUP y UML] Experiencia Académica:[Proyectos escolares en Java] Experiencia Profesional:[Ninguna] No confirma que la información es verídica Pulsar botón "Finalizar"	Muestra mensaje de error: [**Falta aceptar que la información enviada es verídica**]
7	1	Ingresa: Intereses Personales:[Análisis y Diseño de sistemas mediante RUP y UML] Experiencia Académica:[Proyectos escolares en Java] Experiencia Profesional:[Ninguna] Confirma que la información es verídica Pulsar botón "Finalizar"	Muestra mensaje de error: [**Error al guardar la información**]

Tabla 5.2 Casos de Prueba: Registrar Solicitud

5.2.2. Pruebas Unitarias

Como evidencia de Pruebas Unitarias, se realizó una corrida exitosa de la funcionalidad de un catálogo:

Caso de Prueba: Alta de un registro en un catálogo.

Al solicitar la Administración de un catálogo, si éste no tiene datos, el sistema envía un mensaje indicando que no se encontraron datos, como se muestra en la figura 5.1.

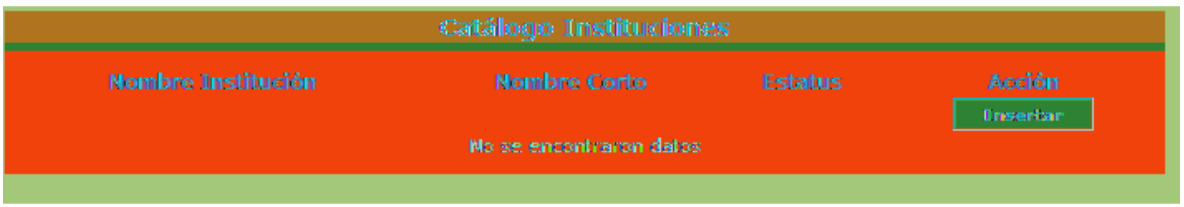


Figura 5.1 Catálogo vacío

Si se solicita ingresar un registro en un catálogo, el sistema habilita los campos que se requieran llenar, como se ve en la figura 5.2.

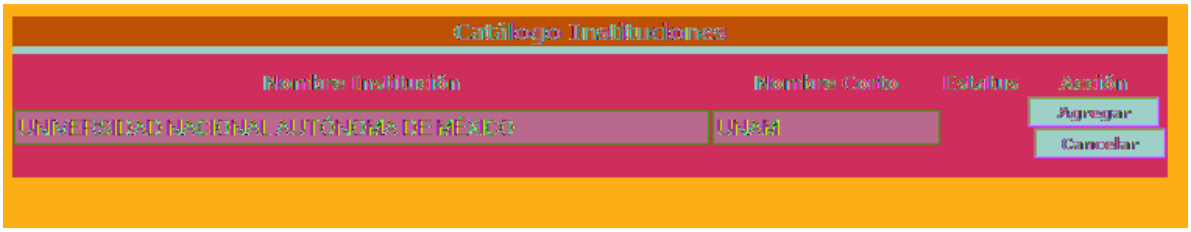


Figura 5.2 Insertar datos en catálogo

Al hacer clic en el botón “Aceptar”, el sistema envía una pregunta de confirmación de alta, como se muestra en la figura 5.3.

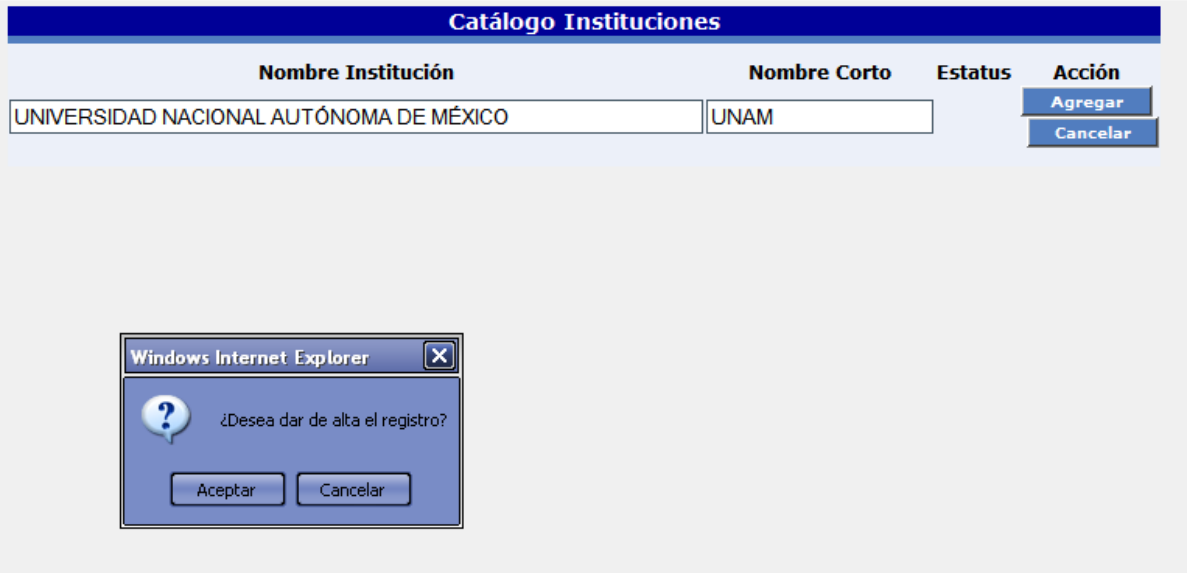


Figura 5.3 Confirmar alta en catálogo

Una vez que se dio clic en el botón “Aceptar”, el sistema despliega la información almacenada, como se puede apreciar en la figura 5.4. En el caso de haberse realizado un alta, en la columna acción, se incluye un tache (X), que posteriormente servirá para inhabilitar el registro si así se desea.

Figura 5.4 Consulta de datos insertados en catálogo

Caso de Prueba: Consulta de los registros en un catálogo.

Al solicitar desde el menú de administración de catálogos, la consulta de alguno de ellos, en caso de existir datos, el sistema desplegará los registros almacenados, esto se puede apreciar en la figura 5.5.

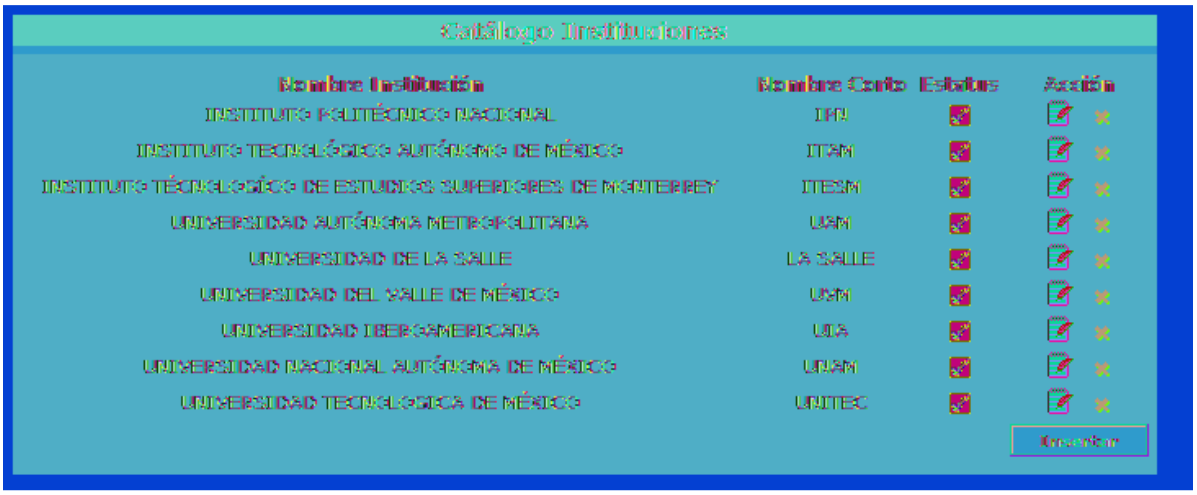


Figura 5.5 Consulta de datos en catálogo

Caso de Prueba: Baja de un registro en un catálogo.

Al dar clic en el ícono X, el sistema despliega la pregunta de confirmación de baja del registro, como se puede apreciar en la figura 5.6.

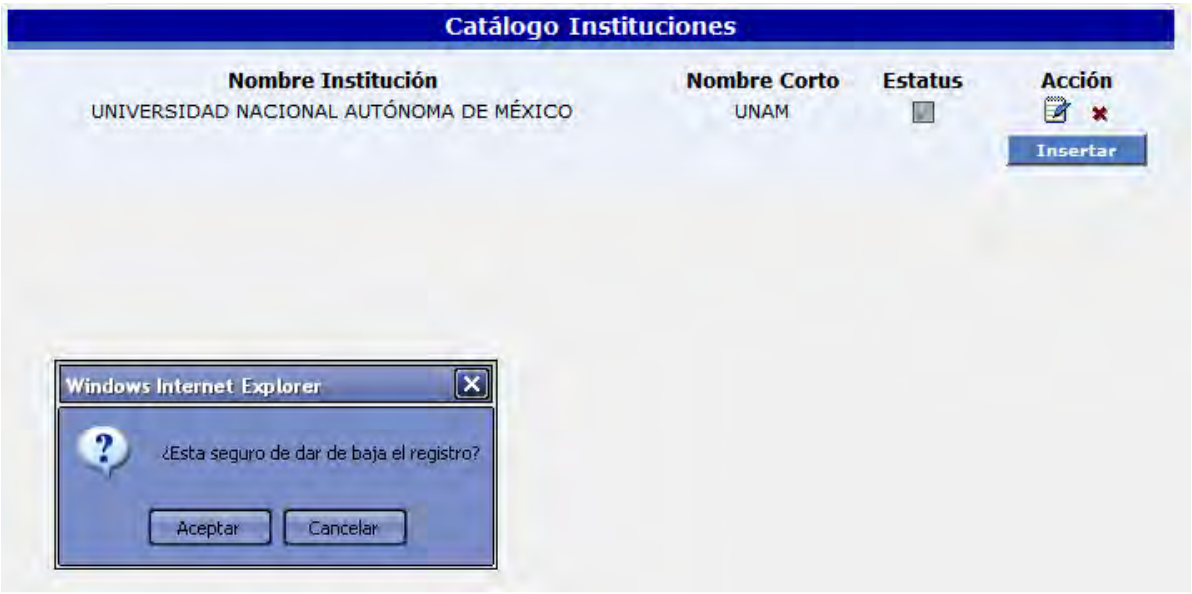


Figura 5.6 Baja de registro en catálogo

En la figura 5.7, se aprecia que al dar clic en el botón “Aceptar” el sistema, despliega la información contenida en el catálogo, pero al tratarse de un registro “dado de baja” en la columna acción mostrará el icono correspondiente a la “Activación”.

Figura 5.7 Consulta de estatus en catálogo

Caso de Prueba: Reactivación de un registro en un catálogo.

Igual que en el caso de la baja del registro, al dar clic en el icono de activación, el sistema despliega la pregunta de confirmación de reactivación del registro, como se aprecia en la figura 5.8.

Al hacer clic en aceptar, se cambia el estatus del registro en la base de datos a activo.

Figura 5.8 Reactivación de datos en catálogo

5.2.3. Pruebas Integrales

Se presenta como evidencia de pruebas una corrida exitosa del registro de un candidato en el sistema de control de prestadores de DGSCA, basada en el caso de prueba de la **Tabla 5.2 Registrar Solicitud.**

En esta prueba, se verifica el correcto funcionamiento del módulo de administración de catálogos y el módulo de registro de servicio, debido a que para registrar un candidato en el sistema, se requieren las consultas a los catálogos de nivel académico, escuela–facultad, carrera, tipo servicio, bloque, área temática e idioma para alimentar los combos que se le presentarán al alumno, así mismo se prueba la correcta comunicación con la base de datos y el envío de un correo electrónico al candidato.

Caso de Prueba: Realizar el registro de un Candidato.

Al presentarse la página de inicio del sistema, se puede acceder de dos maneras a la liga de registro de candidato, una de ellas es mediante el menú principal o bien por medio de la pantalla de acceso al sistema que se presenta en la figura 5.9.

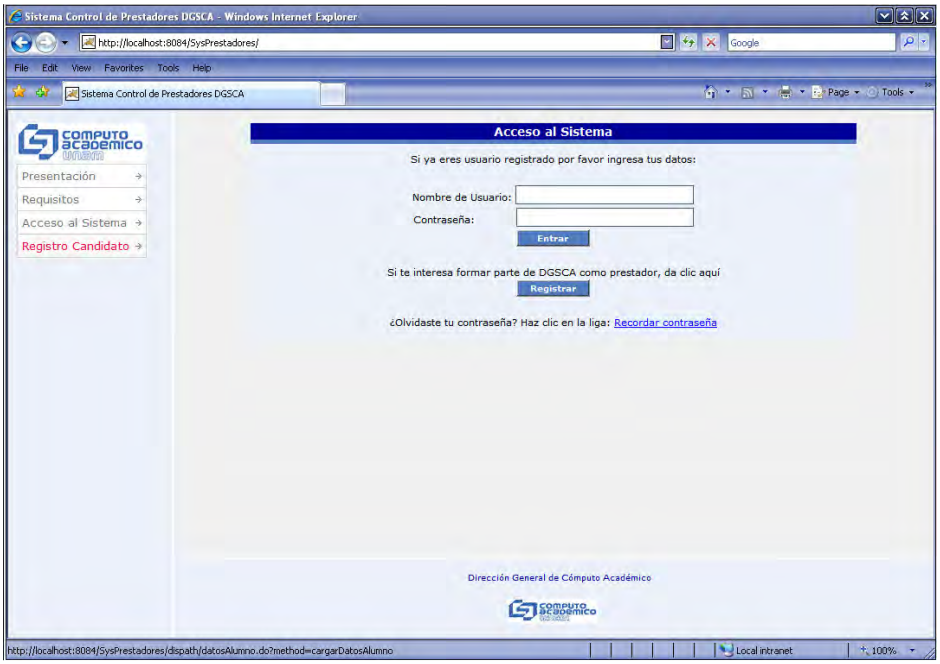


Figura 5.9 Acceso al Sistema

Una vez solicitado el registro del candidato, se presenta la pantalla inicial para la captura de los datos personales y escolares del candidato. Parte de dicha pantalla se puede apreciar en las figuras 5.10

Conforme el candidato va ingresando su información, las pantallas se aprecian como en las figuras 5.11 y 5.12.

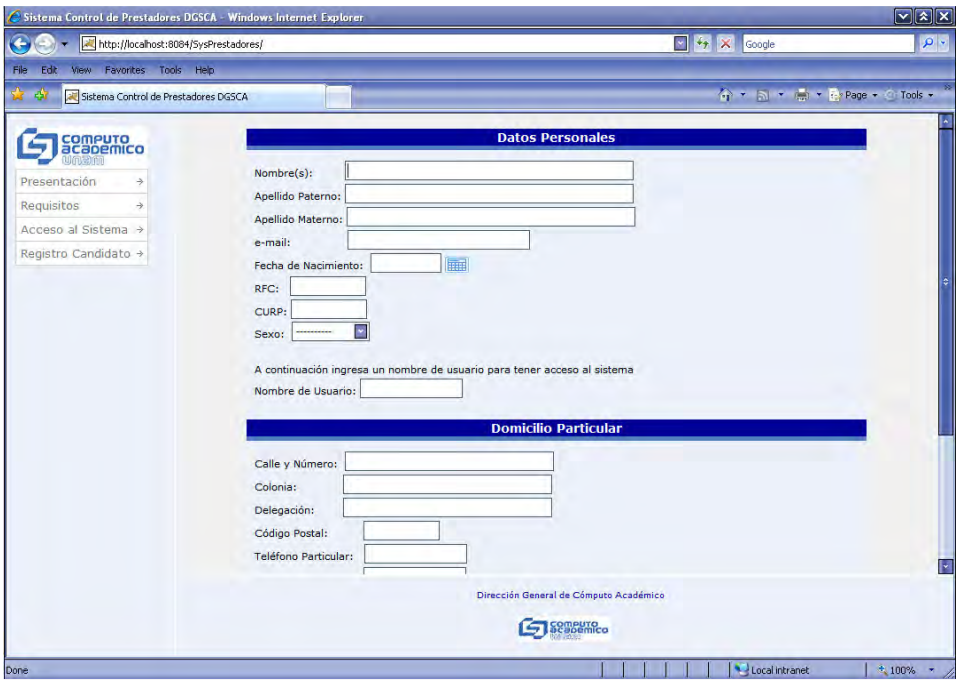


Figura 5.10 Pantalla inicial para ingresar datos personales

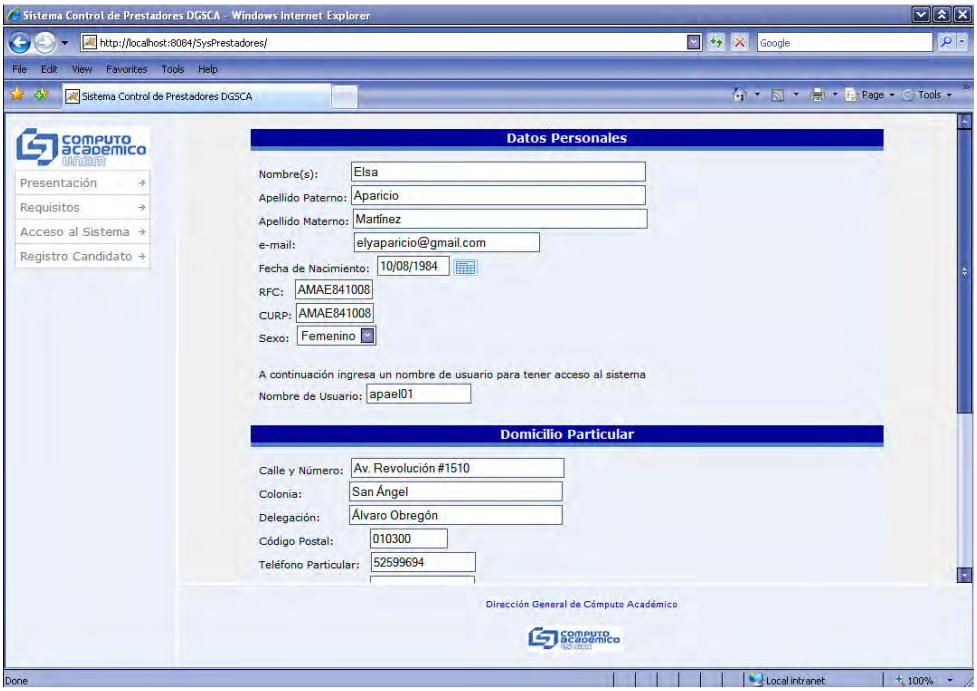


Figura 5.11 Captura de datos personales

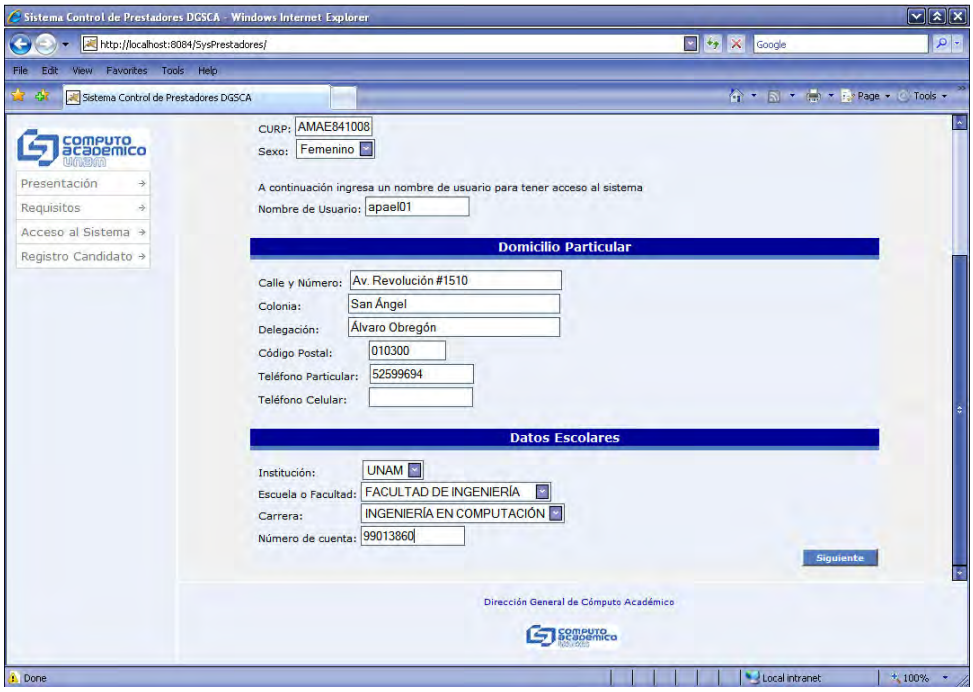


Figura 5.12 Captura de datos personales segunda parte

Al dar clic en el botón siguiente, se muestra la pantalla de las figuras 5.13 y 5.14, para la captura de datos como el semestre que cursa, promedio y para realizar la solicitud del servicio al que se desea ingresar.

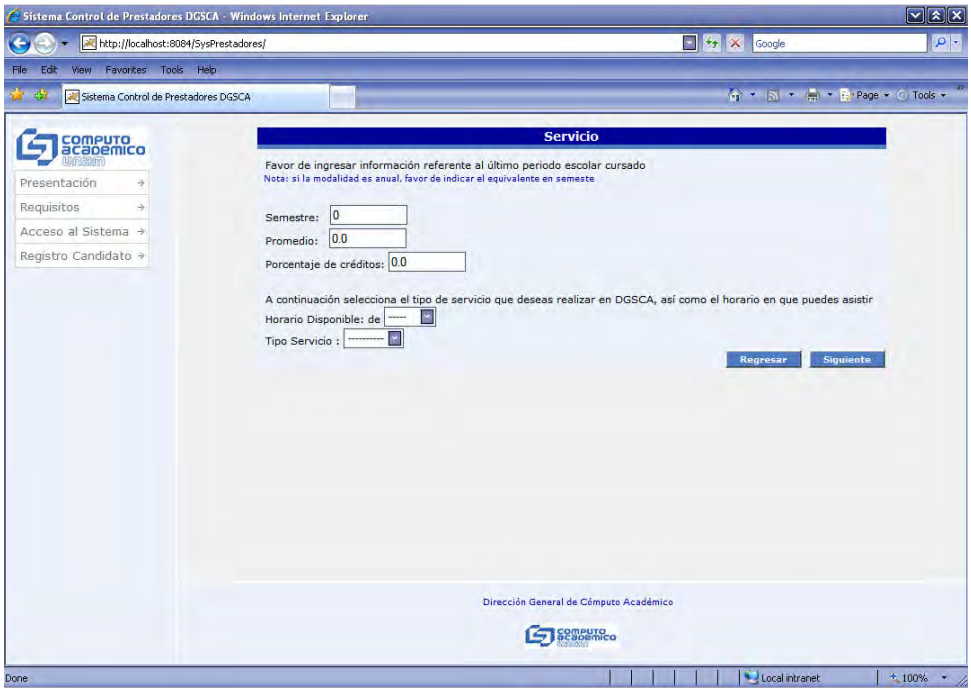


Figura 5.13 14Pantalla de captura de solicitud de servicio

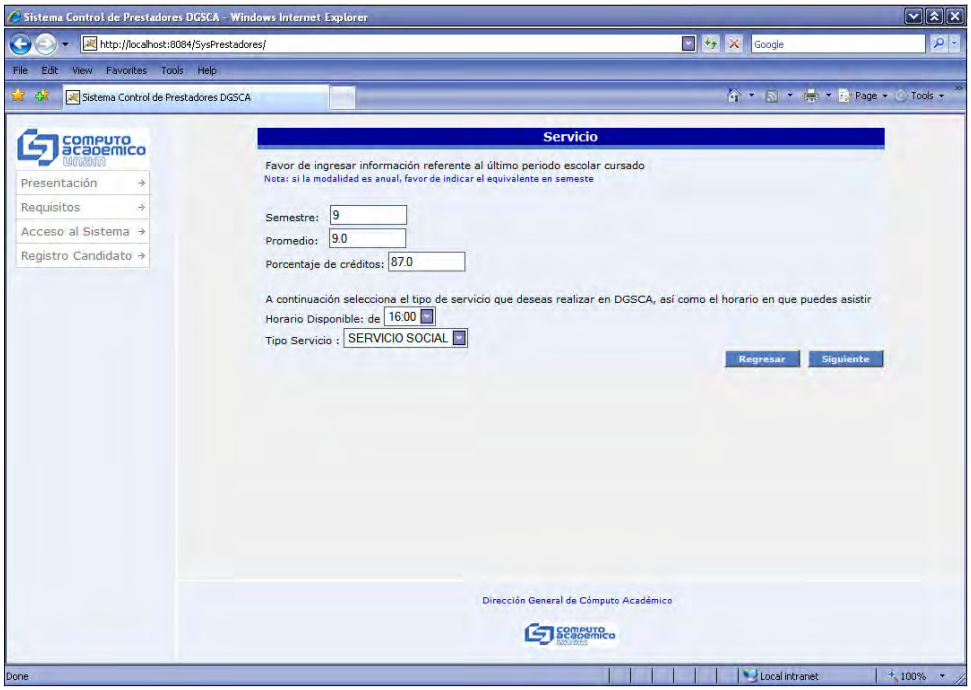


Figura 5.15 Captura de solicitud de servicio

La siguiente pantalla que se presenta es la que permite iniciar la captura del currículum, en primera instancia se permite seleccionar el bloque para registrar áreas temáticas, como en la figura 5.15.

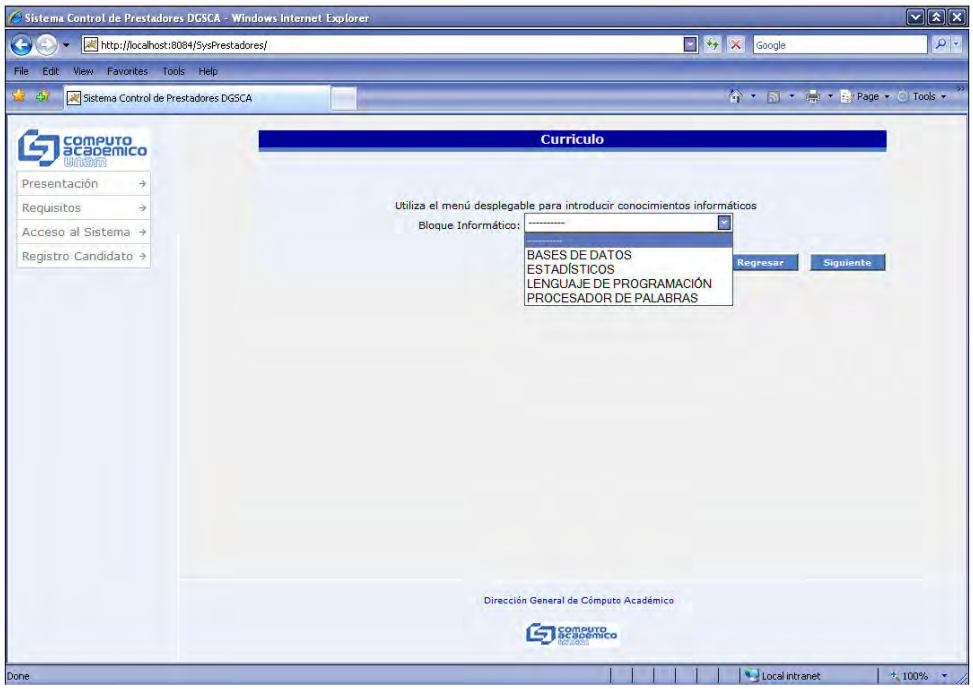


Figura 5.16 Pantalla inicial de captura de currículo

Al realizar la selección, aparecen las áreas temáticas registradas, como en la figura 5.16.

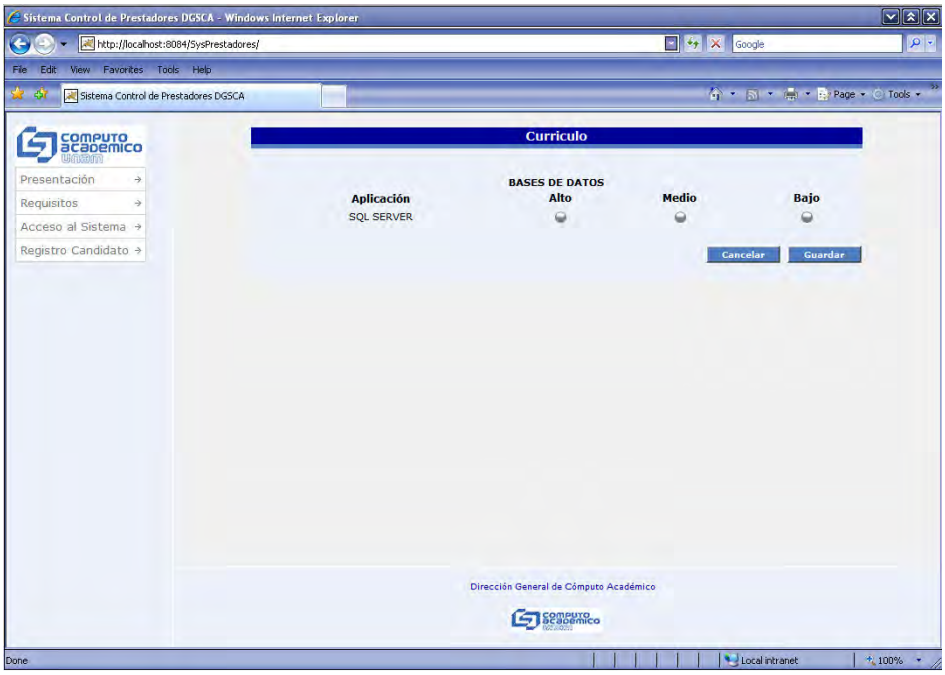


Figura 5.17 Selección de herramientas informáticas

Si se desea ingresar otro bloque, se da clic en Agregar otro y se repite el proceso de seleccionar el bloque deseado a partir de la lista y la vista que tiene el candidato es como la dela figura 5.17.

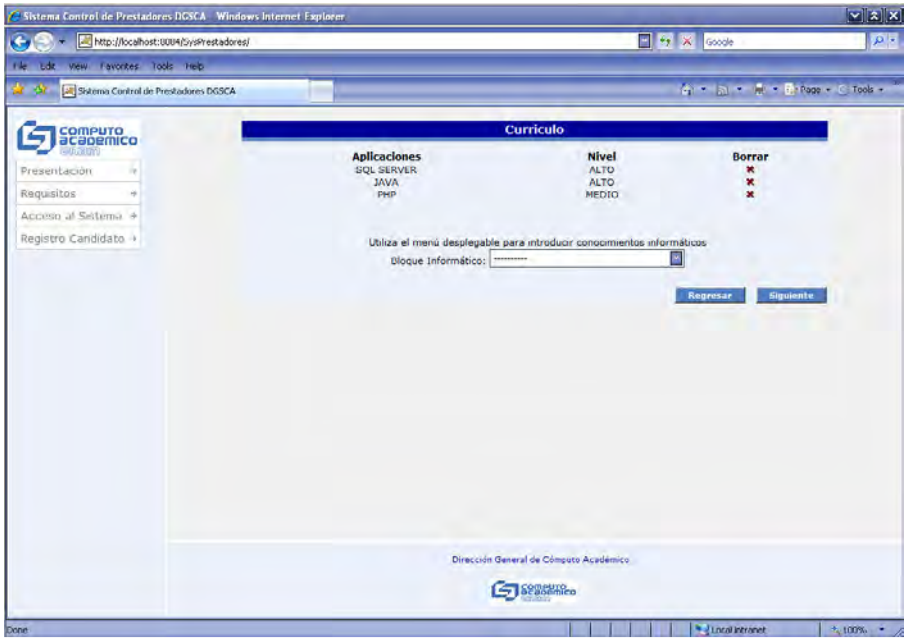


Figura 5.18 Consulta de herramientas seleccionadas

Este proceso se repite hasta que el Candidato ha terminado de registrar sus conocimientos en sistemas.

Posteriormente se presenta la pantalla para registrar Idiomas, que se muestra en las figuras 5.18 y 5.19, con el mismo mecanismo que en la pantalla anterior, se muestra una lista desplegable en la que se selecciona el idioma deseado y mediante los radio botones se selecciona el nivel de manejo del mismo.

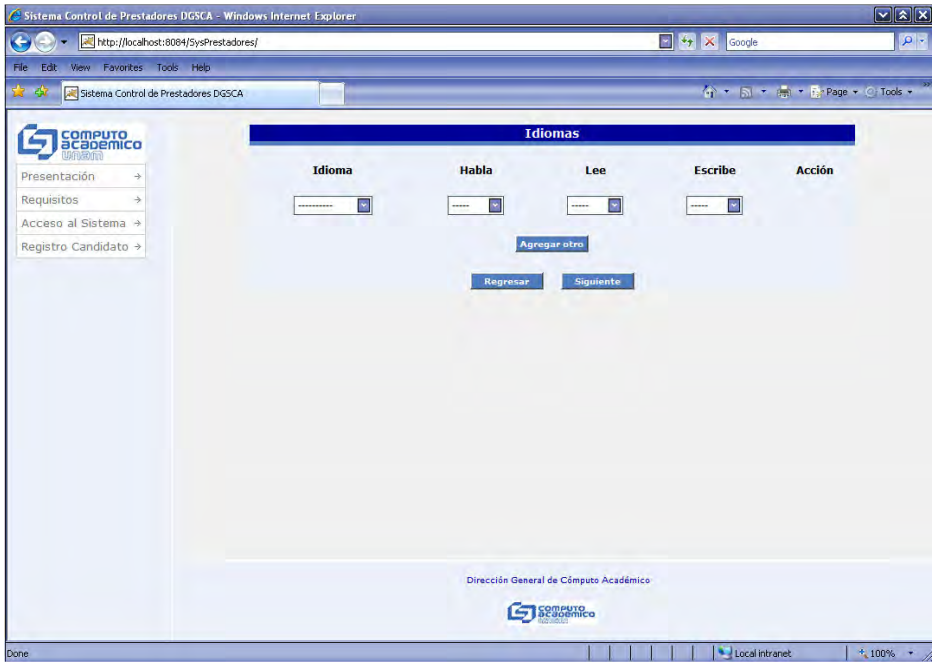


Figura 5.19 Pantalla de captura de idiomas

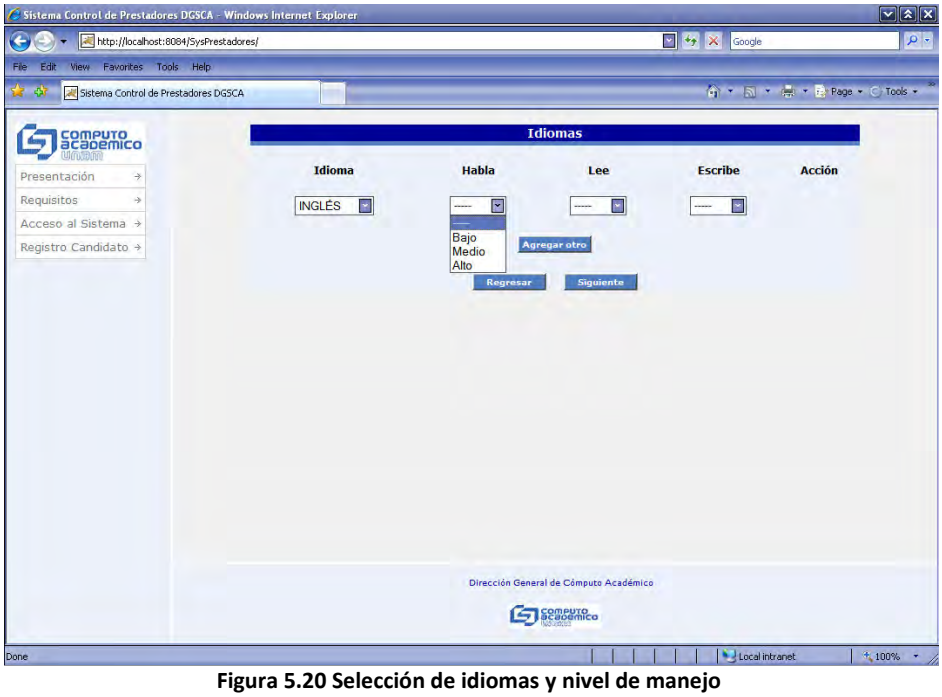


Figura 5.20 Selección de idiomas y nivel de manejo

Si se requiere agregar otro idioma se da clic en el botón agregar otro y así hasta registrar todos los idiomas que se manejen, de manera que al finalizar se tendrá una vista como la de la figura 5.20.

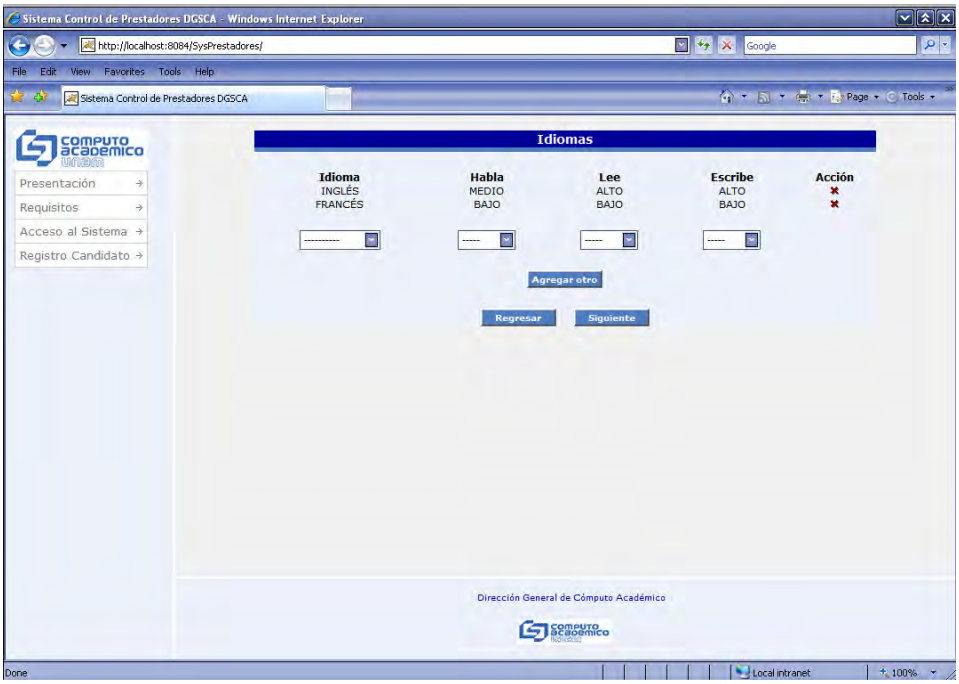


Figura 5.21 Consulta de idiomas seleccionados

Al dar clic en el botón siguiente, se presentan los cuadros de texto que permiten ingresar una explicación de los intereses y experiencia del candidato, tal como aparecen en la figura 5.21.

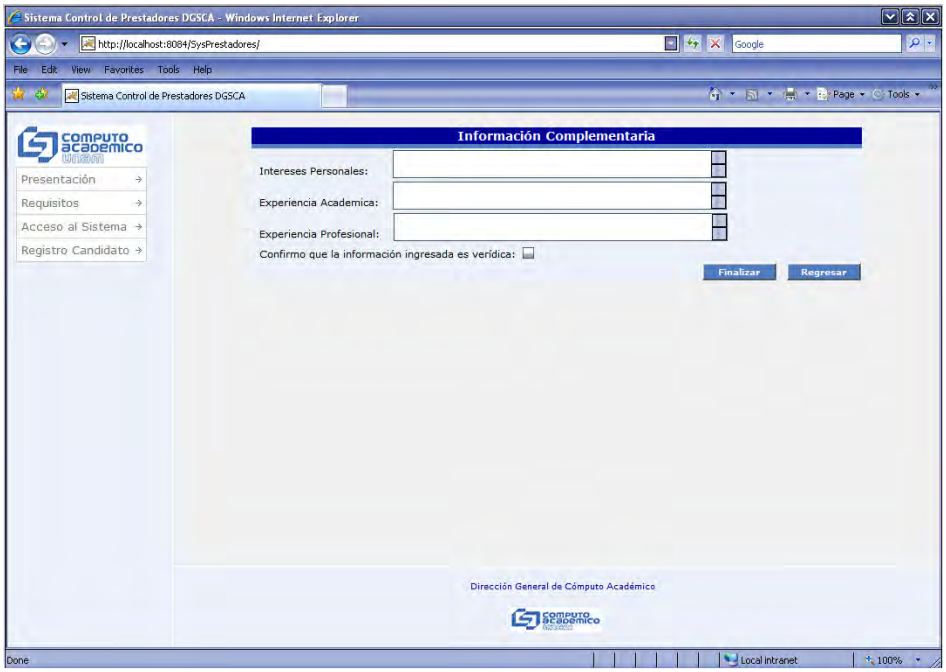


Figura 5.22 Pantalla de captura de información complementaria

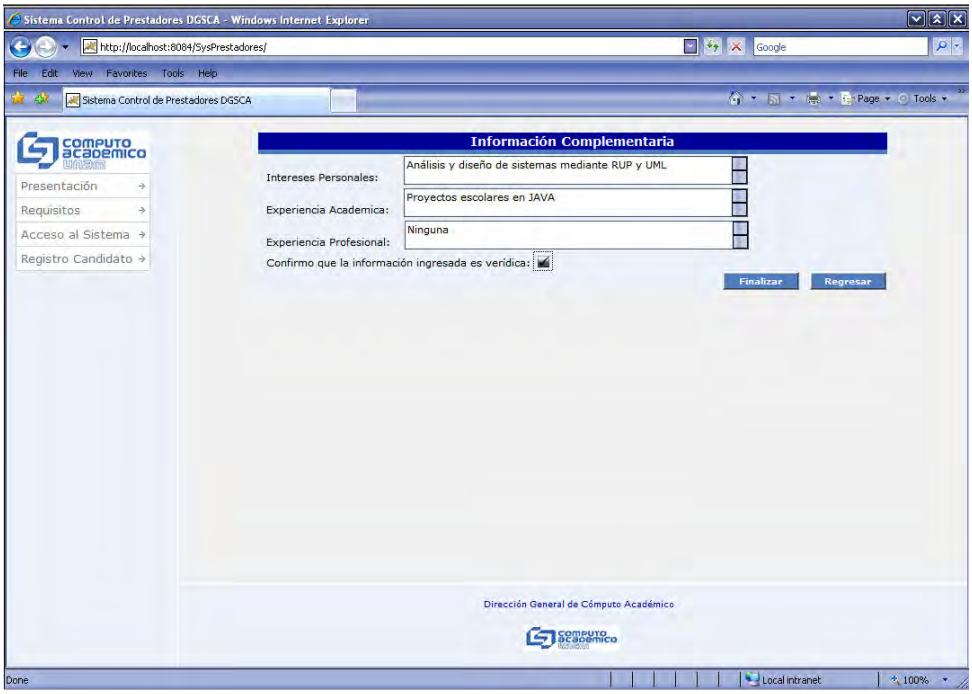


Figura 5.23 Captura de información complementaria

Una vez ingresada la información, se debe indicar por regla de negocio que la información que se ha ingresado es verídica mediante el cuadro de verificación, como se presenta en la figura 5.22.

Al dar clic en finalizar se valida que los datos estén completos y se procede a almacenar el registro en la base de datos del sistema. A continuación se le muestra al candidato una pantalla con la información que ha quedado almacenada y se le indica que por medio de correo electrónico se le hará llegar su contraseña de acceso al sistema para que posteriormente y si así lo requiere pueda realizar modificaciones a la información que ingreso.

En las figuras 5.23 y 5.24 se aprecia el resumen que muestra el sistema, así mismo en las figuras 5.25 y 5.26 se presenta la evidencia del envío de correo electrónico con la contraseña correspondiente para el acceso al sistema.

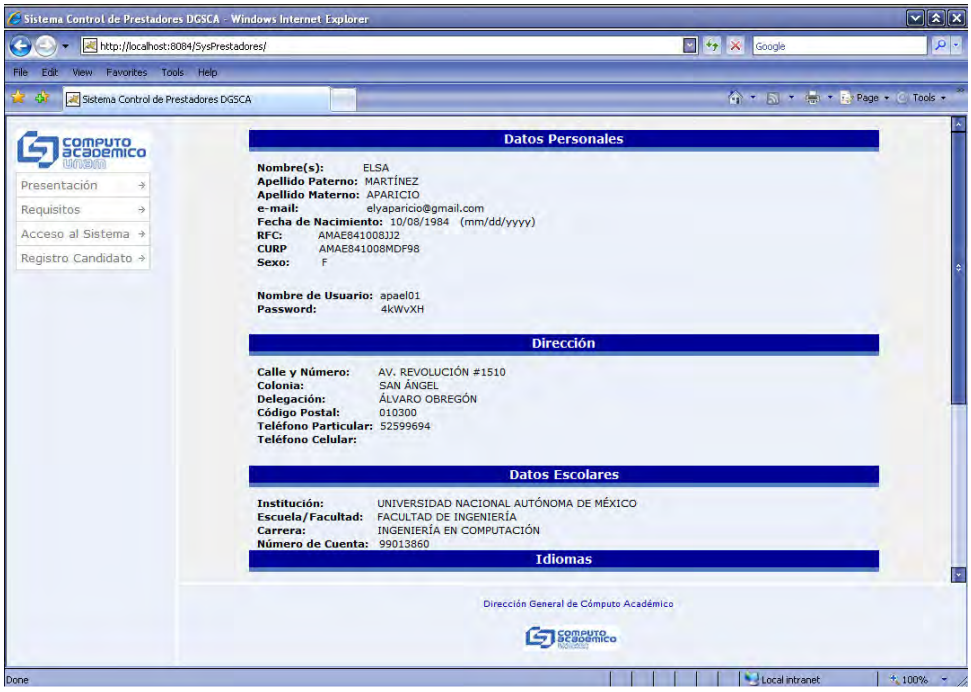


Figura 5.24 Pantalla de resumen de datos ingresados

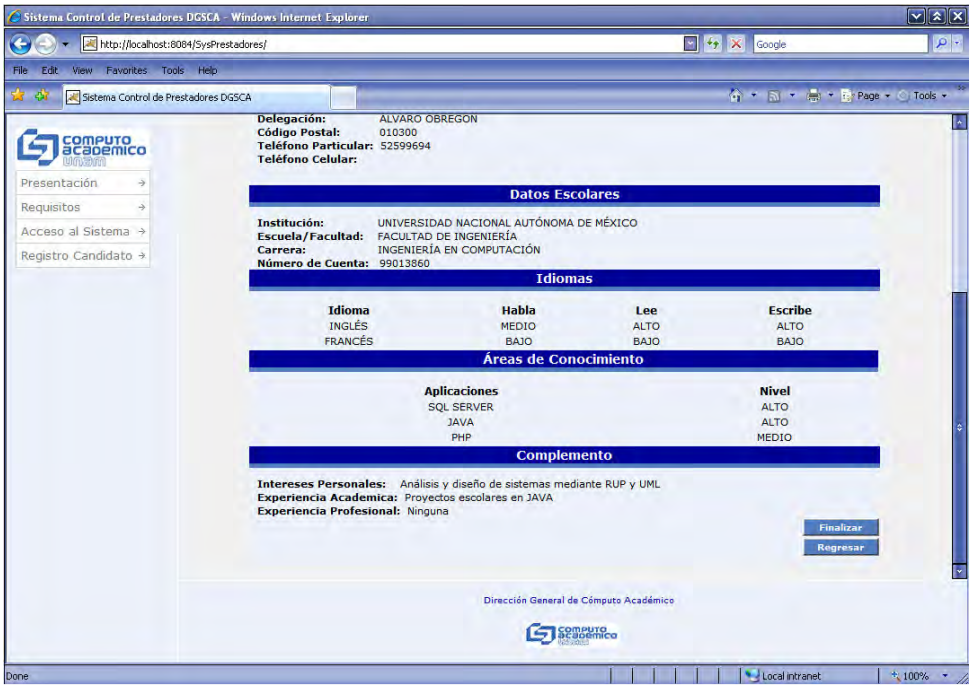


Figura 5.25 Pantalla de resumen de datos ingresados segunda parte

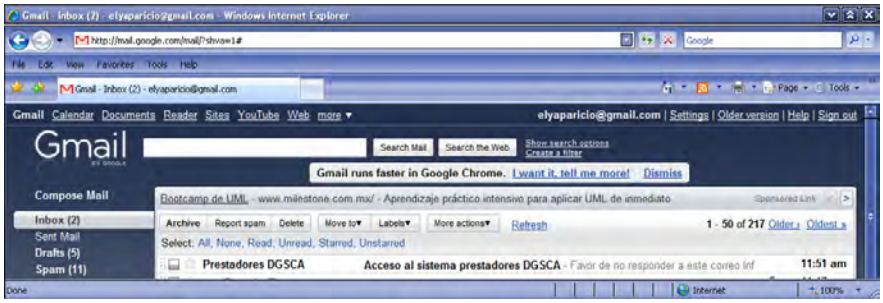


Figura 5.26 E-mail de confirmación de acceso al sistema

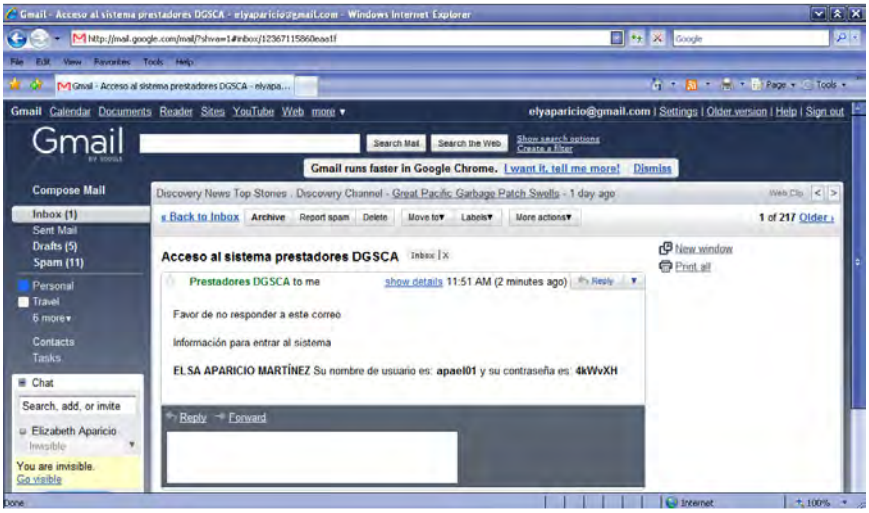


Figura 5.27 Datos de acceso al sistema

Tal como se indica en el caso de prueba, en cada una de las pantallas se realizan validaciones a los datos que se están ingresando, si existe algún error, se envía un mensaje como se aprecia en la figura 5.27.

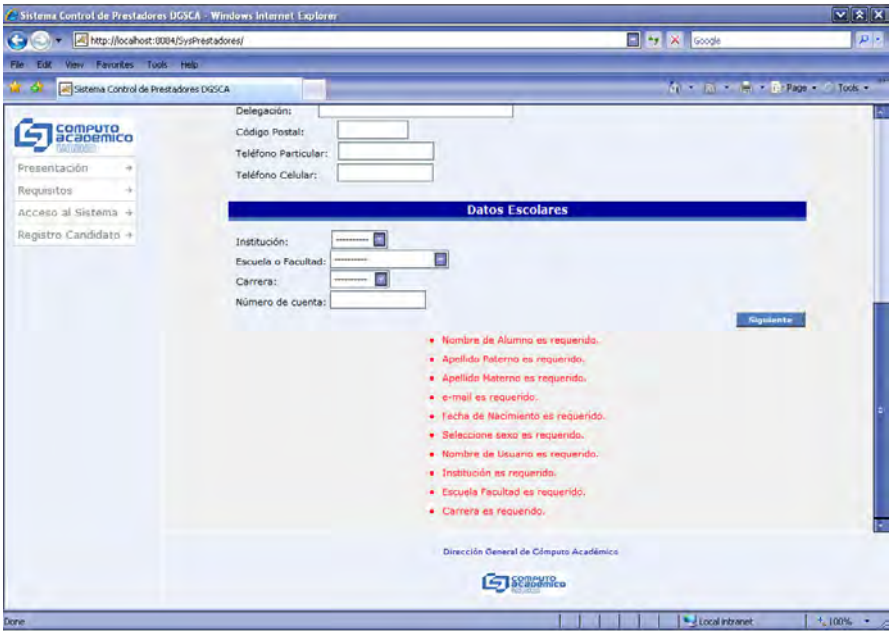


Figura 5.28 Pantalla de errores en datos

5.3. Despliegue

Al tener listos los artefactos de construcción, se prepara el sistema para la liberación, por lo tanto el propósito de la disciplina de Despliegue es configurar el sistema que será entregado, así como habilitar el ambiente para instalar la aplicación, esta información debe estar contenida en el manual de instalación.

A continuación se presenta un fragmento de dicho documento, en el cual se indican los pasos para la liberación así como los requisitos para el mantenimiento de la aplicación.

Liberación y Mantenimiento

La instalación del sistema, únicamente consta de tres archivos, un archivo ***.war*** , el script de la base de datos, en el que se incluyen las sentencias de creación de tablas, así como los store procedures necesarios para el funcionamiento de la aplicación y un archivo de datos con información para la carga inicial de catálogos.

El archivo *.war, contiene la aplicación web, incluyendo todas las librerías y archivos de configuración que se requieren para el correcto funcionamiento de la misma. Para poder ejecutarlo solo es necesario desplegar la aplicación en el servidor Tomcat, siguiendo los pasos que a continuación se indican:

1. Iniciar el Servidor Tomcat.
2. Acceder a http://127.0.0.1:8080
3. Entrar al Tomcat Manager, con usuario y contraseña establecidos.
4. Ir a la opción “Seleccione el archivo war a cargar”, que se encuentra en la parte inferior de la pantalla.
5. Por medio del botón examinar, elegir el archivo *.war y hacer clic en “abrir”.
6. Al tener el archivo seleccionado, oprimir el botón “desplegar” y esperar a que cargue la aplicación.
7. Al finalizar el despliegue, se muestra la pantalla Gestor de Aplicaciones Web de Tomcat, en dicha pantalla se ve la aplicación. Desde esta pantalla se puede arrancar, parar, recargar o replegar la aplicación.
8. Una vez que se ejecutó el comando “Arrancar”, se puede acceder al sistema desde el explorador web.

En el caso del script de la base de datos, solo requiere que éste se ejecute desde el manejador de la base de datos, al igual que el script de carga inicial.

Por otro lado, para poder proporcionar el mantenimiento adecuado al código fuente de esta aplicación es necesario contar con los siguientes requisitos:

- Netbeans 5.1 o superior para edición de código.
- SQL Server 2000 con service pack 3 instalado.

- JDK 1.5 o superior
- Tomcat 5.5
- Librerías:
  - ✓ Struts 1.2.9

  Para acceso a la base de datos:
  - ✓ msbase.jar
  - ✓ mssqlserver.jar
  - ✓ msutil.jar

  Para el envío de correo electrónico desde la aplicación:
  - ✓ mail.jar
  - ✓ activation.jar

6. Transición

De acuerdo a RUP, es la fase final, durante ésta se libera el producto y se entrega al usuario para un uso real. Se incluyen tareas de instalación, configuración, entrenamiento, soporte, mantenimiento. Se completan manuales de usuario. De manera general las actividades a desarrollarse durante la fase son las siguientes:

- Realizar la aceptación técnica del producto.
- Distribuir información.
- Entregar el producto integrado.
- Generar las lecciones aprendidas.
- Cerrar proyecto/fase.

En este apartado se presenta la carta de cierre del proyecto, mientras que las lecciones aprendidas se detallan en el siguiente capítulo, como parte de las conclusiones.

CARTA DE CIERRE DEL PROYECTO

Dirección General de Servicios de Cómputo Académico		Proveedor	
Solicitante	Nombre	**Responsable Proyecto**	Nombre (s)
	Hortensia Cano Granados		Elizabeth Aparicio Arista Rosario Rosas Bernal

Nombre del Proyecto

Sistema de Control para Prestadores de Servicio y Becarios de la DGSCA.

Objetivo del servicio

Desarrollar un sistema bajo la metodología RUP, que permita automatizar la administración de solicitudes de ingreso a DGSCA, de alumnos que requieran realizar servicio social, prestar servicio de apoyo o bien obtener una beca; así como el seguimiento de las actividades realizadas por los mismos una vez aceptados dentro de un programa de servicio.

Fecha de inicio:

01, Agosto, 2008.

Fecha de término:

03, Febrero, 2009.

Alcance

El alcance acordado para el Sistema de Control de Prestadores es el siguiente:

Fase	Actividad
Inicio	1. Realizar levantamiento de requerimientos. 2. Modelado de Negocio
Elaboración	3. Realizar análisis y diseño de requerimientos mediante UML.

Construcción	4. Desarrollar los módulos: • Administración del Sistema • Requisitos Previos • Registro de Servicio • Control y Seguimiento de Servicio • Generación de Documentación
Construcción	5. Generar base de datos.
Construcción	6. Realizar pruebas unitarias y de integración.
Transición	7. Instalar aplicación.

Inventario de actividades:

Descripción de actividades realizadas se detalla en el plan de trabajo, incluido en el **Anexo B**.

Inventario de productos entregados

Documentación (La documentación generada se incluye como anexos al presente trabajo)

Fase	Id.	Nombre	Referencia
Inicio	0010	Estimación de tiempo y costo	Anexo A
Inicio	0020	Plan de trabajo	Anexo B
Inicio	0030	Control de riesgos	Anexo C
Inicio	0040	Diagramas de casos de uso	Anexo D, Capítulo III: Inicio
Inicio	0050	Especificaciones de casos de uso	Anexo E, Capítulo III: Inicio
Inicio	0060	Diagramas de actividad	Anexo F, Capítulo III: Inicio
Elaboración	0070	Diagramas de clases	Anexo G, Capítulo IV: Elaboración
Elaboración	0080	Diagramas de secuencia	Anexo H, Capítulo IV: Elaboración
Elaboración	0090	Diagrama de componentes	Capítulo IV: Elaboración
Elaboración	0100	Diagrama E-R	Anexo I
Elaboración	0110	Diagrama de Modelo Lógico de la base de datos	Capítulo IV: Elaboración
Elaboración	0120	Diagrama de Modelo Físico de la base de datos	Anexo J
Elaboración	0130	Diccionario de datos	Anexo K
Construcción	0140	Casos de Prueba	Capítulo V: Construcción
Construcción	0150	Manuales de Instalación y Usuario	Capítulo V: Construcción

Componentes entregados

Nombre	Lenguaje	Versión	Calificativo de Creación	Descripción
SysPrestadores.war	JAVA, Struts 1.3	V 1.0	Nuevo	Archivo war que contiene el ejecutable de la aplicación.
crearControlPrestadores.sql	SQL	V 1.0	Nuevo	Script de creación de la base de datos ControlPrestadores, contiene los store procedures necesarios para el funcionamiento de la aplicación.
SysPrestadores	JAVA, Struts 1.3	V 1.0	Nuevo	Código Fuente de la aplicación, editable en Netbeans.

Evidencias de pruebas.

- *Equipo en que se desarrollaron las pruebas:* Equipo de pruebas con las siguientes características
  - ✓ Sistema Operativo: Windows XP
  - ✓ Memoria: I GB
  - ✓ Explorador: Internet Explorer
- *Definición de pruebas realizadas y ambientes creados:* Se realizaron pruebas unitarias y de integración de componentes.
- *Evidencias entregadas:* Los casos de prueba ejecutados, se describen en el capítulo V Construcción en la sección de Pruebas.

Cierre del proyecto

El día de hoy 2009/02/05 cubrimos con los requerimientos solicitados para el proyecto Sistema de Control para Prestadores de Servicio y Becarios de la DGSCA cubriendo con los productos mencionados.

Consideraciones especiales:

NA.

Aceptación de los productos

Hemos recibido los productos entregados por el proveedor y estamos de acuerdo con su contenido por lo que confirmamos el cumplimiento de los compromisos señalados en el convenio de servicio.

Por DGSCA	*Por el Proveedor*	*Por el Proveedor*
Hortensia Cano Granados	*Elizabeth Aparicio Arista*	*Rosario Rosas Bernal*

7. Conclusiones

Como parte del cierre del proyecto se identificaron las lecciones aprendidas durante el desarrollo del mismo.

El hecho de contar con una metodología que soporte el desarrollo de un sistema, implica grandes mejorías ya que se tiene una ruta trazada que ha sido probada y en la cual se consideran aquellos aspectos que deberían tenerse en cuenta durante el proceso de desarrollo. Hoy en día, cada empresa adapta según sus necesidades una metodología para el desarrollo de software, generalmente apoyada en el desarrollo en cascada.

Para el desarrollo del sistema de control de prestadores decidimos adaptar la metodología RUP dando como resultado el proceso descrito en este trabajo.

RUP ofrece, a diferencia de otras metodologías como XP (Extreme Programming), un proceso iterativo incremental, permitiendo revisiones continúas con el usuario y conforme se van cerrando los requerimientos, estos se van implementando con la posibilidad de regresar a fases iníciales en caso de ocurrir un cambio de alcance.

Por otro lado, XP es una metodología que se puede aplicar a proyectos de corta duración en los cuales el éxito del proyecto se basa en la rápida codificación, pruebas unitarias y corrección de hallazgos, pudiendo ocasionar en algunos casos retrasos en las entregas debido al mal entendimiento de los requerimientos, en cambio RUP plantea que el entendimiento de las necesidades del usuario es indispensable para lograr un proyecto con éxito, de manera que un punto fuerte de esta metodología es la elaboración de artefactos que permitan esclarecer los requerimientos, diseñar la solución e implementar el sistema.

El haber elegido RUP como metodología para el desarrollo del presente trabajo, significó obtener varias ventajas a lo largo del proyecto, principalmente en la administración del mismo permitiendo el seguimiento de los avances de cada una de las fases y la administración de cambios que incluye actualizar la documentación generada, visualizar correctamente los impactos que podrían derivarse de dichos cambios y de esta forma mantener bajo control el desarrollo del proyecto, evitando retrasos o retrabajo.

Como se puede apreciar en las fases de inicio y elaboración de este proyecto se le dio gran peso a la identificación de requerimientos, así como al análisis y diseño de la solución, con la finalidad de reducir tiempo en la fase de construcción, ya que gran parte del éxito o fracaso de un proyecto tiene que ver con qué tan bien identificados y delimitados están los requerimientos a partir del modelado de negocio.

Si durante la definición de requerimientos no se logra un consenso con el usuario, el alcance del proyecto no quedaría definido, implicando que durante la implementación se tengan que realizar cambios constantemente, provocando atrasos e inconformidades de todo el equipo de desarrollo.

Durante el proyecto comprobamos que el desarrollo de diagramas de casos de uso así como de las especificaciones y diagramas de actividad facilitan el entendimiento entre personas de negocio y de sistemas, aún cuando las personas involucradas de negocio no tengan ningún conocimiento en desarrollo de sistemas.

Asimismo la generación de prototipos durante la fase de elaboración es indispensable, porque le brinda al usuario una aproximación visible de lo que será el sistema solicitado, al presentarlos éste se puede involucrar para identificar si realmente lo que se le esta ofreciendo es lo que necesita, de manera que se puedan hacer los ajustes necesarios a tiempo y no cuando el sistema este terminado.

Adicionalmente, la facilidad al implementar el sistema, radica en un diseño de arquitectura bien planteada a partir de los diagramas de clases, secuencia y componentes, con el detalle de dichos diagramas tuvimos una reducción significativa en el tiempo de codificación.

Por otro parte, todo sistema que implique manejo de información debe tener la capacidad de almacenarla y de manipularla fácilmente, para fines de este proyecto decidimos utilizar una base de datos que a diferencia de un sistema de archivos, maneja consistencia, evita la redundancia y reduce los tiempos de acceso a la información.

En relación a la fase de construcción, sabemos que hoy en día existen nuevas tecnologías, orientadas a disminuir el esfuerzo de implementar lógica común en los sistemas partiendo de la reutilización de soluciones que han sido probadas en varias ocasiones; es por ello que decidimos aprovechar el patrón MVC, ya que este patrón permite la escalabilidad en la funcionalidad del sistema, facilita el mantenimiento de los módulos implementados así como el añadir nueva funcionalidad, dando como resultado la reducción del riesgo de afectar otra.

Para el desarrollo de este proyecto se evaluaron dos posibilidades para la implementación, por un lado se pretendía utilizar Servlets y JSP´s (modelo 2) y por el otro se planteo manejar el framework Struts, ambos basados en MVC.

El resultado utilizando una u otra tecnología, es el mismo ya que finalmente la solución esta enfocada a cumplir con el objetivo planteado, sin embargo al trabajar con un framework estructurado se reduce el esfuerzo ya que es más claro el manejo de la configuración de peticiones.

Al trabajar con Struts encontramos que:

- Se reduce considerablemente el esfuerzo al implementar el patrón MVC.
- Se facilita la separación del controlador, del modelo y de la interfaz de usuario. En el caso del controlador, el flujo de la aplicación se configura en un solo archivo xml.
- Esta misma característica, permite que incluso el desarrollo de la vista y de la lógica del negocio, se puedan hacer en paralelo.
- El realizar validaciones de campos que requieren formatos especiales e identificar si se han ingresado los datos obligatorios desde la vista, mejora el rendimiento de la

aplicación ya que no es necesario realizar llamados al controlador y luego a la base de datos para tal fin.

Por otro lado, al utilizar una tecnología como JAVA se pretendía, reducir los costos en la compra de licencias como en el caso de .NET.

Junto con el correcto entendimiento de los requerimientos, el buen funcionamiento de la aplicación depende de que tan bien se realicen las pruebas y de que tan completas sean, de manera que optamos por elaborar la matriz de casos de prueba en base a los casos de uso sin dejar fuera ningún escenario, comprobamos a través de los casos de prueba, que los resultados que arrojó el sistema fue lo que solicitó el usuario.

Debido a que se trata de un sistema nuevo, pensamos que es indispensable contar con los manuales de instalación y de usuario, para darle un mejor mantenimiento así como explotar la funcionalidad del sistema.

Como resultado de la fase de transición generamos una carta de cierre de proyecto, pues es de suma importancia el dejar por escrito que el usuario esta de acuerdo tanto con las actividades realizadas como con los artefactos entregados y que estos cumplen con lo solicitado, en un desarrollo a través de un contrato, dicha carta sirve para evitar que en un futuro se presenten problemas legales por inconformidad.

Finalmente respecto al desarrollo de este proyecto en cuestión de crecimiento profesional se concluye lo siguiente:

1. Independientemente de la poca o mucha experiencia que se pueda tener en el ámbito profesional, al ver el costo que tendría este proyecto si se hubiera desarrollado por parte de la iniciativa privada, nos damos cuenta que el desarrollo de sistemas es un negocio muy rentable para las consultorías que ofrecen servicios de Tecnologías de la Información.

2. El paso por cada una de las asignaturas de la carrera, sin duda nos brinda la posibilidad de ampliar nuestros puntos de vista y fomentar la búsqueda de soluciones no sólo para cumplir el objetivo sino para cumplirlo de manera óptima.

3. Es responsabilidad de cada una de nosotras como ingenieras la actualización constante y la búsqueda de nuevos conocimientos para poder competir activamente en el ámbito laboral.

8. Anexos

8.1. Anexo A: Estimación de Esfuerzo y Costo

Una vez comprendido el dominio del negocio e identificadas las mejoras y después de haber generado una propuesta de solución se procede a realizar la estimación de tiempo y costo para el desarrollo del sistema. Cabe mencionar que la estimación esta basada en la identificación de casos de uso y como buena práctica, la metodología RUP indica que se debe realizar al inicio de cada una de las fases, sin embargo para fines del proyecto se realizó la estimación completa.

Para la obtención de la estimación de esfuerzo se utilizó la técnica PERT, cuyas siglas en inglés significan: *Program Evaluation and Review Technique.* PERT es un modelo para la administración de proyectos que se usa para analizar las tareas involucradas en completar un proyecto dado, especialmente el tiempo para completar cada tarea, e identificar el tiempo mínimo necesario para completar el proyecto total.

Para estimar la duración esperada de cada actividad es deseable tener experiencia previa en la realización de tareas similares. En planificación y programación de proyectos se estima que la duración esperada de una actividad es una variable aleatoria de distribución con parámetros (a, m, b) donde:

t_a= Se define como el tiempo optimista al menor tiempo que puede durar una actividad.
t_b= Éste es el tiempo pesimista, o el mayor tiempo que puede durar una actividad.
t_m= Es el tiempo más probable que podría durar una actividad.
t_e= Corresponde al tiempo esperado para una actividad.

El valor (o tiempo) esperado en esta distribución se expresa en la siguiente fórmula:

$$t_e = \frac{t_a + 4t_m + t_b}{6}$$

El método PERT, aporta la herramienta que permite planear en forma objetiva, sencilla y práctica, pero a la vez eficaz, todas y cada una de las actividades a realizar para conseguir éxito en los objetivos que se pretenden obtener en el proyecto.

En base a los requerimientos definidos y contemplando la participación de dos recursos con jornadas laborales de lunes a viernes de 4 horas, durante todas las etapas, con el siguiente perfil:

Recurso	Perfil	Comentarios	Fases
1. Recurso 1 2. Recurso 2	Analista Sr.	• Con conocimiento en UML • Experiencia en análisis y diseño OO mínimo de 2 años.	• Inicio • Planeación

	Programador JAVA Jr.	• Con conocimiento en JAVA, Servlets, JSP´s, HTML • Manejo del Framework Struts • Bases de datos (manejador SQL Server) • Experiencia en desarrollo Orientado a Objetos de 1 año.	• A&D • Construcción • Pruebas • Liberación

Tabla Recursos

Se realizó la siguiente evaluación, identificando el tiempo optimista, más probable y pesimista en días, para después calcular el tiempo esperado:

		Duración Propuesta	Optimista	Más Probable	Pesimista	Estimado
Fase	**Actividad**	**Días**	**Ta**	**Tm**	**Tb**	**Te**
1. Inicio		**70**	**44**	**70**	**96**	**70**
1.1	Levantamiento e identificación de requerimientos	30	20	30	40	30
1.2	Elaborar documento de Identificación de requerimientos	5	3	5	7	5
1.3	Elaborar Propuesta de Solución	10	5	10	15	10
1.4	Elaborar Diagramas de Casos de Uso	5	3	5	7	5
1.5	Elaborar Especificaciones de Casos de Uso	15	10	15	20	15
1.6	Elaborar Diagramas de Actividad	5	3	5	7	5
2. Elaboración		**36**	**18**	**36**	**54**	**36**
2.1	**Diseño de interfaz de Usuario**	**10**	**5**	**10**	**15**	**10**
2.1.1	Elaborar prototipos	10	5	10	15	10
2.2	**Realizar análisis y diseño con UML**	**18**	**9**	**18**	**27**	**18**
2.2.1	Diagramas de Clases	3	1	3	5	3
2.2.2	Diagramas de Secuencia	10	5	10	15	10
2.2.3	Diagrama de Componentes	5	3	5	7	5
2.3	**Realizar diseño de Base de Datos**	**8**	**4**	**8**	**12**	**8**
2.3.1	Modelo lógico	5	3	5	7	5
2.3.2	Modelo físico	3	1	3	5	3
3. Construcción		**150**	**88**	**150**	**212**	**150**
3.1	**Base de Datos**	**15**	**8**	**15**	**22**	**15**
3.1.1	Implementar Base de Datos en SQL Server	5	3	5	7	5
3.1.2	Generar SP de acuerdo a funcionalidad de sistema	10	5	10	15	10
3.2	**Sistema**	**105**	**64**	**105**	**146**	**105**
3.2.1	Desarrollar módulo de Administración de Catálogos	15	10	15	20	15
3.2.2	Desarrollar módulo de Administración de Responsables	5	3	5	7	5
3.2.3	Desarrollar módulo de Administración de Proyectos	5	3	5	7	5
3.2.4	Desarrollar módulo de Administración de Servicios	5	3	5	7	5
3.2.5	Desarrollar módulo para Registro de	20	15	20	25	20

	Candidatos					
3.2.6	Desarrollar módulo de Administración de Prestadores	40	20	40	60	40
3.2.7	Desarrollar módulo Administración de Perfiles de Usuario	15	10	15	20	15
3.3	**Pruebas Unitarias y de Integración**	**20**	**10**	**20**	**30**	**20**
3.3.1	Realización de pruebas	10	5	10	15	10
3.3.2	Corrección de defectos encontrados	10	5	10	15	10
3.4	**Pruebas UAT**	**10**	**6**	**10**	**14**	**10**
3.4.1	Realización de Pruebas con Usuario	5	3	5	7	5
3.4.2	Corrección de defectos encontrados	5	3	5	7	5
5. Transición		**7**	**4**	**7**	**10**	**7**
5.1	Instalar piezas de sistema	2	1	2	3	2
5.2	Monitorear instalación	5	3	5	7	5
	Total=	**263**	**154**	**263**	**372**	**263**

Tabla Estimación PERT

La evaluación de costos se realizó con el tiempo estimado a partir de la columna t_e de la tabla anterior y con la siguiente relación de tarifas vigentes en el mercado, acorde a los perfiles mencionados en la Tabla Recursos:

Recurso	Costo / hora
Programador Java Jr.	$ 243.00
Analista UML Sr.	$ 288.00

Tabla Tarifas reales promedio por recurso

Inicialmente se calculó el esfuerzo total en horas, considerando las 4 horas por recurso de lunes a viernes y posteriormente se calculó el costo por fase para así obtener el costo total del proyecto, dando como resultado la siguiente tabla:

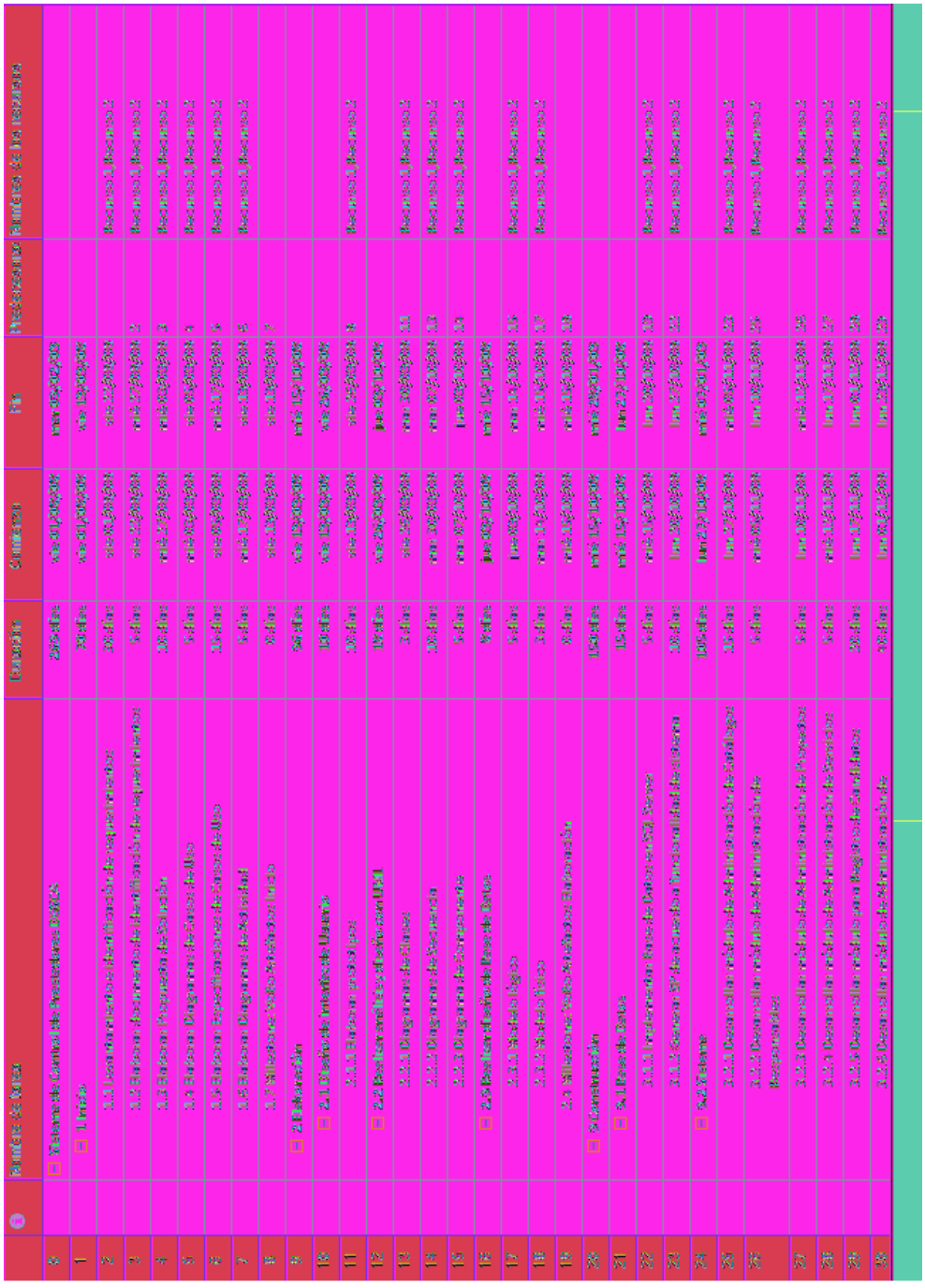
Fase	Actividad	Duración	Esfuerzo	Costo
		Días	Horas	
1. Inicio		**70**	**560**	$ 161,280.00
1.1	Levantamiento e identificación de requerimientos	30		
1.2	Elaborar documento de Identificación de Requerimientos	5		
1.3	Elaborar Propuesta de Solución	10		
1.4	Elaborar Diagramas de Casos de Uso	5		
1.5	Elaborar Especificaciones de Casos de Uso	15		
1.6	Elaborar Diagramas de Actividad	5		
2. Elaboración		**36**	**288**	$82,944.00
2.1	**Diseño de interfaz de Usuario**	**10**		
2.1.1	Elaborar prototipos	10		
2.2	**Realizar análisis y diseño con UML**	**18**		
2.2.1	Diagramas de Clases	3		
2.2.2	Diagrama de Estados	2		
2.2.3	Diagramas de Secuencia	10		

2.2.4	Diagrama de Componente	5		
2.3	**Realizar diseño de Base de Datos**	**8**		
2.3.1	Modelo lógico	5		
2.3.2	Modelo físico	3		
3. Construcción		**150**	**1200**	$291,600.00
3.1	**Base de Datos**	**15**		
3.1.1	Implementar Base de Datos en SQL Server	5		
3.1.2	Generar SP de acuerdo a funcionalidad de sistema	10		
3.2	**Sistema**	**105**		
3.2.1	Desarrollar módulo de Administración de Catálogos	15		
3.2.2	Desarrollar módulo de Administración de Responsables	5		
3.2.3	Desarrollar módulo de Administración de Proyectos	5		
3.2.4	Desarrollar módulo de Administración de Servicios	5		
3.2.5	Desarrollar módulo para Registro de Candidatos	20		
3.2.6	Desarrollar módulo de Administración de Prestadores	40		
3.2.7	Desarrollar módulo Administración de Perfiles de Usuario	15		
3.3	**Pruebas Unitarias y de Integración**	**20**		
3.3.1	Realización de pruebas	10		
3.3.2	Corrección de defectos encontrados	10		
3.4	**Pruebas UAT**	**10**		
3.4.1	Realización de Pruebas con Usuario	5		
3.4.2	Corrección de defectos encontrados	5		
5. Transición		**7**	**56**	$16,128.00
5.1	Instalar piezas de sistema	2		
5.2	Monitorear instalación	5		
	Total=	**263**	**2104**	**$551,952.00**

Tabla Costo

Como se puede apreciar en la tabla el costo del desarrollo del sistema es aproximadamente **$551,952.00 MN,** con una duración estimada de **263 días**.

8.2. Anexo B: Plan de Trabajo



		Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
31		3.2.7 Desarrollar módulo Administración de Perfiles de Usuario	15 días	lun 29/12/08	mié 07/01/09	30	Recurso 1,Recurso 2
32		3.3 Pruebas Unitarias y de Integración	20 días	mié 07/01/09	mié 21/01/09		
33		3.3.1 Realización de pruebas	10 días	mié 07/01/09	mié 14/01/09	31	Recurso 1,Recurso 2
34		3.3.2 Corrección de defectos encontrados	10 días	mié 14/01/09	mié 21/01/09	33	Recurso 1,Recurso 2
35		3.4 Pruebas UAT	10 días	mié 21/01/09	mié 28/01/09		
36		3.4.1 Realización de Pruebas con Usuario	5 días	mié 21/01/09	lun 26/01/09	34	Recurso 1,Recurso 2
37		3.4.2 Corrección de defectos encontrados	5 días	lun 26/01/09	mié 28/01/09	36	Recurso 1,Recurso 2
38		3.5 Milestone: VoBo Artefactos Construcción	0 días	mié 28/01/09	mié 28/01/09	37	
39		4 Transición	7 días	mié 28/01/09	mar 03/02/09		
40		4.1 Instalar piezas de sistema	2 días	mié 28/01/09	jue 29/01/09	38	Recurso 1,Recurso 2
41		4.2 Monitorear instalación	5 días	jue 29/01/09	mar 03/02/09	40	Recurso 1,Recurso 2
42		4.3 Milestone: Cierre del Proyecto	0 días	mar 03/02/09	mar 03/02/09	41	

8.3. Anexo C: Bitácora de Control de Riesgos

El objetivo de la administración de los riesgos es identificar y prepararse para cualquier amenaza que pudiera presentarse durante la realización del proyecto. El hecho de tener controlados los riesgos permite realizar una mejor toma de decisiones, de manera que se concrete con éxito el proyecto.

El proceso de administración de riesgos consiste en los siguientes pasos:

1. Identificar el riesgo.
2. Evaluar el impacto.
3. Establecer prioridades.
4. Desarrollar respuestas para cada riesgo identificado.
5. Lograr aceptación.
6. Seguir un control.

En seguida se presenta una tabla con los riesgos identificados durante el proyecto,

Estado	Descripción	Afectación	P	I	Prioridad	Disparadores	Mitigación	Contingencia
X	Ambigüedades en la definición de requerimientos.	Actividades de recolección y análisis de requerimientos, aprobación de requerimientos.	A	A	A	Tiempo por parte del usuario para atender las ambigüedades.	• Realizar sesiones con el usuario para eliminar ambigüedades. • Escalar la solicitud para obtener una pronta respuesta por parte del responsable.	• Negociar el impacto en tiempo ocasionado por la demora de la respuesta.
X	Cambios de alcance en los requerimientos.	Alcance del proyecto, desviación en calendario.	M	A	A	Inseguridad por parte del Usuario con los requerimientos identificados	• Analizar factibilidad de implementación.	• Negociar el impacto en tiempo ocasionado por los cambios en los requerimientos.
X	Inadecuada estimación de esfuerzo.	Tiempo insuficiente	M	A	A	Tiempo corto.	• Tener revisiones periódicas del calendario de trabajo.	• Negociar el impacto en tiempo.
I	Retraso en calendario a causa de una dependencia de una nueva versión y/o una herramienta.	Desviación en calendario	M	M	M	Retraso en las actividades.	• Asegurarse que sean liberadas las nuevas versiones de las herramientas y versiones a utilizar antes de que se empiece con la codificación.	
X	Falta de experiencia por parte de los recursos de desarrollo en el uso del framework Struts.	Retrasos en la fase de construcción debido a la curva de aprendizaje de los recursos.	M	A	A	Solicitud de implementación en un framework de reciente creación.	• En etapas previas al desarrollo, se capacitará a los recursos.	• Negociar el impacto en tiempo.

C	No contar con el ambiente para la instalación del sistema.	Retrasos en la liberación del producto terminado.	M	M	M	Escasa atención por parte del Usuario para la generación del ambiente, compra de licencias, falta de permisos para realizar configuraciones en los servidores.	• Planear adecuadamente las fechas para la liberación e iniciar trabajos de ambientación en las fechas acordadas.	• Negociar el impacto en tiempo.

Notación:

a) Estado del riesgo:
I = Identificado.
E = Evaluado.
M = En mitigación.
C = En contingencia.
X = Cerrado.

b) Probabilidad de que el riesgo se presente:
A = Alto
M = Medio
B = Bajo

c) Impacto: Es la medición de la afectación del riesgo si se presenta:
A = Alto.
M = Medio.
B = Bajo.

d) Prioridad: Es la combinación entre probabilidad e impacto, con base a tabla anexa.

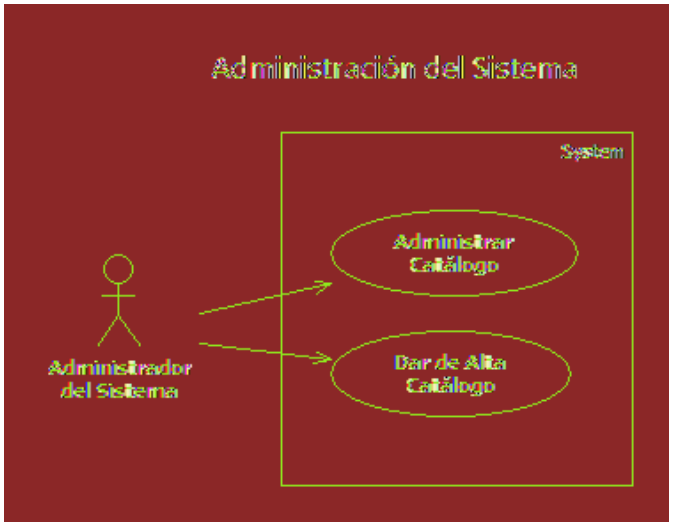
		Impacto		
		Bajo	Medio	Alto
Probabilidad	Alta	M	A	A
	Media	B	M	A
	Baja	B	B	M

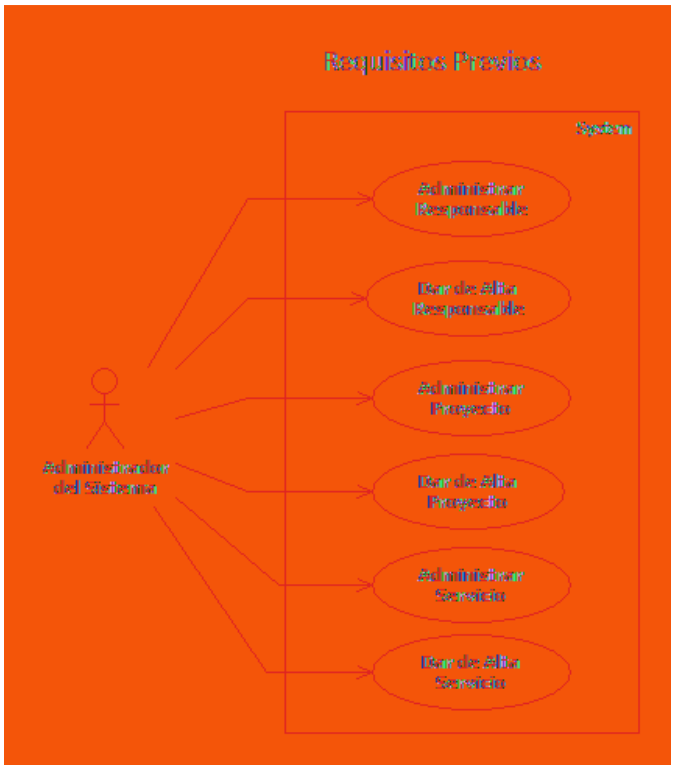
Prioridad		
Prioridad del riesgo	Probabilidad del riesgo	Impacto del riesgo
Alto (A)	Alto (A)	Alto (A)
Alto (A)	Alto (A)	Medio (M)
Alto (A)	Medio (M)	Alto (A)
Bajo (B)	Medio (M)	Bajo (B)
Bajo (B)	Bajo (B)	Medio (M)
Bajo (B)	Bajo (B)	Bajo (B)
Medio (M)	Alto (A)	Bajo (B)
Medio (M)	Medio (M)	Medio (M)
Medio (M)	Bajo (B)	Alto (A)

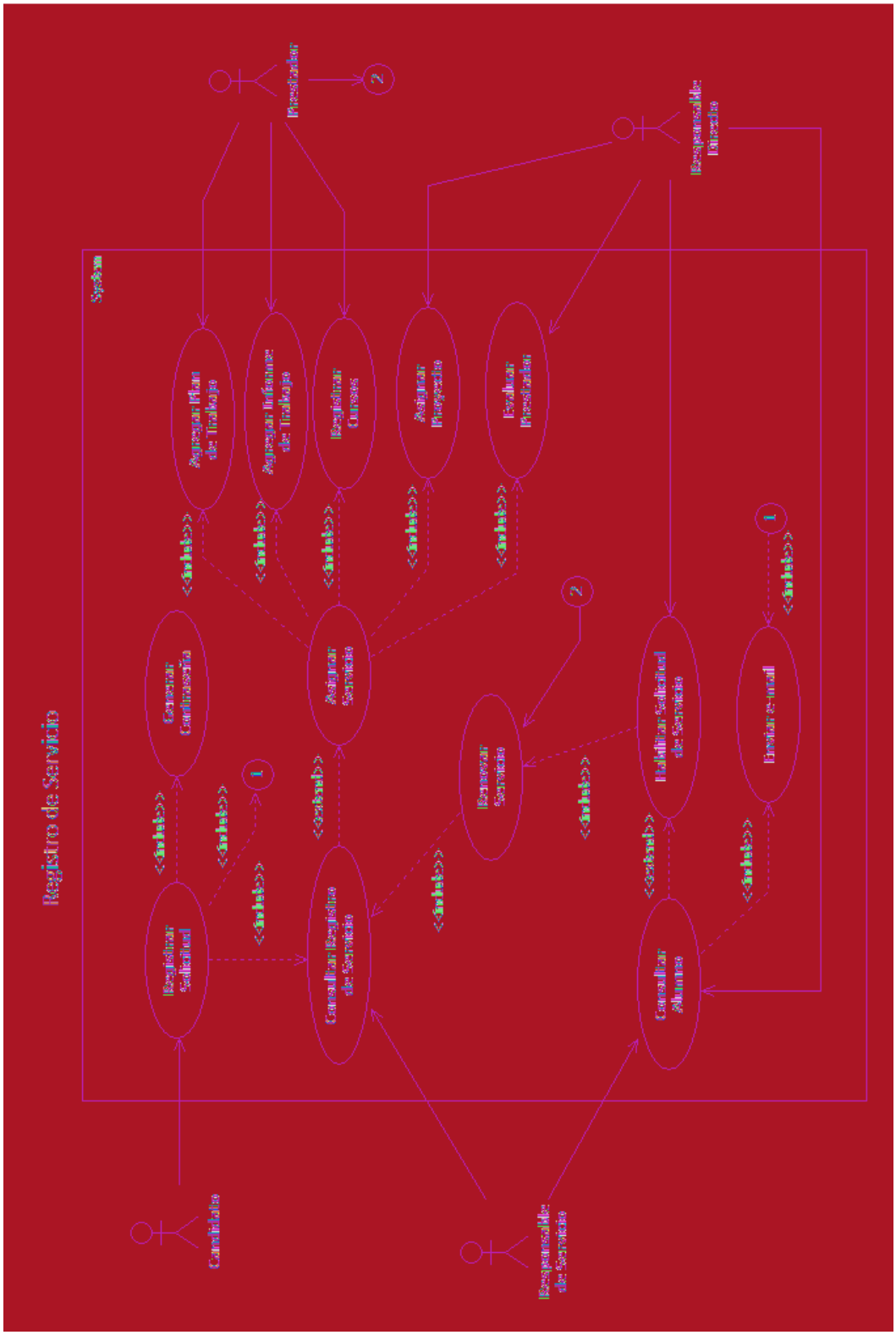
e) Mitigación: Son las actividades necesarias para disminuir la probabilidad de presencia del riesgo y/o disminuir el impacto en caso ocurrir el riesgo.

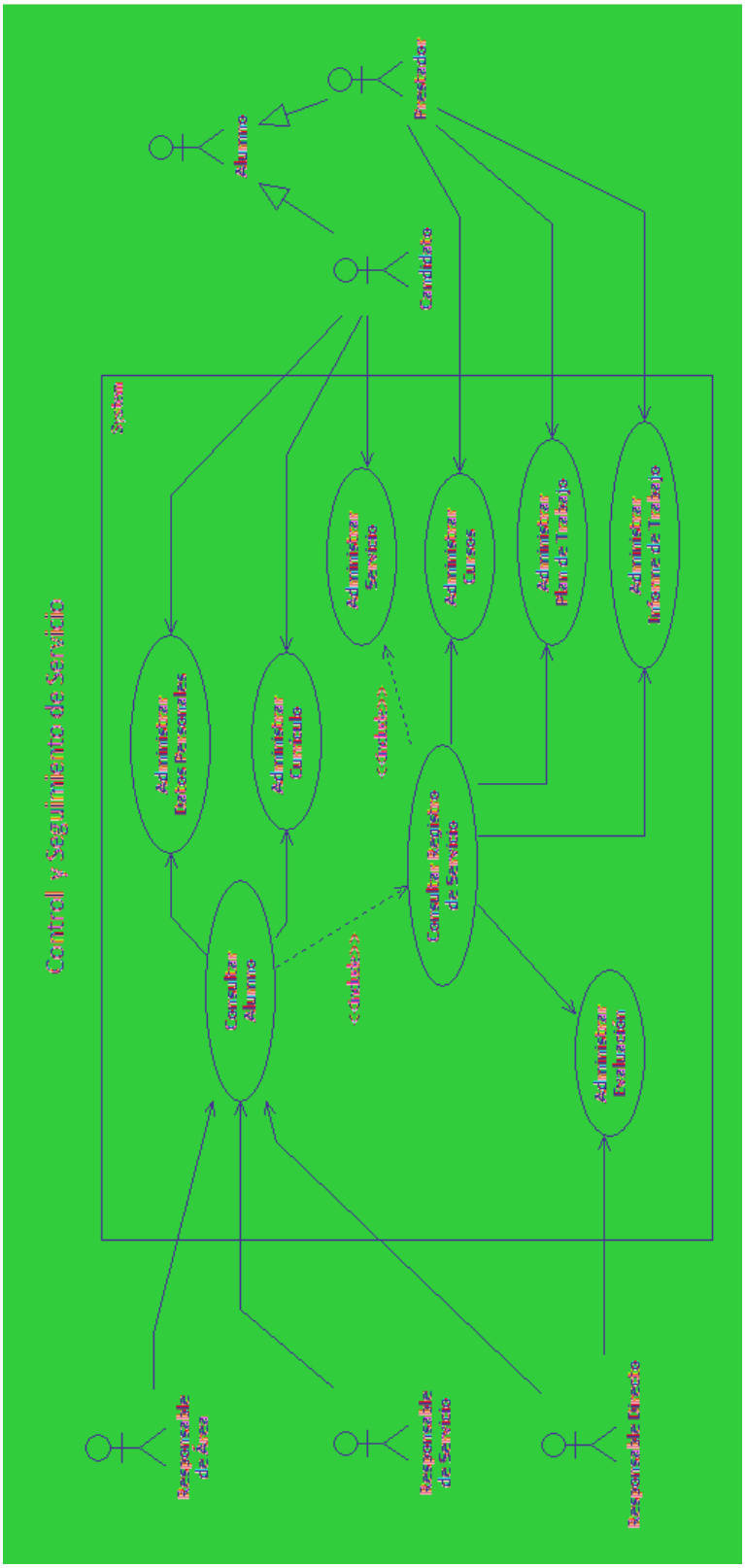
f) Contingencia: Son las actividades que se tienen que realizar una vez que el riesgo se presente.

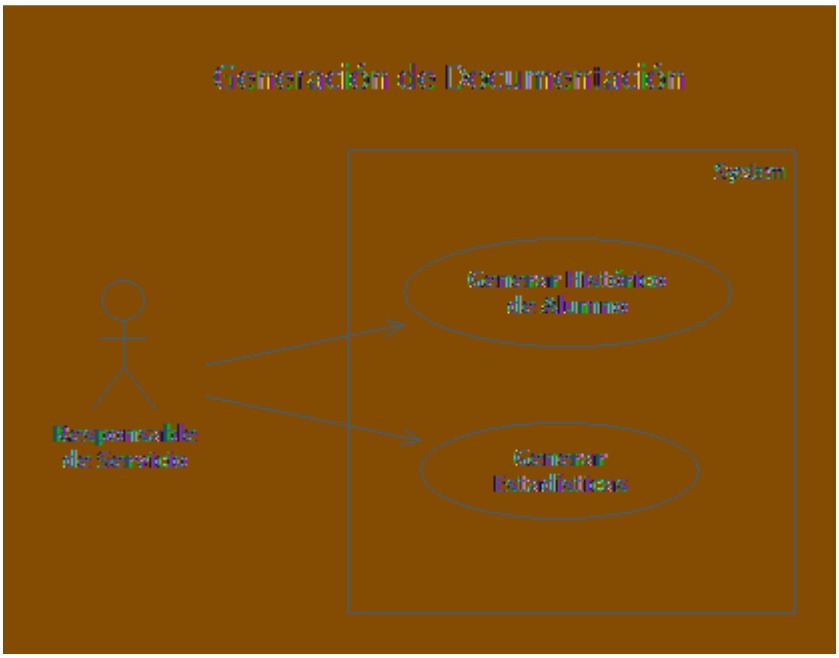
8.4. Anexo D: Diagramas de Casos de Uso






Registro de Servicio
Sistema
Candidato
Prestador
Responsable de Servicio
Responsable Directo
Registrar Solicitud
Entrevistar Coordinación
Cancelar Registro de Servicio
Asignar Servicio
Aceptar Servicio
Entrevistar Jefatura
Solicitar Cancelación de Servicio
Cancelar Alumno
Agregar Plan de Trabajo
Agregar Informe de Trabajo
Registrar Cursos
Asignar Proyecto
Evaluar Prestador
<<include>>
<<extend>>
1
2


Prestador
Alumno
Candidato
Control y Seguimiento de Servicio
Gestion
Administrar Datos Personales
Administrar Expediente
Administrar Servicio
Administrar Cursos
Administrar Plan de Trabajo
Administrar Informe de Trabajo
Consultar Alumno
Consultar Registro de Servicio
Administrar Evaluacion
<<include>>
<<include>>
Responsable de Area
Responsable de Servicio
Responsable Directo


Generación de Documentación
System
Generar Histórico
de Alumno
Responsable
de Servicio
Generar
Estadísticas

8.5. Anexo E : Especificaciones de Casos de Uso

CU-0100 Generar Contraseña

Generales del Caso de Uso			
Nombre Caso Uso	CU-0100 Generar Contraseña		
Creación	Aparicio Arista Reyna Elizabeth Rosas Bernal María del Rosario	**Fecha**	Agosto – 2008

Objetivo		
Crear una contraseña de acceso al sistema; única e irrepetible, automáticamente.		
Nivel del Caso de Uso	**Prioridad**	**Complejidad**
Sistema	Alta	Baja

Actores involucrados

- **Sistema**.- Es el encargado de generar una contraseña de acceso para un Usuario, ya sea un Candidato o un Responsable.

Precondiciones

1. Debe haberse realizado la solicitud de registro en el caso de un Candidato.
2. Debe haberse dado de alta un Responsable.

Post-condiciones

1. Se contará con una contraseña para acceso al sistema que le será enviada a un Candidato o Responsable.

Escenario Principal

Paso	Acción
1.	El sistema solicita la generación de una contraseña.
2.	El Sistema selecciona un carácter aleatoriamente para formar la contraseña. ***Ver regla de negocio RN-001 y RN-002***.
3.	El sistema forma una cadena, concatenando el carácter que seleccionó hasta que tenga 6 caracteres.
4.	El sistema verifica que la contraseña que se generó no la tenga algún otro usuario.
5.	En caso de que la contraseña sea única, el Sistema continúa con el alta del registro del Candidato o del Responsable. En caso contrario regresa al punto 1 del escenario principal.

Escenarios Alternos

EA01 – NA	
Paso	**Acción**
1.	

Excepciones

EX01 – NA.	
Paso	**Acción**
1.	

Reglas de Negocio

Id	Regla de Negocio
RN-001	La contraseña puede estar formada por una combinación de los siguientes caracteres: • Letras mayúsculas: A,B,C,D,E,F,G,H,J,K,L,M,P,Q,R,S,T,U,V,W,X,Y • Letras minúsculas: a,b,c,d,e,f,h,j,k,m,n,p,q,r,s,t,u,v,w,x,y • Números: 2,3,4,5,6,7,8,9
RN-002	La contraseña no puede contener ningún otro carácter diferente a los especificados en la RN-001.

Requerimientos No-Funcionales

Id	Requerimiento No-Funcional
RNF-001	

Notas para implementación
1.

CU-0110 Enviar e-mail

Generales del Caso de Uso			
Nombre Caso Uso	CU-0110 Enviar e-mail		
Creación	Aparicio Arista Reyna Elizabeth Rosas Bernal María del Rosario	**Fecha**	Agosto – 2008

Objetivo

Enviar por correo electrónico notificaciones de generación de contraseña, de recuperación de contraseña o bien enviar una notificación a un Candidato para presentarse a entrevista.

Nivel del Caso de Uso	Prioridad	Complejidad
Sistema	Alta	Baja

Actores involucrados

- **Sistema**.- Es el encargado de generar y enviar un correo electrónico para alguno de los siguientes casos: generación de contraseña, recuperación de contraseña o notificación para presentarse a entrevista.

Precondiciones

1. Debe haber registrado un correo electrónico válido para cada Usuario del sistema.
2. Debe haberse generado una contraseña ya sea para un Candidato o para un Responsable.
3. Debe haberse solicitado la recuperación de una contraseña olvidada.
4. Debe haberse seleccionado un Candidato para que asista a una entrevista con algún Responsable.

Post-condiciones

1. Se enviará un correo electrónico de forma automática para los siguientes casos:
   - Cuando se genere una contraseña nueva para un Candidato o un Responsable.
   - Cuando se haya solicitado la recuperación de una contraseña.
   - Cuando se desee notificar a un Candidato para que asista a una entrevista.

Escenario Principal

Paso	Acción
1.	El Sistema obtiene la información correspondiente al Usuario a quién esta dirigido el e-mail. ***Ver regla de negocio RN-001 y RN-002***.
2.	El Sistema guarda como parte del cuerpo el siguiente mensaje: "**FAVOR DE NO RESPONDER A ESTÉ CORREO**".
3.	El Sistema verifica el tipo de mail que tiene que generar.
4.	El Sistema genera el asunto del correo electrónico. ***Ver regla de negocio RN-003*** en caso de que se haya solicitado enviar un mail de registro en el sistema, en caso contrario ***Ver escenario alterno EA01.***
5.	El Sistema complementa el cuerpo del correo electrónico. ***Ver regla de negocio RN-004.***
6.	El Sistema envía el correo electrónico a la dirección identificada.
7.	Fin de Caso de Uso.

Escenarios Alternos

EA01 – Correo electrónico para notificación de entrevista.	
Paso	**Acción**
1.	El Sistema genera el asunto del correo electrónico. ***Ver regla de negocio RN-005***. En caso de que se haya solicitado recuperar contraseña, ***ver escenario alterno EA02*.**
2.	El Sistema arma el cuerpo del correo electrónico. ***Ver regla de negocio RN-006.***
3.	*Regresar al punto 6 del escenario principal.*

EA02 –Recuperación de contraseña.	
Paso	**Acción**
1.	El Sistema genera el asunto del correo electrónico. ***Ver regla de negocio RN-007***.
2.	El Sistema arma el cuerpo del correo electrónico. ***Ver regla de negocio RN-008.***
3.	*Regresar al punto 6 del escenario principal.*

Excepciones

EX01 – NA.	
Paso	**Acción**
1.	

Reglas de Negocio

Id	**Regla de Negocio**
RN-001	El correo electrónico para notificar la generación de una contraseña será el mismo que para la recuperación de la contraseña.
RN-002	La información que deberá recuperar el Sistema es la correspondiente al Usuario: Nombre completo Usuario Password
RN-003	El asunto del correo electrónico para notificación de contraseña es: "***Registro en el Sistema de Prestadores DGSCA***".
RN-004	El cuerpo del correo electrónico para la notificación de contraseña es: "***Favor de no responder a este e-mail.*** ***[nombre de Candidato/Responsable]*** ***Le informamos que sus datos han sido registrados en el Sistema de Prestadores de DGSCA, si requiere hacer alguna actualización puede entrar al Sistema con la siguiente información:*** ***Usuario : [nombre de usuario]*** ***Contraseña: [contraseña]*** ***Atentamente: Servicio Social DGSCA***"

RN-005	El asunto del correo electrónico cuando se solicita notificarle al alumno que se presente a una entrevista es: "***Notificación de Entrevista DGSCA***".
RN-006	El cuerpo del correo electrónico para la notificación de entrevista es: "***Favor de no responder a este e-mail.*** ***Estimado Alumno:*** ***Se te notifica que has sido seleccionado para presentarte a una entrevista con un Responsable de Servicio, favor de comunicarte a la brevedad a la Coordinación de Servicio Social.*** ***Atentamente: Servicio Social DGSCA***"
RN-007	El asunto del correo electrónico cuando se solicita recuperar la contraseña es: "***Recuperación de contraseña Sistema de Prestadores DGSCA***".
RN-008	El cuerpo del correo electrónico recuperación de contraseña es: "***Favor de no responder a este e-mail.*** ***[nombre de Candidato/Prestador/Responsable]*** ***La información para tener acceso al sistema de prestadores de DGSCA es la siguiente.*** ***Usuario : [nombre de usuario]*** ***Contraseña: [contraseña]*** ***Atentamente: Servicio Social DGSCA***"

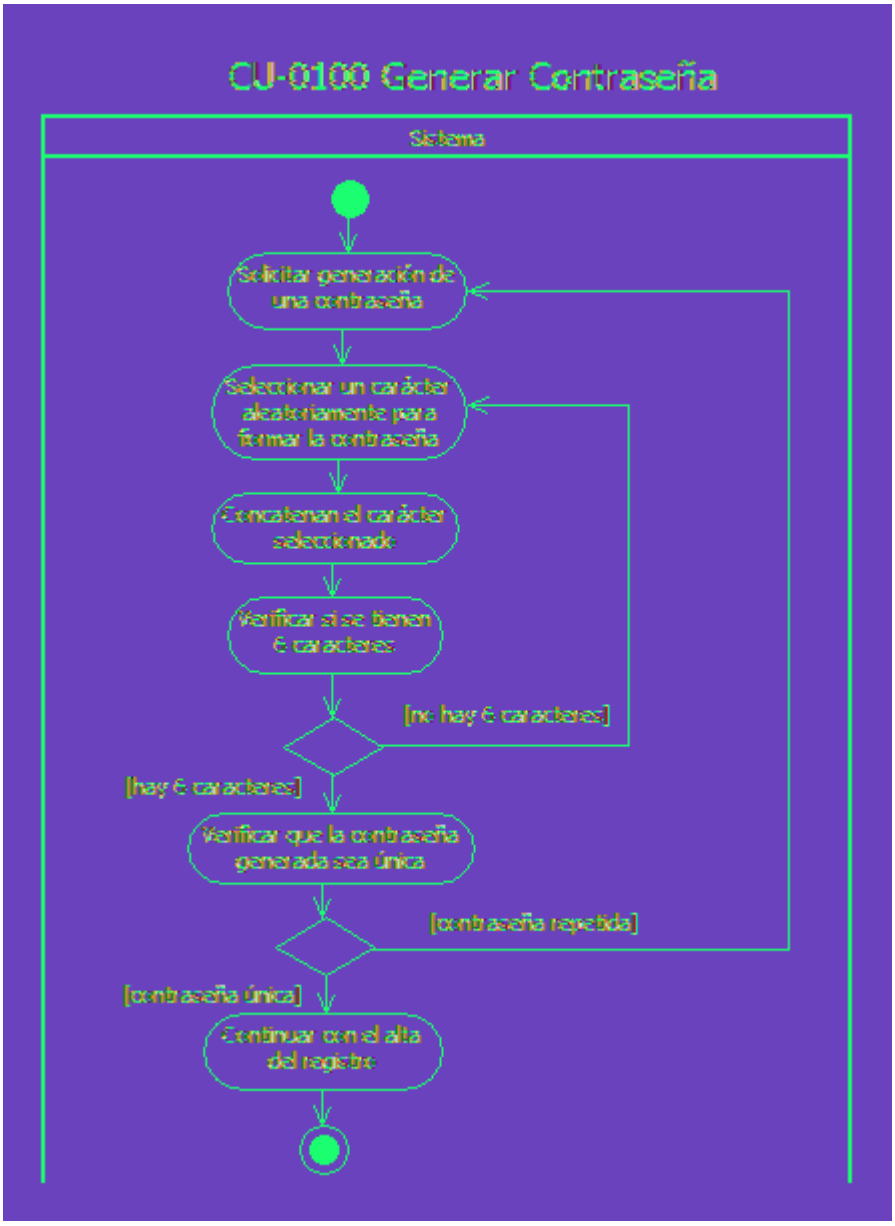
Requerimientos No-Funcionales

Id	Requerimiento No-Funcional
RNF-001	

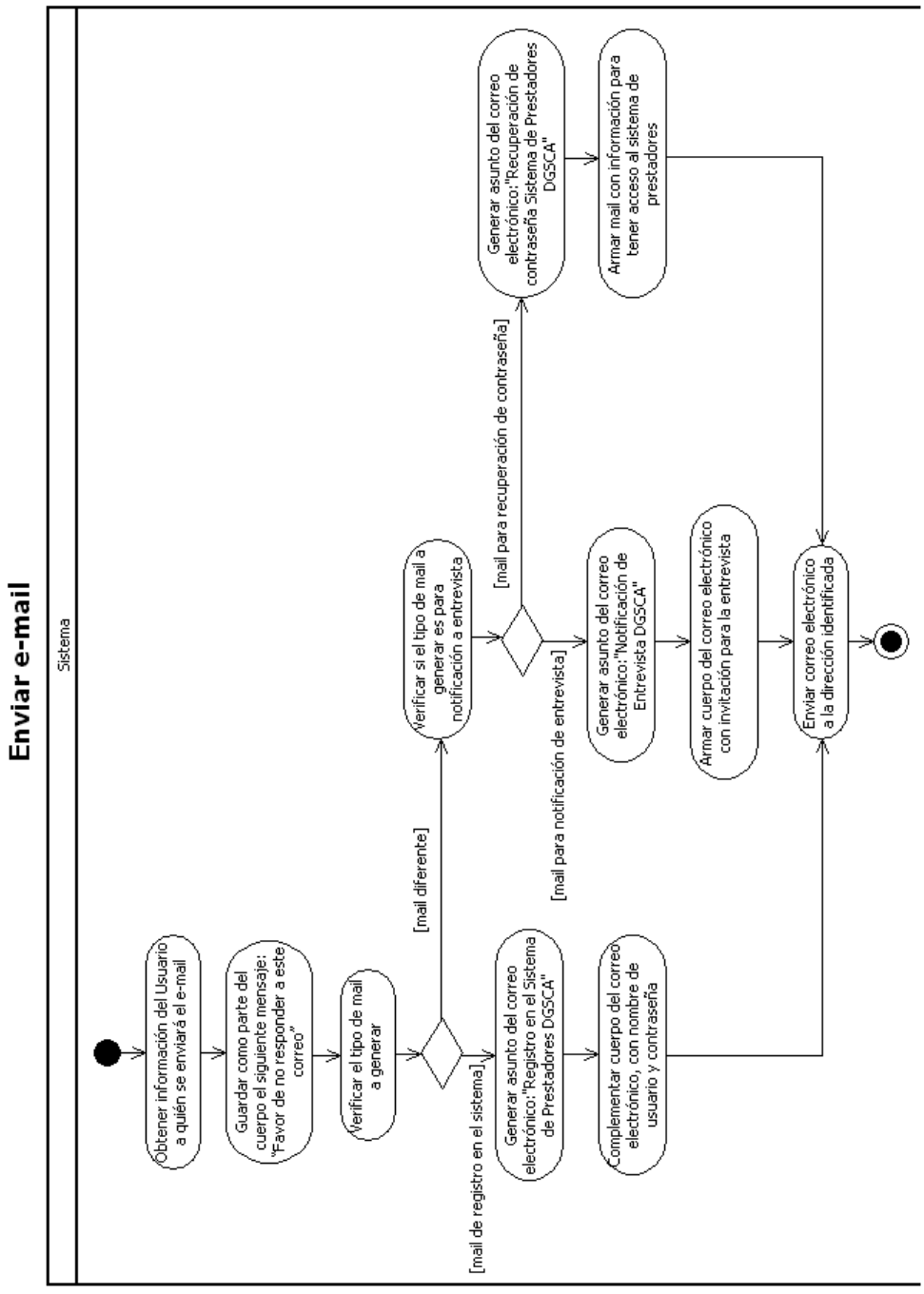
Notas para implementación
1.

8.6. Anexo F: Diagramas de Actividad

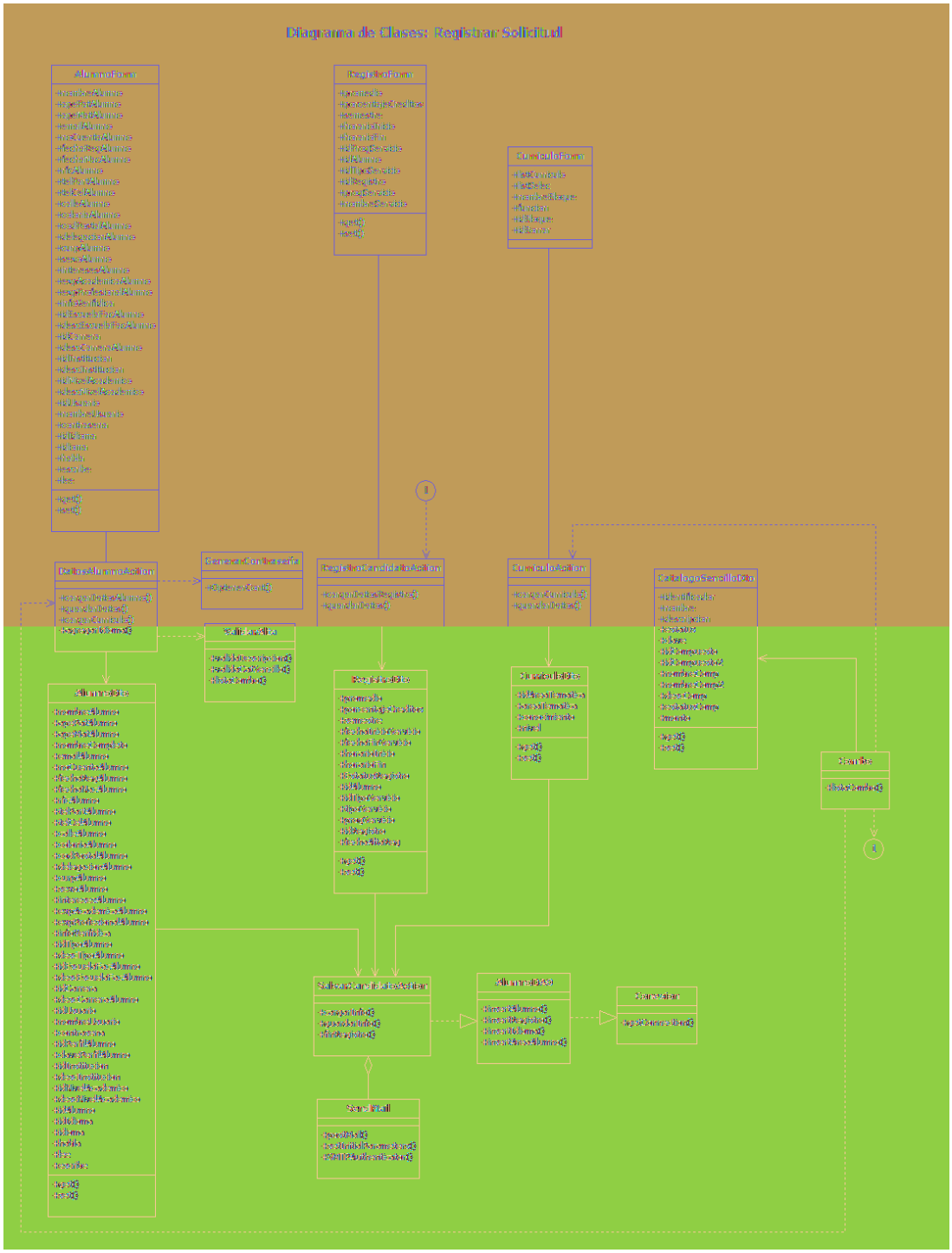
CU-0100 Generar Contraseña



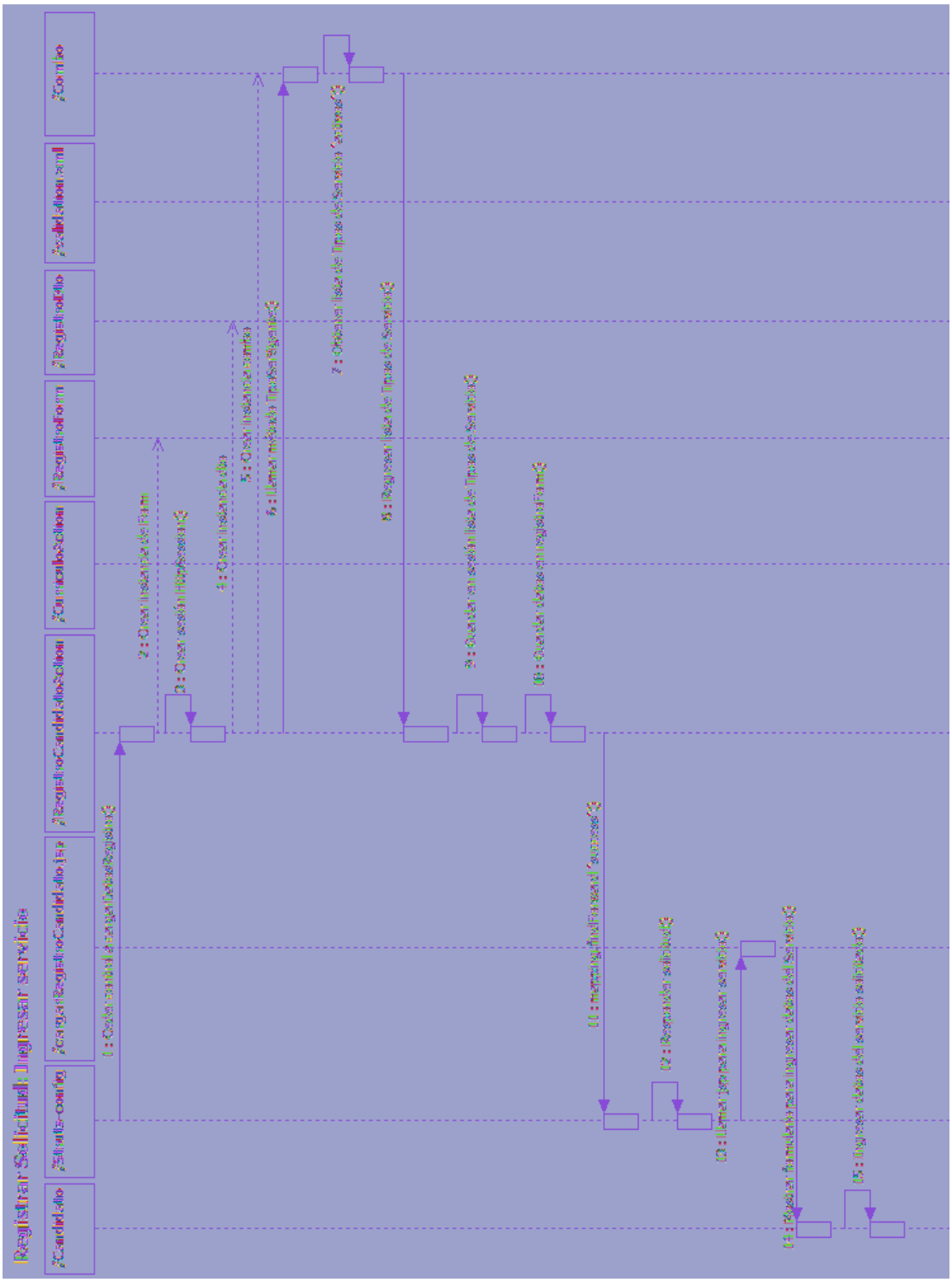
CU-0110 Enviar e-mail

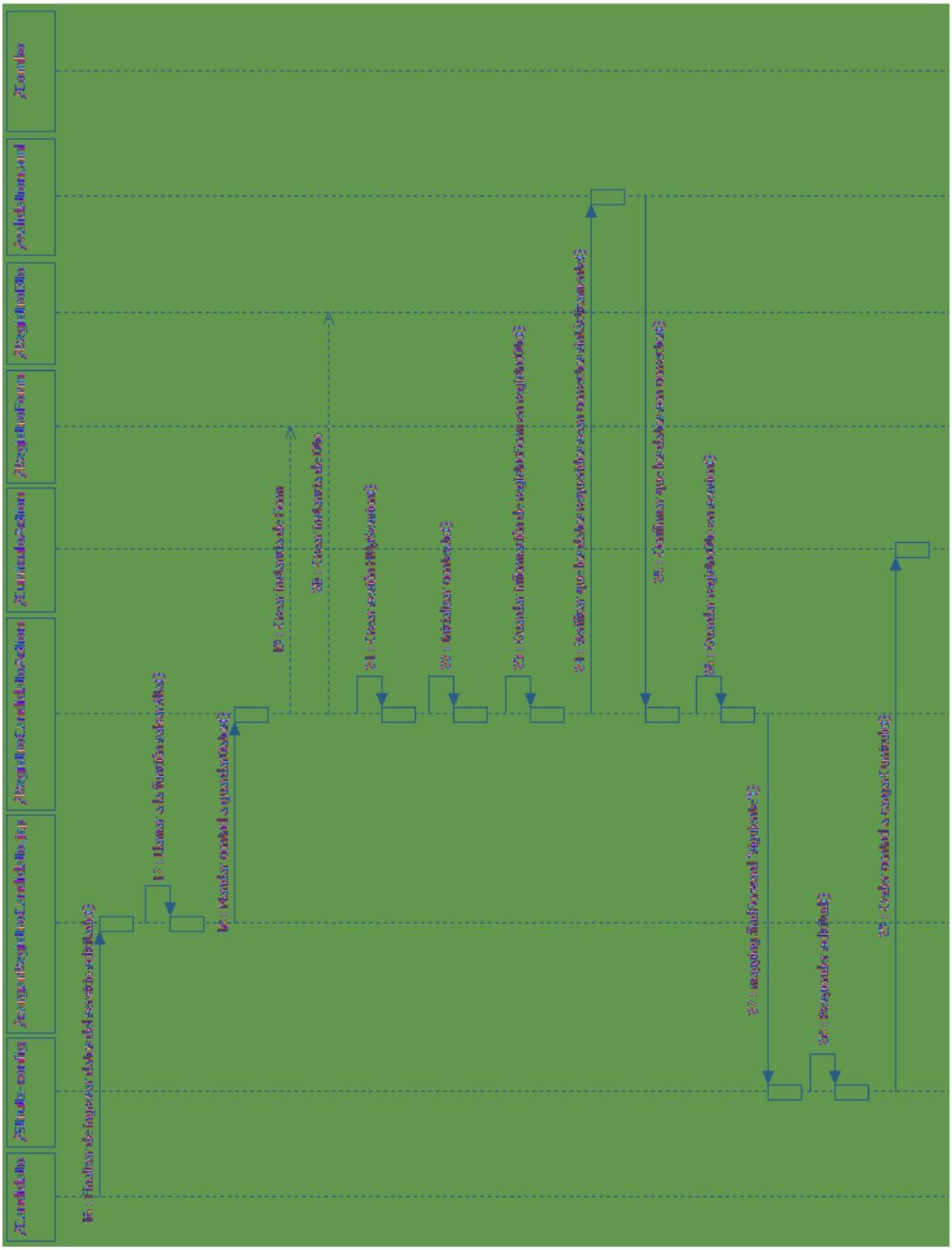


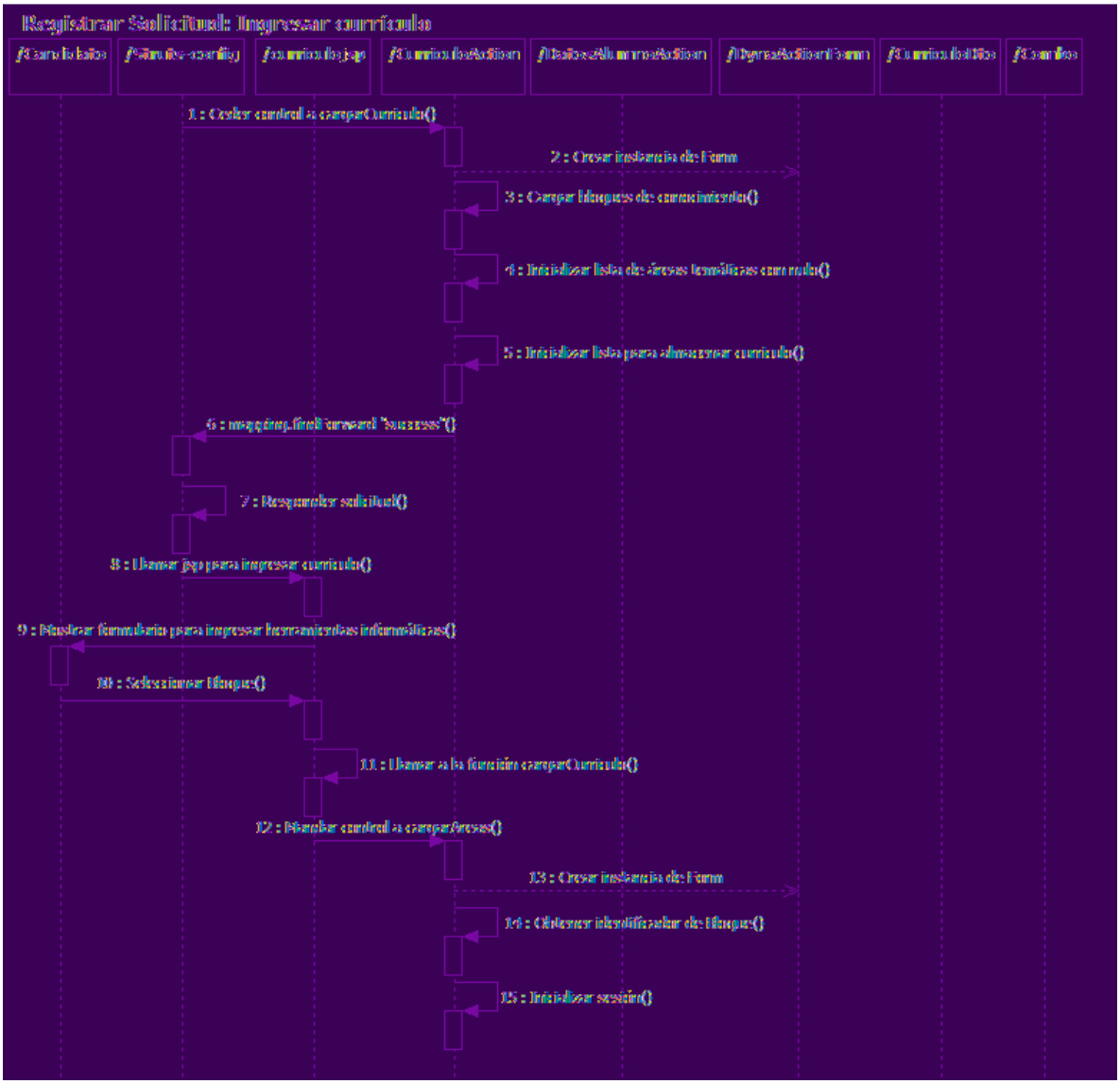
8.7. Anexo G: Diagrama de Clases

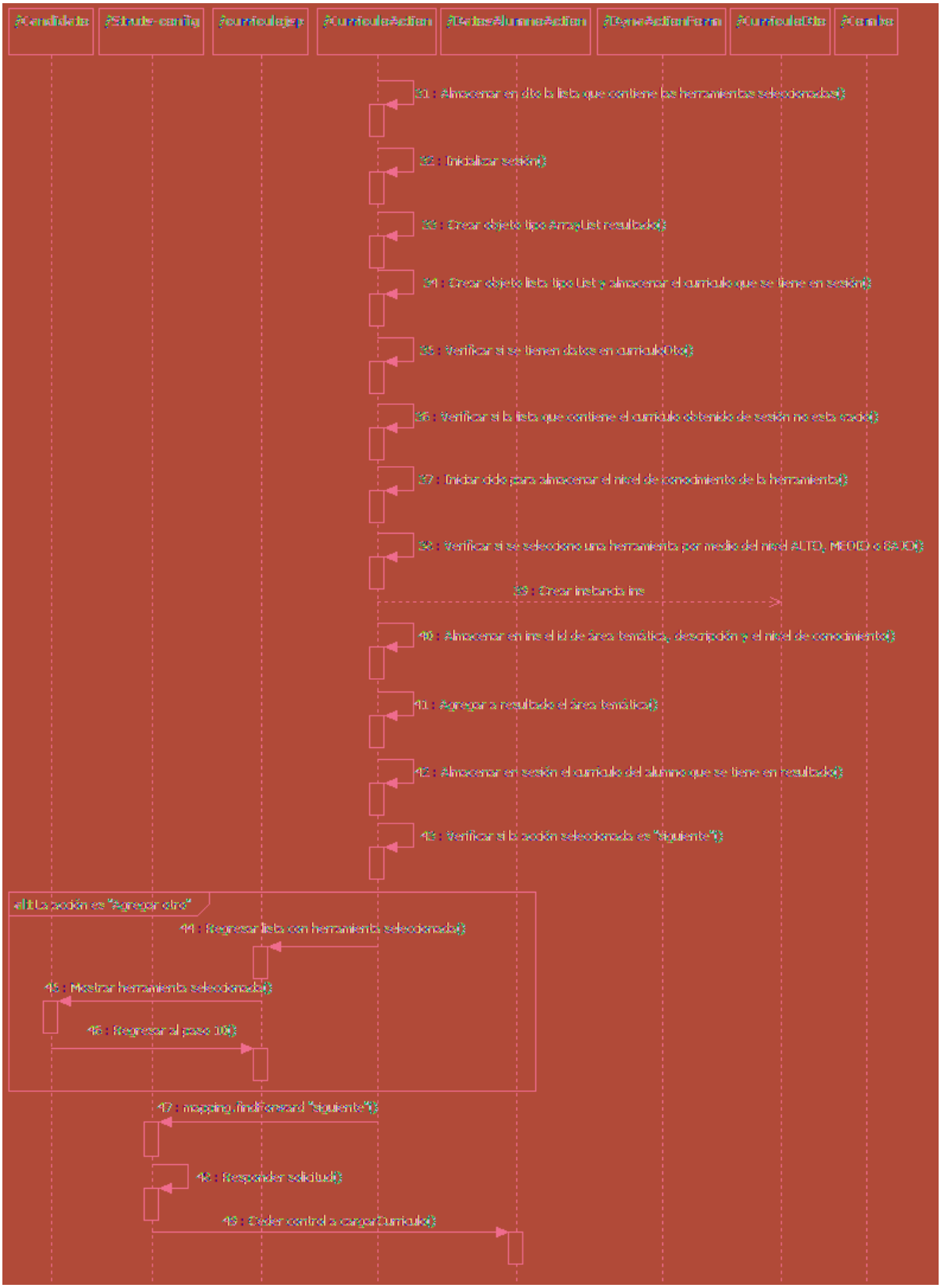


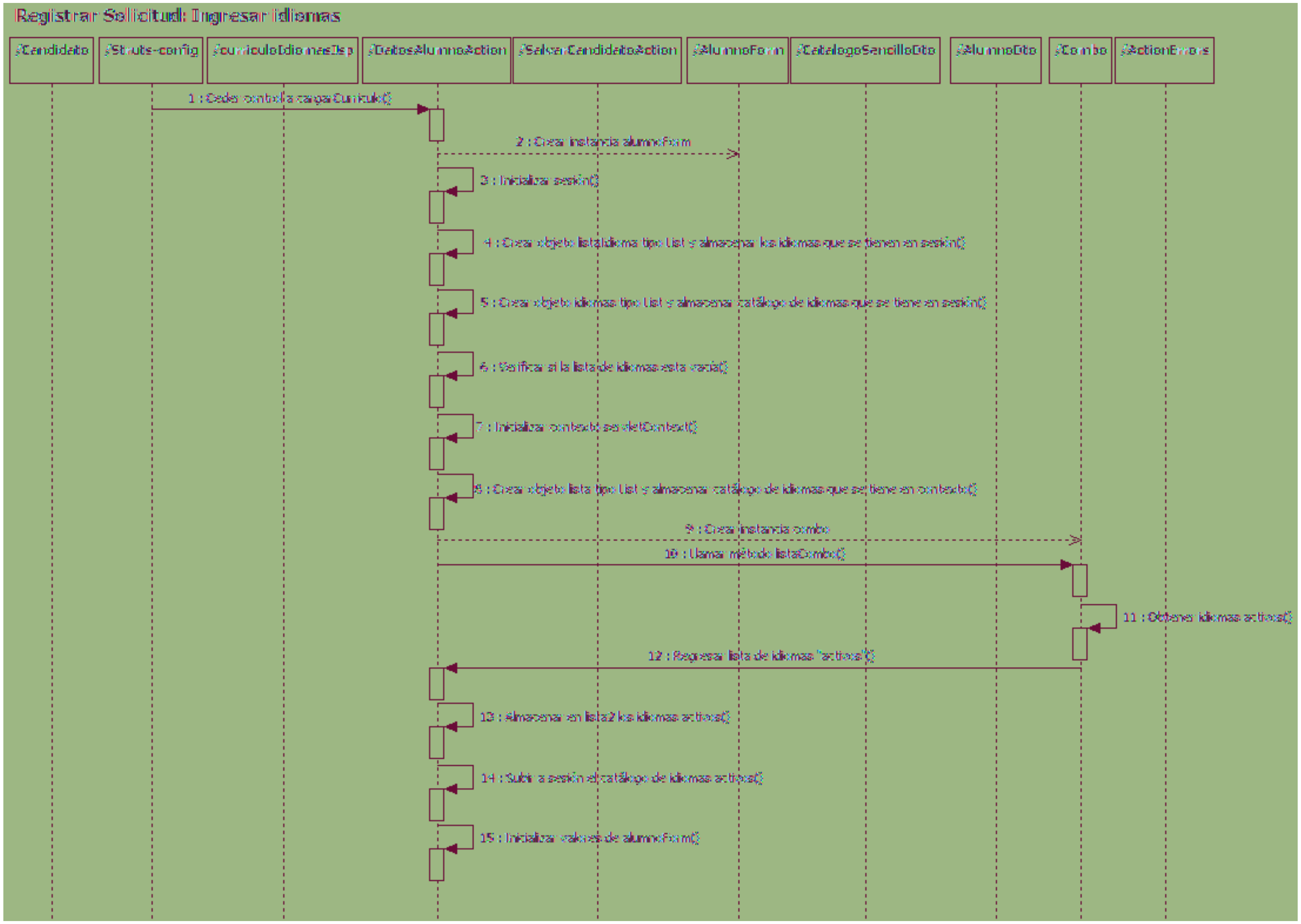
8.8. Anexo H: Diagramas de Secuencia



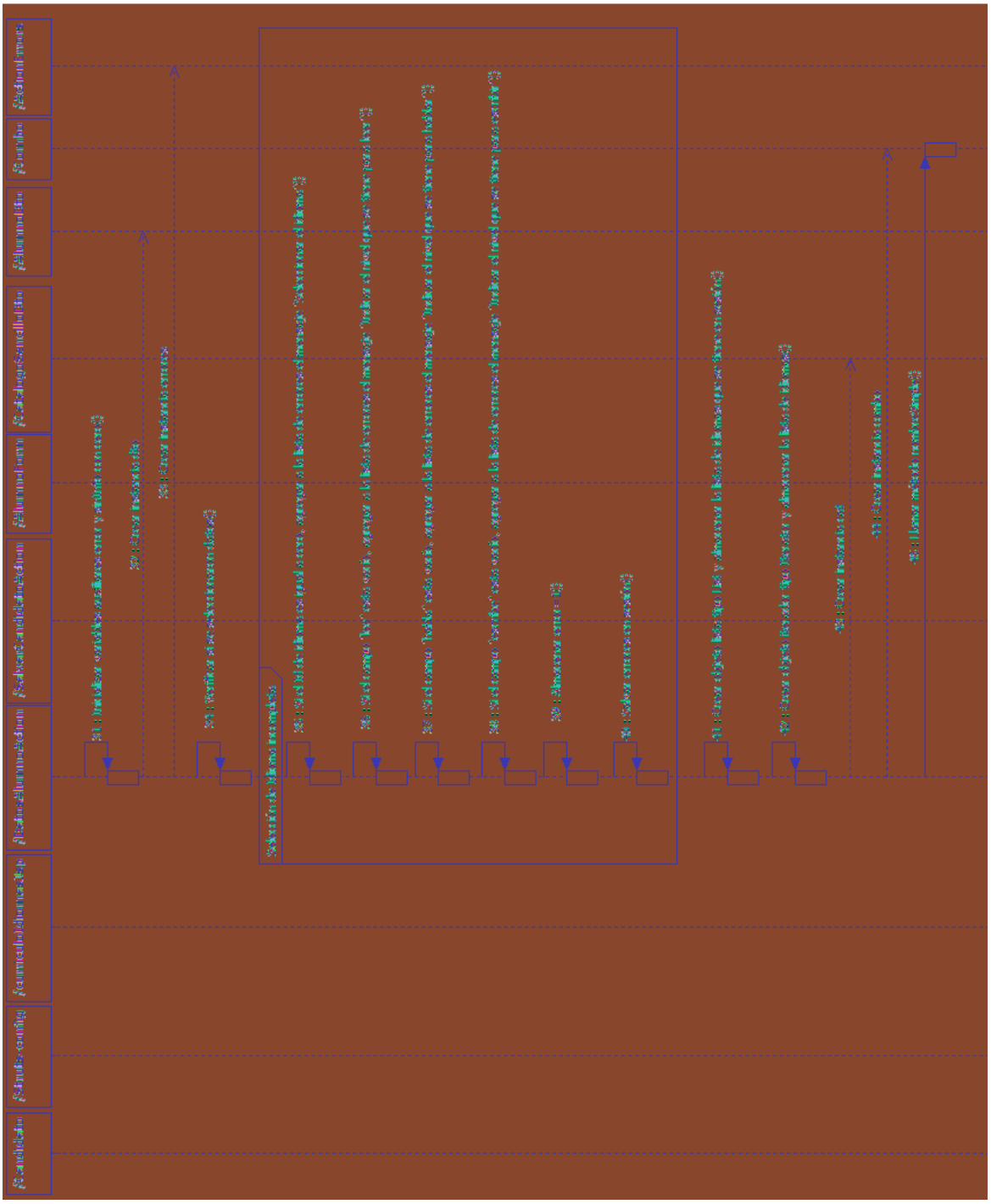


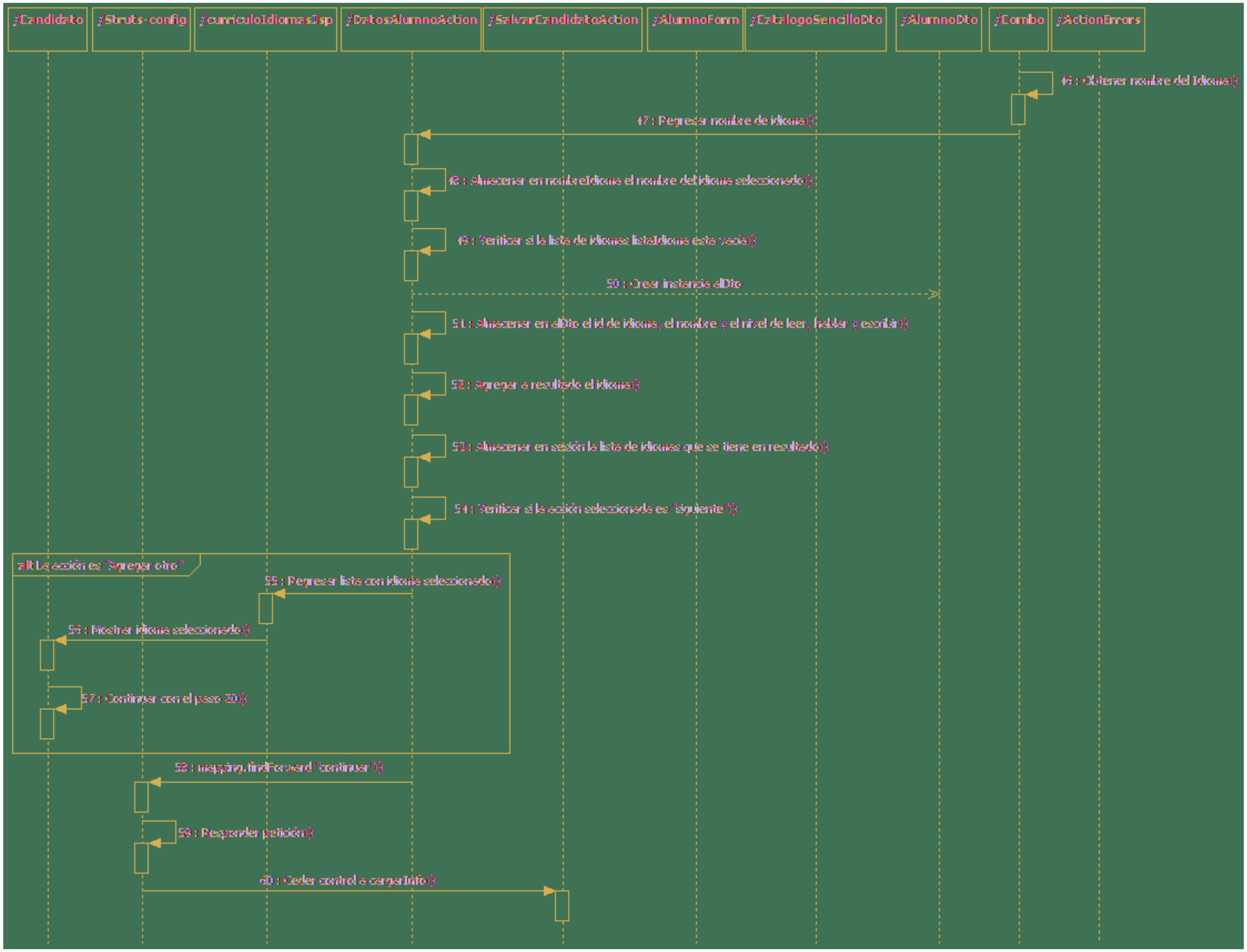

Registrar Solicitud: Ingresar curriculo
1 : Crear control a cargarCurriculo()
2 : Crear instancia de Form
3 : Cargar Ligues de conocimiento()
4 : Inicializar lista de áreas temáticas curriculo()
5 : Inicializar lista para almacenar curriculo()
6 : mapping.findForward "success"()
7 : Responder solicitud()
8 : Llamar jsp para ingresar curriculo()
9 : Mostrar formulario para ingresar herramientas informáticas()
10 : Seleccionar Ligues()
11 : Llamar a la función cargarCurriculo()
12 : Mandar control a cargarAreas()
13 : Crear instancia de Form
14 : Obtener identificador de Ligues()
15 : Inicializar session()


:Candidato
:Struts-config
:curriculum.jsp
:CurriculoAction
:DatosAlumnoAction
:DynaActionForm
:CurriculoDto
:Combo
31 : Almacenar en dto la lista que contiene las herramientas seleccionadas()
32 : Iniciar sesión()
33 : Crear objeto tipo ArrayList resultado()
34 : Crear objeto lista tipo List y almacenar el currículo que se tiene en sesión()
35 : Verificar si se tienen datos en currículoDto()
36 : Verificar si la lista que contiene el currículo obtenido de sesión no esta vacía()
37 : Iniciar ciclo para almacenar el nivel de conocimiento de la herramienta()
38 : Verificar si se seleccionó una herramienta por medio del nivel AUTO, MEDIO o BAJO()
39 : Crear instancia ins
40 : Almacenar en ins el id de área temática, descripción y el nivel de conocimiento()
41 : Agregar a resultado el área temática()
42 : Almacenar en sesión el currículo del alumno que se tiene en resultado()
43 : Verificar si la acción seleccionada es "Siguiente"()
alt La acción es "Agregar otro"
44 : Regresar lista con herramienta seleccionada()
45 : Mostrar herramienta seleccionada()
46 : Regresar al paso 10()
47 : mapping.findForward "siguiente"()
48 : Responder solicitud()
49 : Ceder control a cargarCurriculo()

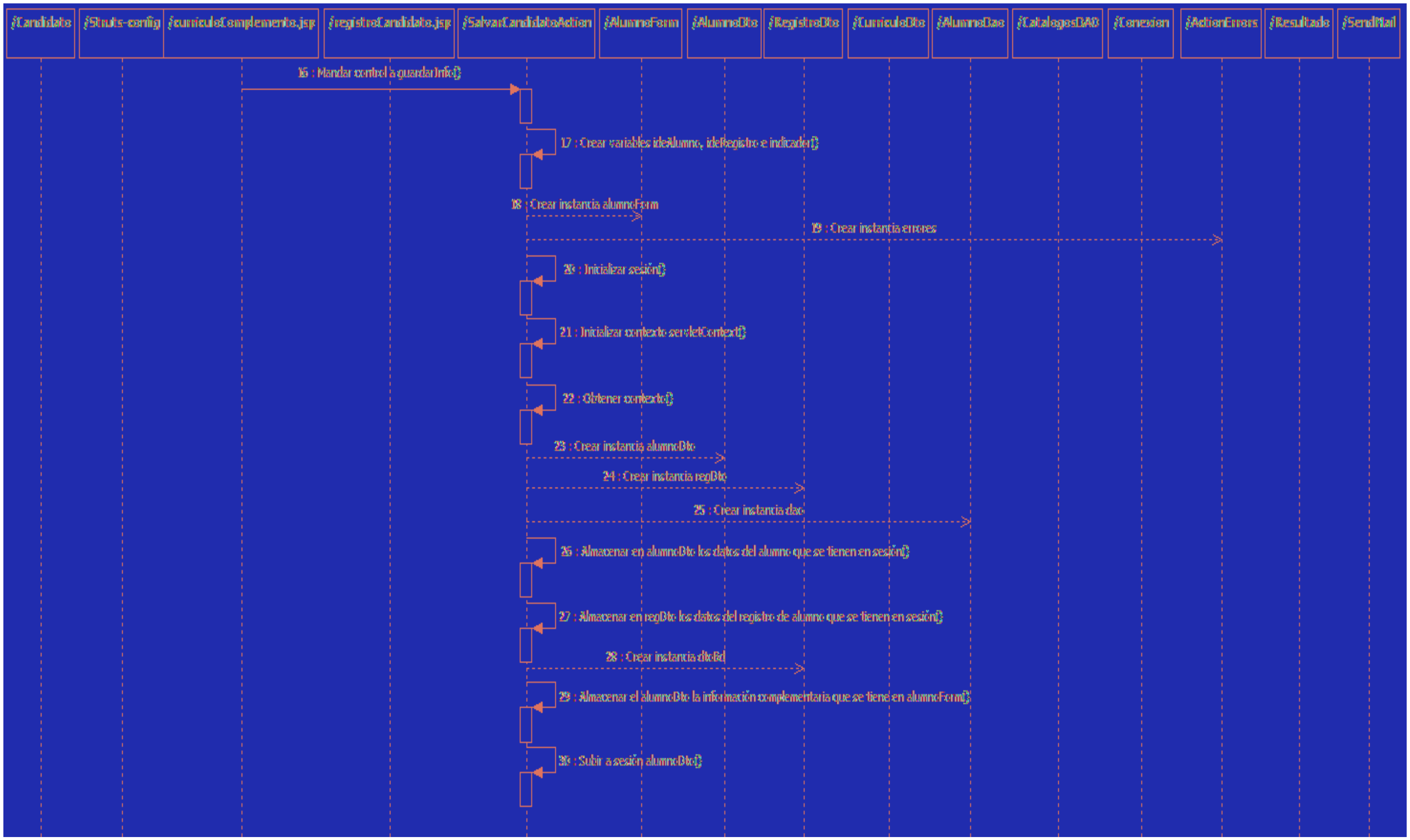

Registrar Solicitud: Ingresar idiomas
/Candidato
/Struts-config
/curriculoIdiomasJsp
/DatosAlumnoAction
/SalvarCandidatoAction
/AlumnoForm
/CatalogoSencilloDto
/AlumnoDto
/Combo
/ActionErrors
1 : Ceder control a cargarCurriculo()
2 : Crear instancia alumnoForm
3 : Inicializar sesión()
4 : Crear objeto listaIdioma tipo List y almacenar los idiomas que se tienen en sesión()
5 : Crear objeto idiomas tipo List y almacenar catálogo de idiomas que se tiene en sesión()
6 : Verificar si la lista de idiomas esta vacia()
7 : Inicializar contexto servletContext()
8 : Crear objeto lista tipo List y almacenar catálogo de idiomas que se tiene en contexto()
9 : Crear instancia combo
10 : Llamar método listaCombo()
11 : Obtener idiomas activos()
12 : Regresar lista de idiomas "activos"()
13 : Almacenar en lista2 los idiomas activos()
14 : Subir a sesión el catálogo de idiomas activos()
15 : Inicializar valores de alumnoForm()

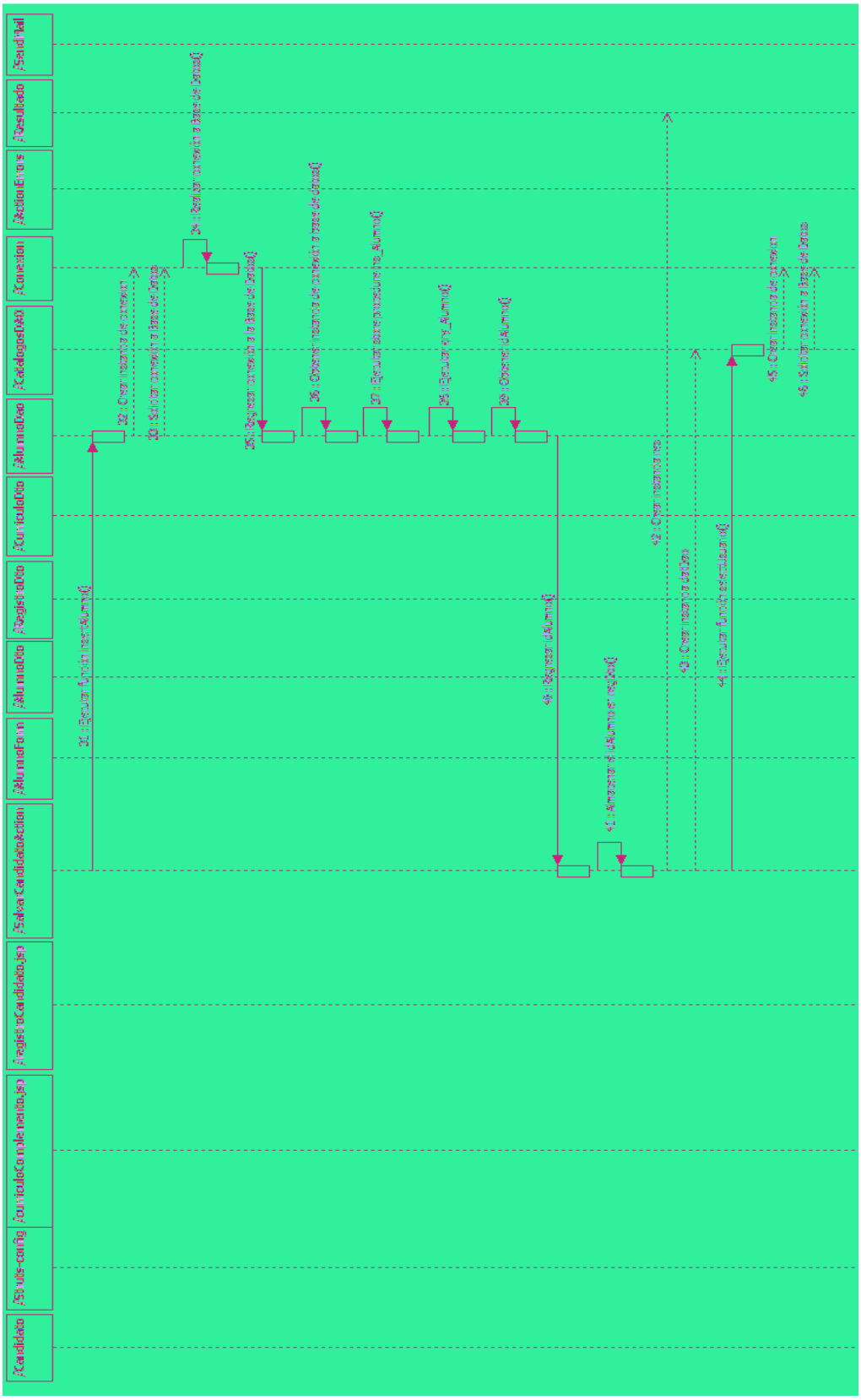

/Candidato
/Struts-config
/curriculoIdiomasJsp
/DatosAlumnoAction
/SalvarCandidatoAction
/AlumnoForm
/CatalogoSencilloDto
/AlumnoDto
/Combo
/ActionErrors
16 : mapping.findForward "complemento"()
17 : Responder solicitud()
18 : Llamar jsp para insertar idiomas()
19 : Mostrar formulario para ingresar idiomas()
20 : Seleccionar idioma()
21 : Seleccionar nivel de manejo alto, medio o bajo, para habla, escribe o lee()
22 : Indicar acción()
23 : Llamar a función salvarAlta()
24 : Mandar control a agregarIdioma()
25 : Crear instancia alumnoForm
26 : Inicializar sesión()
27 : Crear objeto tipo ArrayList resultado()
28 : Crear objeto lista tipo List y almacenar la lista de idiomas que se tiene en sesión()
29 : Inicializar el nombre de idioma con ""()
30 : Inicializar bandera auxiliar con true()

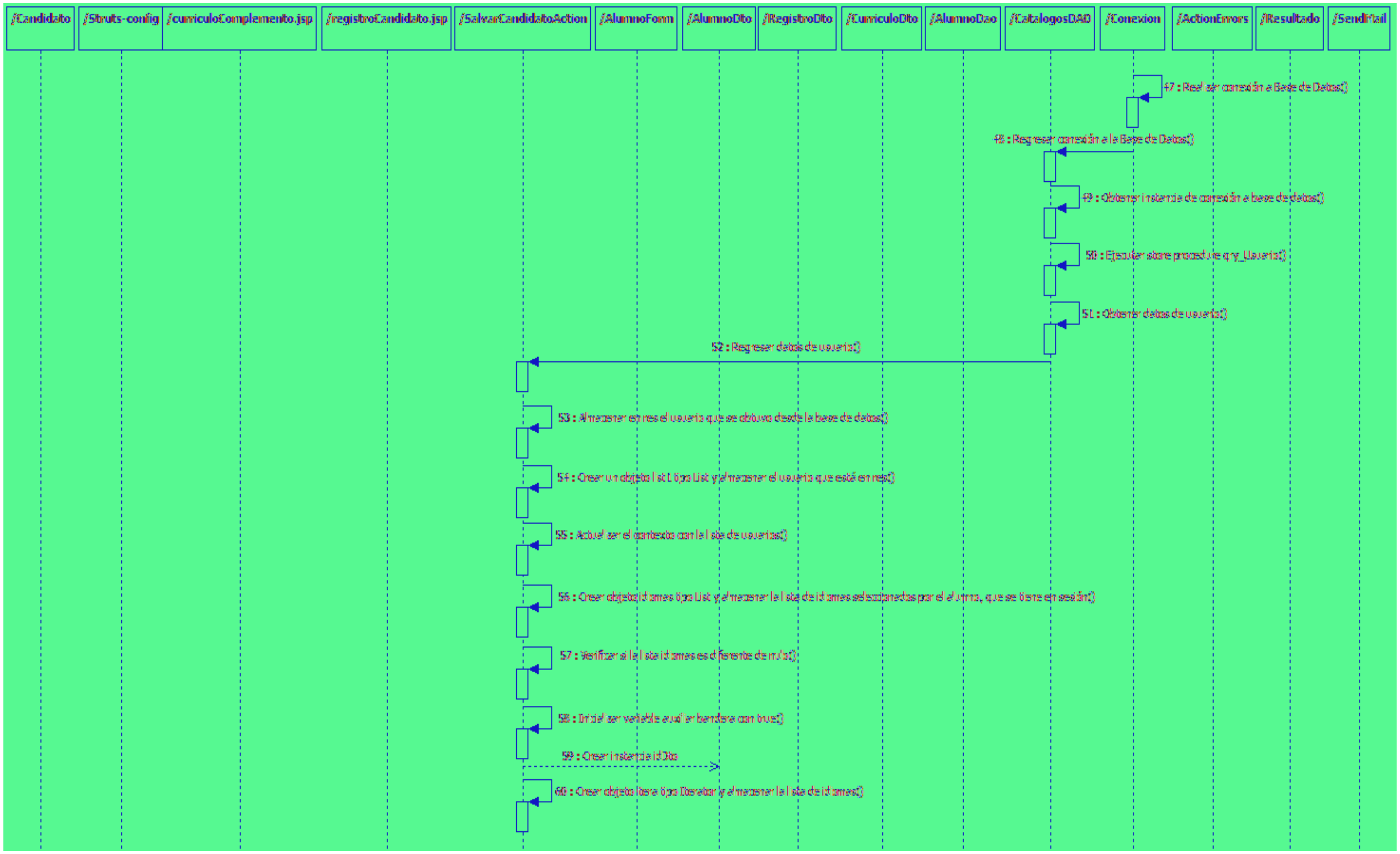



/Candidato
/Struts-config
/curriculoIdiomasJsp
/DatosAlumnoAction
/SalvarCandidatoAction
/AlumnoForm
/CatalogoSencilloDto
/AlumnoDto
/Combo
/ActionErrors
46 : Obtener nombre del Idioma()
47 : Regresar nombre de idioma()
48 : Almacenar en nombreIdioma el nombre del idioma seleccionado()
49 : Verificar si la lista de idiomas listaIdioma esta vacia()
50 : Crear instancia alDto
51 : Almacenar en alDto el id de idioma, el nombre y el nivel de leer, hablar y escribir()
52 : Agregar a resultado el idioma()
53 : Almacenar en sesión la lista de idiomas que se tiene en resultado()
54 : Verificar si la acción seleccionada es "siguiente"()
alt La acción es "Agregar otro"
55 : Regresar lista con idioma seleccionado()
56 : Mostrar idioma seleccionado()
57 : Continuar con el paso 20()
58 : mapping.findForward "continuar"()
59 : Responder petición()
60 : Ceder control a cargarInfo()

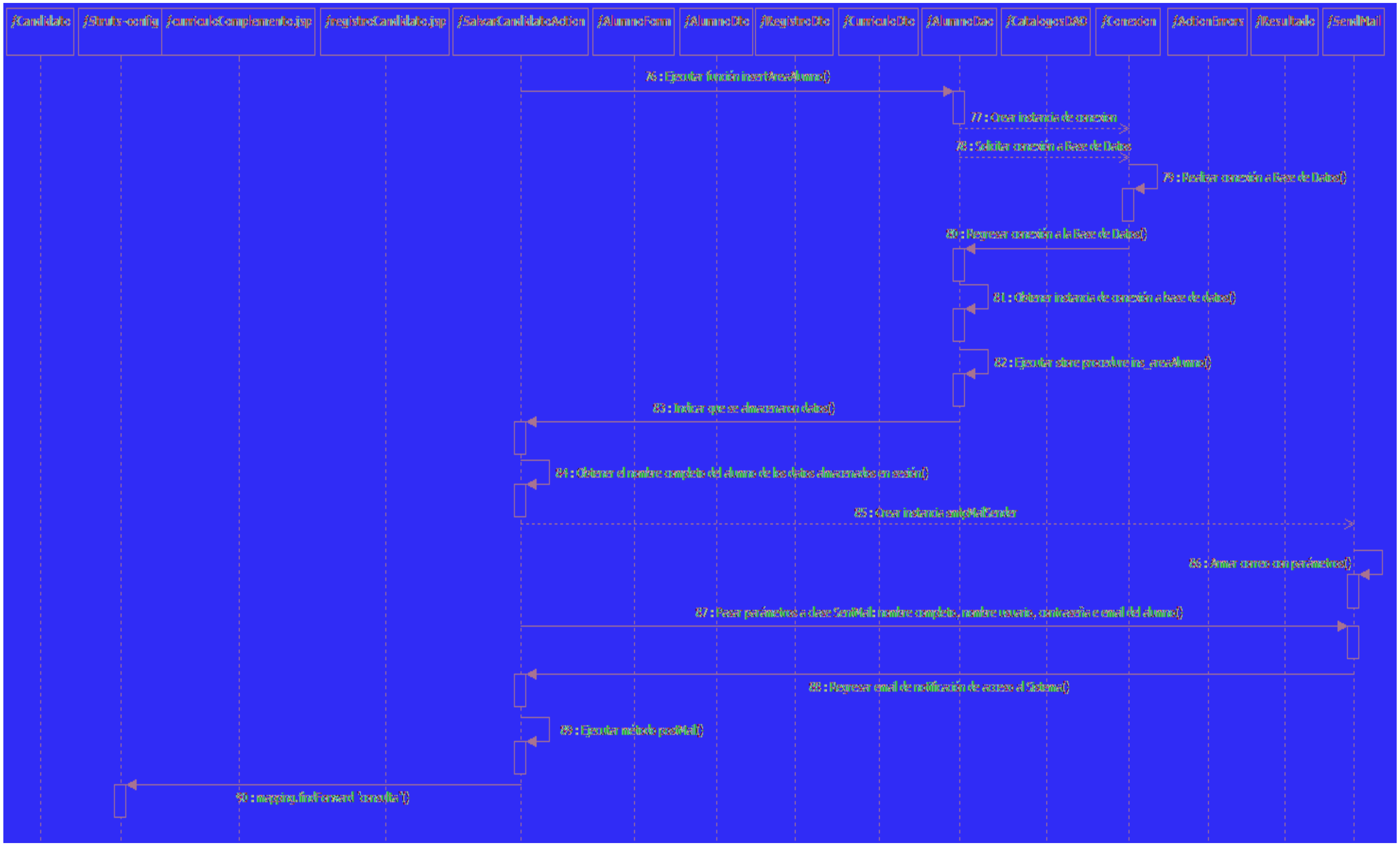

Registrar Solicitud: Ingresar complemento
/Candidato
/Struts-config
/curriculoComplemento.jsp
/registroCandidato.jsp
/SalvarCandidatoAction
/AlumnoForm
/AlumnoDto
/RegistroDto
/CurriculoDto
/AlumnoDao
/CatalogosDAO
/Conexion
/ActionErrors
/Resultado
/SendMail
1 : Ceder control a cargarInfo()
2 : Crear instancia alumnoForm
3 : Inicializar sesion()
4 : Crear instancia alumnoDto
5 : Obtener de sesión la información complementaria del alumno y almacenarla en alumnoDto()
6 : Verificar si alumnoDto esta vacio()
7 : Inicializar campos de alumnoForm()
8 : mapping.findForward("informacion")
9 : Responder solicitud()
10 : Llamar jsp para ingresar complemento de curriculo()
11 : Mostrar formulario para ingresar experiencia e intereses()
12 : Ingresar experiencia académica, profesional e intereses()
13 : Indicar que la información es verídica()
14 : Solicitar guardar información()
15 : Llamar a la función salvarAlta()

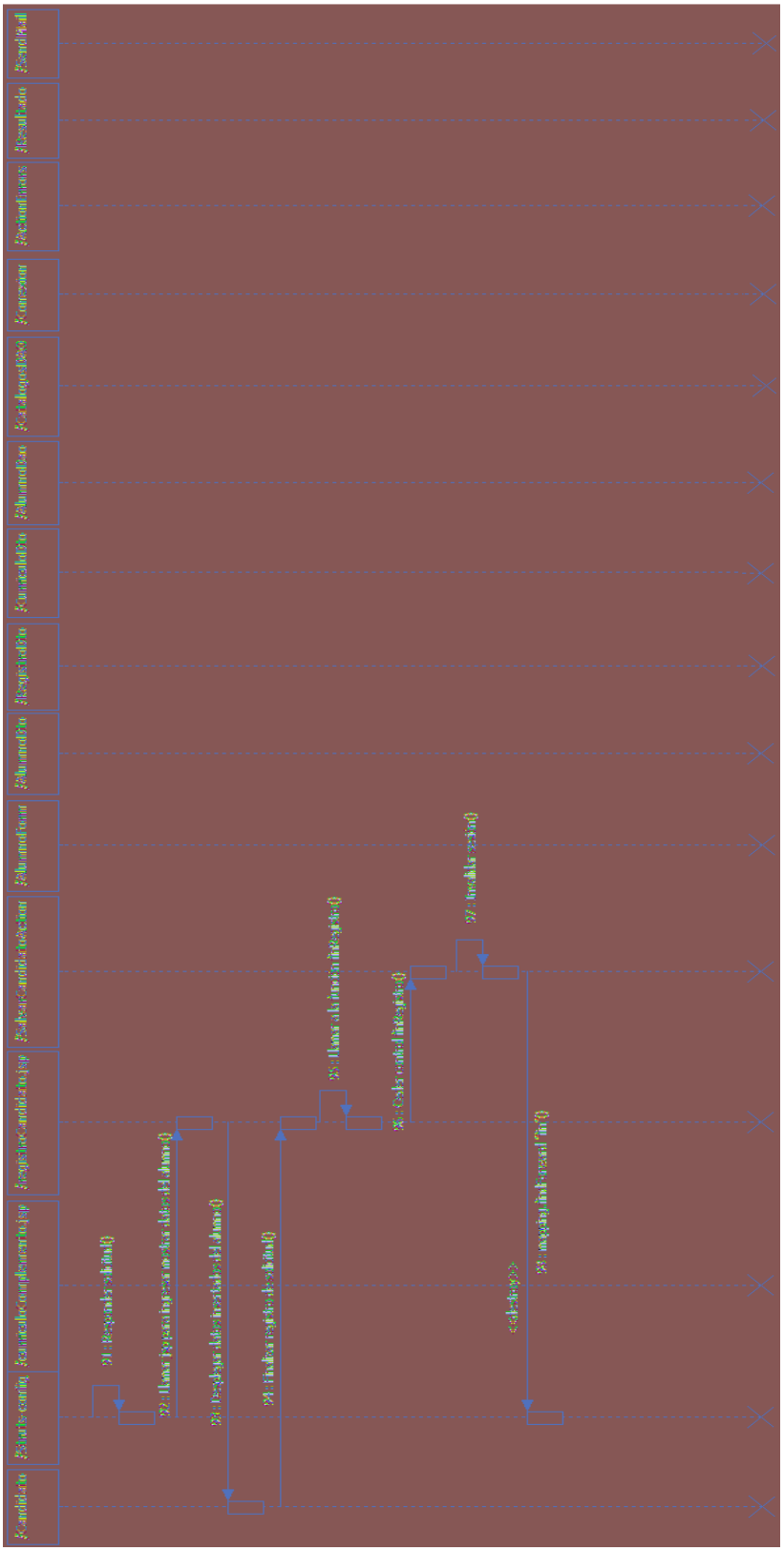

/Candidato
/Struts-config
/curriculoComplemento.jsp
/registroCandidato.jsp
/SalvarCandidatoAction
/AlumnoForm
/AlumnoDto
/RegistroDto
/CurriculoDto
/AlumnoDao
/CatalogosDAO
/Conexion
/ActionErrors
/Resultado
/SendMail
16 : Mandar control a guardarInfo()
17 : Crear variables idAlumno, idRegistro e indicador()
18 : Crear instancia alumnoForm
19 : Crear instancia errores
20 : Inicializar sesión()
21 : Inicializar contexto servletContext()
22 : Obtener contexto()
23 : Crear instancia alumnoDto
24 : Crear instancia regDto
25 : Crear instancia dao
26 : Almacenar en alumnoDto los datos del alumno que se tienen en sesión()
27 : Almacenar en regDto los datos del registro de alumno que se tienen en sesión()
28 : Crear instancia dtoBd
29 : Almacenar el alumnoDto la información complementaria que se tiene en alumnoForm()
30 : Subir a sesión alumnoDto()


:Candidato
:Servlet-config
:CurriculoComplementario.jsp
:RegistroCandidato.jsp
:EvaluarCandidatoAction
:AlumnoForm
:AlumnoDao
:RegistroDao
:AlumnoDao
:CurriculoDao
:AlumnoDao
:CatalogosDAO
:Conexion
:ActionErrors
:Resultado
:ResultSet

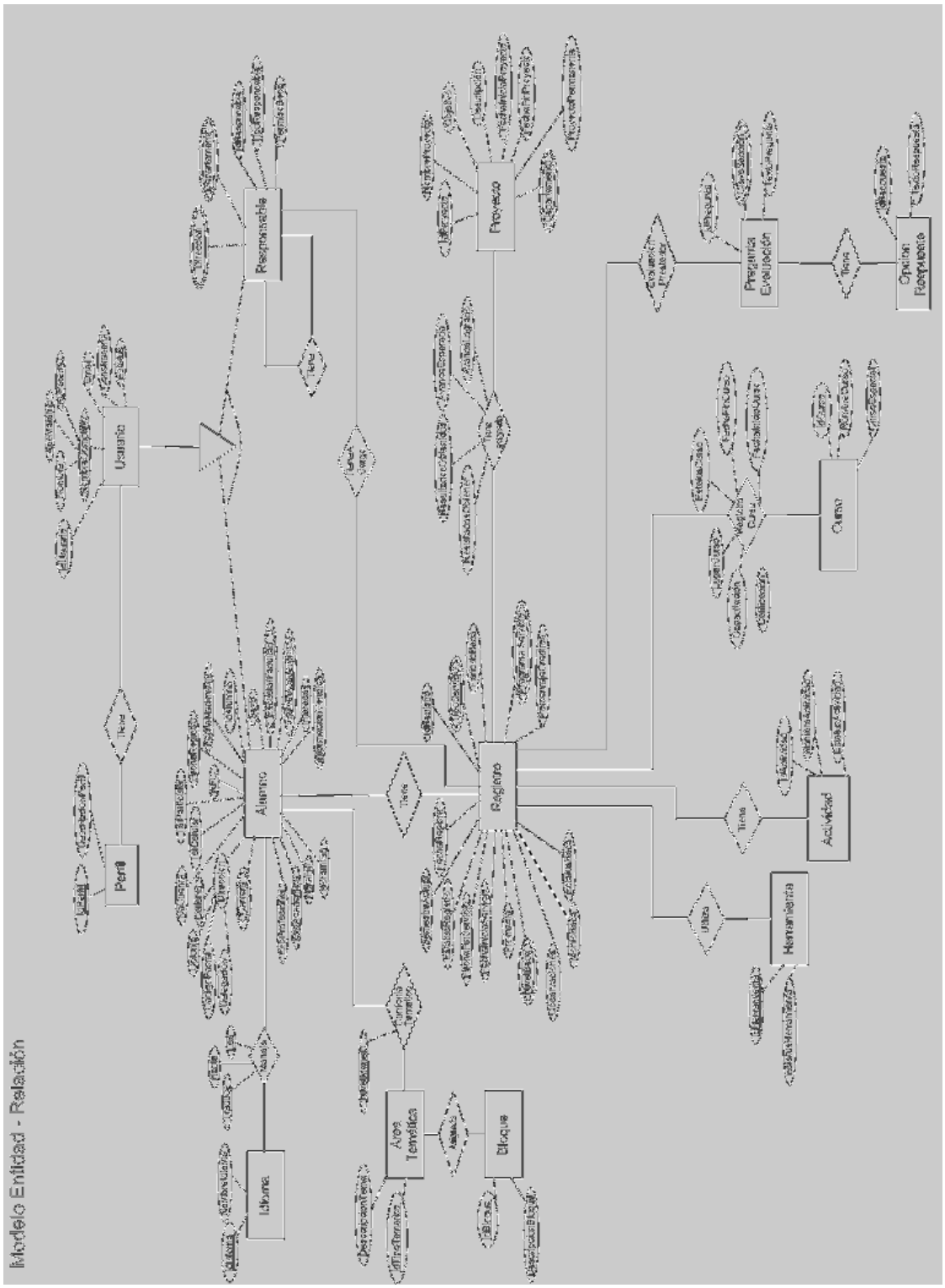

/Candidato
/Struts-config
/curriculoComplemento.jsp
/registroCandidato.jsp
/SalvarCandidatoAction
/AlumnoForm
/AlumnoDto
/RegistroDto
/CurriculoDto
/AlumnoDao
/CatalogosDAO
/Conexion
/ActionErrors
/Resultado
/SendMail
47 : Realizar conexión a Base de Datos()
48 : Regresar conexión a la Base de Datos()
49 : Obtener instancia de conexión a base de datos()
50 : Ejecutar store procedure qry_Usuario()
51 : Obtener datos de usuario()
52 : Regresar datos de usuario()
53 : Almacenar en res el usuario que se obtuvo desde la base de datos()
54 : Crear un objeto list1 tipo List y almacenar el usuario que está en res()
55 : Actualizar el contexto con la lista de usuarios()
56 : Crear objeto idiomas tipo List y almacenar la lista de idiomas seleccionados por el alumno, que se tiene en sesión()
57 : Verificar si la lista idiomas es diferente de nula()
58 : Inicializar variable auxiliar bandera con true()
59 : Crear instancia idDto
60 : Crear objeto itera tipo Iterator y almacenar la lista de idiomas()


/Candidato
/Struts-config
/curriculoComplemento.jsp
/registroCandidato.jsp
/SalvarCandidatoAction
/AlumnoForm
/AlumnoDto
/RegistroDto
/CurriculoDto
/AlumnoDao
/CatalogosD&D
/Conexion
/ActionErrors
/Resultado
/SendMail
61 : Crear instancia idiomaDto
62 : Almacenar en idiomaDto el id de idioma, el nivel de leer, hablar, escribir y el idAlumno()
63 : Ejecutar función insertIdioma()
64 : Crear instancia de conexion
65 : Solicitar conexión a Base de Datos
66 : Realizar conexión a Base de Datos()
67 : Regresar conexión a la Base de Datos()
68 : Obtener instancia de conexión a base de datos()
69 : Ejecutar store procedure ins_nivelIdioma()
70 : Indicar que se almacena información()
71 : Crear objeto conocimiento tipo List y almacenar la lista de herramientas seleccionadas por el alumno, que se tiene en sesión()
72 : Verificar si la lista conocimientos diferente de nulo()
73 : Iniciar variable auxiliar bandera con true()
74 : Crear instancia dudaDto()
75 : Almacenar en dudaDto el id de área temática, el nivel de conocimiento y el idAlumno()

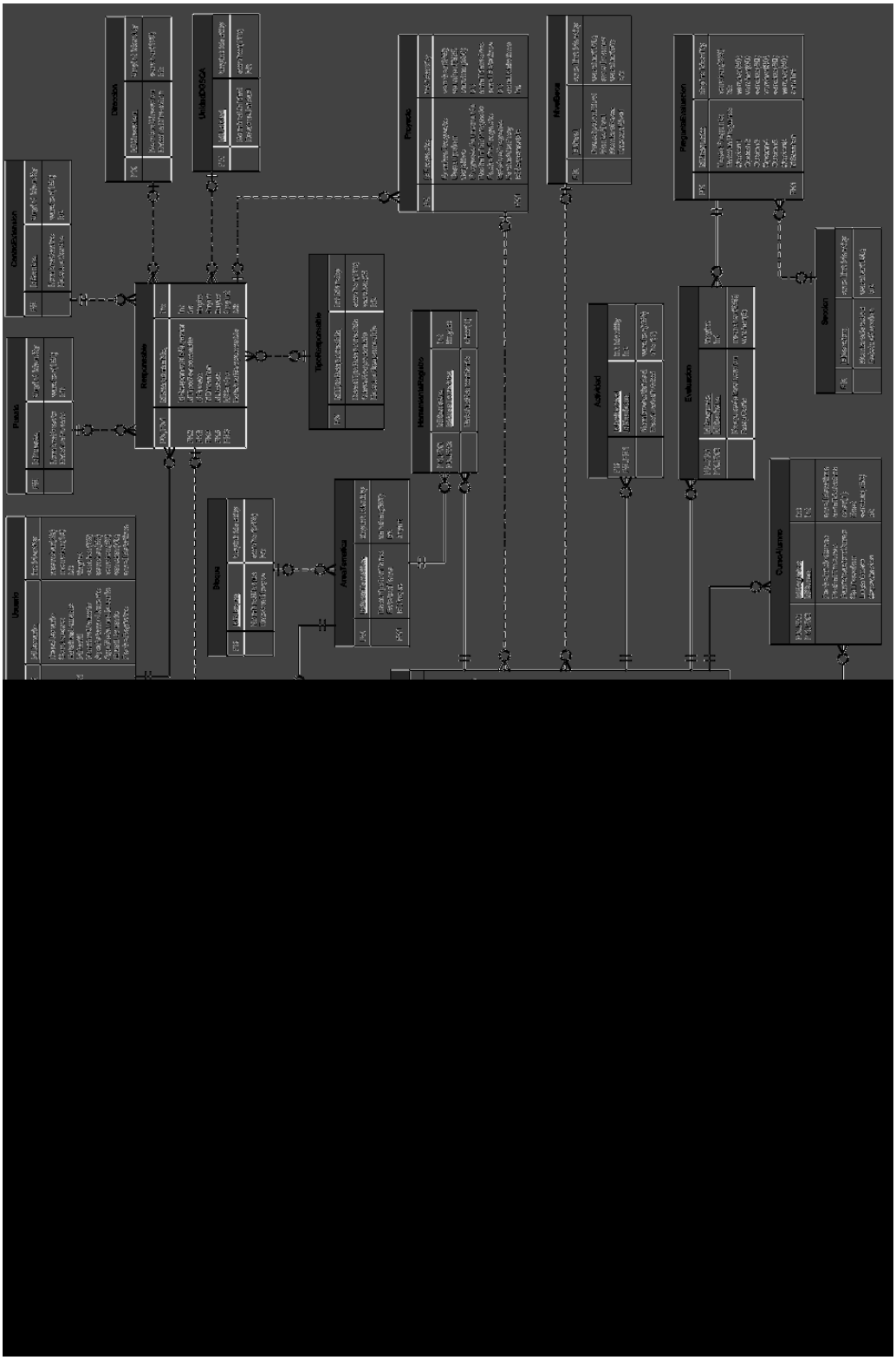

/Candidato
/Struts-config
/curriculoComplemento.jsp
/registroCandidato.jsp
/SalvarCandidatoAction
/AlumnoForm
/AlumnoDto
/RegistroDto
/CurriculoDto
/AlumnoDao
/CatalogosDAO
/Conexion
/ActionErrors
/Resultado
/SendMail
76 : Ejecutar función insertAreaAlumno()
77 : Crear instancia de conexion
78 : Solicitar conexión a Base de Datos
79 : Realizar conexión a Base de Datos()
80 : Regresar conexión a la Base de Datos()
81 : Obtener instancia de conexión a base de datos()
82 : Ejecutar store procedure ins_areaAlumno()
83 : Indicar que se almacenaron datos()
84 : Obtener el nombre completo del alumno de los datos almacenados en sesión()
86 : Armar correo con parámetros()
88 : Regresar email de notificación de acceso al Sistema()
89 : Ejecutar método postMail()



8.9. Anexo I: Modelo E – R



8.10. Anexo J: Modelo Físico de la Base de Datos



8.11. Anexo K: Diccionario de Datos

ControlPrestadores

Database summary

Target DBMS:	Microsoft SQL Server
Number of tables:	35
Number of columns:	208
Number of foreign keys:	38

Tables	Columns	Foreign keys
Actividad	4	1
Alumno	21	5
AreaTematica	4	1
Bloque	3	0
Carrera	3	0
CentroExtension	3	0
Conocimiento	3	2
Curso	4	1
CursoAlumno	8	2
Direccion	3	0
EscuelaCarrera	2	2
EscuelaFacultad	5	2
Evaluacion	4	2
HerramientaRegistro	3	2
Idioma	3	0
Institucion	4	0
NivelAcademico	3	0
NivelBeca	5	0
NivelIdioma	5	2
Perfil	4	0
PreguntaEvaluacion	10	1
ProgramaServicio	5	0
Proyecto	10	1
Puesto	3	0
Registro	33	4
Responsable	8	6
Seccion	3	0
TipoAlumno	4	0
TipoCurso	4	0
TipoResponsable	4	0
TipoServicio	4	0
UnidadDGSCA	3	0
Usuario	10	1

Table: Actividad

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdActividad	PK	int identity	[0-9]	Not allowed	Clave que identifica cada una de las actividades registradas por un prestador, en el sistema.
IdRegistro	PK, FK	int	[0-9]	Not allowed	Identificador del Registro asociado a la actividad reportada.
NombreActividad		varchar(255)	Todos los valores	Not allowed	Descripción de la actividad.
EstatusActividad		char(1)	P = Propuesta R = Realizada N = No propuesta y realizada	Not allowed	Estatus que indica si la actividad fue propuesta en un plan de trabajo, si fue propuesta y realizada o bien si no fue propuesta y se realizó.

Table: Alumno

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdAlumno	PK, FK	int	[0-9]	Not allowed	Identificador del Alumno, corresponde al identificador de Usuario que tiene en el sistema.
NoCuenta		nvarchar(15)	[0-9]	Allowed	Número o matrícula con la cual está registrado el Alumno en su Escuela o Facultad.
FechaNacimiento		smalldatetime	AAAAMMDD	Not allowed	Fecha de Nacimiento del Alumno.
RFC		varchar(15)	Todos los valores	Allowed	Registro Federal de Contribuyentes, si es que se tiene.
TelParticular		nvarchar(15)	[0-9]	Allowed	Teléfono particular del Alumno.
TelCelular		nvarchar(15)	[0-9]	Allowed	Teléfono celular del Alumno.
CalleNo		nvarchar(50)	Todos los valores	Not allowed	Nombre de la calle y número donde está ubicado el domicilio del Alumno.
Colonia		varchar(50)	Todos los valores	Not allowed	Nombre de la colonia donde está ubicado el domicilio del Alumno.
CodigoPostal		nvarchar(10)	[0-9]	Not allowed	Código Postal de la zona donde está ubicado el domicilio del Alumno.
Delegacion		varchar(50)	Todos los valores	Not allowed	Nombre de la Delegación donde está ubicado el domicilio del Alumno.
CURP		nvarchar(25)	Todos los valores	Allowed	Clave Única de Registro de Población.
Sexo		char(1)	F = Femenino M=Masculino	Not allowed	Género del Alumno: Femenino o Masculino.

Intereses		text	[A-Z][a-z]	Allowed	Breve descripción de los intereses del Candidato. Se llena cuando se solicita el servicio por primera vez
ExpAcademica		text	[A-Z][a-z]	Allowed	Breve descripción de la experiencia del Candidato en proyectos escolares. Se llena cuando se solicita el servicio por primera vez.
ExpProfesional		text	[A-Z][a-z]	Allowed	Breve descripción de la experiencia profesional del Candidato. Se llena cuando se solicita el servicio por primera vez.
InformacionVeridica		bit	0 = Default / sin confirmar 1 = Aceptación de veracidad	Not allowed	Bandera de confirmación de veracidad en la información proporcionada, si el Candidato acepta que la información es verídica, el campo toma el valor de 1.
PermisoBeca		bit	0 = Sin permiso 1 = Con permiso	Allowed	Este campo se activará con un valor de (1) cuando un Responsable haya dado permiso a un Alumno para solicitar una beca, cuando el campo tiene un valor de (0) significa que el alumno no tiene permiso de solicitar beca.
IdTipoAlumno	FK	tinyint	[0-9]	Not allowed	Identificador de Tipo Alumno.
IdEscuelaFacultad	FK	tinyint	[0-9]	Allowed	Identificador de Escuela Facultad a la que asiste el Alumno.
IdCarrera	FK	tinyint	[0-9]	Allowed	Identificador de Carrera en la que esta incsrito el Alumno.
IdResponsable	FK	int	[0-9]	Allowed	Identificador del Responsable que supervisa las actividades de un Alumno.

Table: AreaTematica

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdAreaTematica	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica cada uno de los lenguajes, bases de datos, paquetes que el Candidato maneja.
DescripcionTema		varchar(255)	Todos los valores	Not allowed	Descripción del área temática. Como áreas temáticas se tiene por ejemplo JAVA, SQL.

EstatusTema		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus que indica si el área de conocimiento se mostrará en las opciones (1) o no (0) al hacer una solicitud, para que un alumno pueda elegirla.
IdBloque	FK	tinyint	[0-9]	Allowed	Identificador del bloque de conocimiento al cuál pertenece un tema.

Table: Bloque

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdBloque	PK	tinyint identity	[0-9]	Not allowed	Identificador del bloque de conocimiento al cuál pertenece un tema.
NombreBloque		varchar(255)	Todos los valores	Not allowed	Es la descripción del bloque de conocimiento, por ejemplo: Bases de datos, Lenguajes de programación
EstatusBloque		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si el bloque esta activo (1) o no (0).

Table: Carrera

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdCarrera	PK	tinyint identity	[0-9]	Not allowed	Clave asignada a la Carrera.
NombreCarrera		varchar(255)	Todos los valores	Not allowed	Nombre de la Carrera.
EstatusCarrera		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus de la Carrera, indica si esta activa (1) o no (0).

Table: CentroExtension

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdCentro	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica a cada uno de los Centros pertenecientes a DGSCA.
NombreCentro		varchar(150)	Todos los valores	Not allowed	Nombre de cada uno de los Centros de DGSCA.
EstatusCentro		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus que indica si el centro esta activo (1) o no (0).

Table: Conocimiento

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdAlumno	PK, FK	int	[0-9]	Not allowed	Identificador del Alumno asociado a una área temática.
IdAreaTematica	PK, FK	tinyint	[0-9]	Not allowed	Identificador del área temática sobre la cuál se repota un nivel de conocimiento.
NivelManejo		varchar(10)	B = Básico M = Medio A = Avanzado	Not allowed	Nivel de manejo del área temática, el cuál será tomado como básico, medio o avanzado.

Table: Curso

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdCurso	PK	int identity	[0-9]	Not allowed	Clave que identifica a cada uno de los cursos registrados en el sistema.
NombreCurso		varchar(255)	Todos los valores	Not allowed	Nombre del curso.
EstatusCurso		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si el curso esta activo (1) o no (0).
IdTipoCurso	FK	tinyint	[0-9]	Allowed	Clave que identifica al Tipo de Curso.

Table: CursoAlumno

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdRegistro	PK, FK	int	[0-9]	Not allowed	Clave consecutiva para llevar el control de los registros de servicio de un Prestador.
IdCurso	PK, FK	int	[0-9]	Not allowed	Clave que identifica a cada uno de los cursos registrados en el sistema.
FechaInicioCurso		smalldatetime	AAAAMMDD	Not allowed	Fecha en la que inicia el curso.
FechaFinCurso		smalldatetime	AAAAMMDD	Not allowed	Fecha en la que finaliza el curso.
ParticipacionCurso		char(1)	T = Curso tomado I = Curso impartido	Not allowed	Indica si el curso fue tomado (T) o impartido por el Prestador (I).
Calificacion		float	[0-9], .	Not allowed	Calificación obtenida por el Prestador durante su participación en el curso.
LugarCurso		varchar(150)	Todos los valores	Allowed	Cede donde se impartió el curso.
Capacitacion		bit	0 = Curso que no esta en plan de capacitación 1 = Curso de plan de capacitación	Allowed	Bandera que indica si el curso se tomo dentro de un plan de capacitación o no.

Table: Direccion

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdDireccion	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica a cada Dirección de DGSCA.
NombreDireccion		varchar(150)	Todos los valores	Not allowed	Nombre de la Dirección.
EstatusDireccion		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si la Dirección esta activa o no.

Table: EscuelaCarrera

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdCarrera	PK,FK	tinyint	[0-9]	Not allowed	Identificador de Carrera.
IdEscuelaFacultad	PK,FK	tinyint	[0-9]	Not allowed	Identificador de la Escuela o Facultad donde se imparte una Carrera determinada.

Table: EscuelaFacultad

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdEscuelaFacultad	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica a cada una de las Escuelas o Facultades registradas en el sistema.
NombreEscuelaFacultad		varchar(255)	Todos los valores	Not allowed	Nombre de la Escuela o Facultad.
EstatusEscuelaFacultad		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus de la Escuela o Facultad, indica si esta activa o no. Sirve para indicar si se pueden recibir registros de esa Escuela o no.
IdInstitucion	FK	tinyint	[0-9]	Not allowed	Identificador de la Institución a la que pertenece la Escuela o Facultad.
IdNivelAcademico	FK	tinyint	[0-9]	Not allowed	Identificador del Nivel Académico que se imparte en la Escuela o Facultad.

Table: Evaluacion

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdPregunta	PK,FK	tinyint	[0-9]	Not allowed	Clave asociada a cada una de las preguntas incluidas en la Evaluación.
IdRegistro	PK,FK	int	[0-9]	Not allowed	Identificador del Registro asociado a la Evaluación
RespuestaEvaluacion		nvarchar(255)	Todos los valores	Not allowed	Respuesta asignada a la pregunta de Evaluación.
RespCorta		varchar(5)	Todos los valores	Allowed	Respuesta Corta que se puede dar a alguna pregunta de la Evaluación.

Table: HerramientaRegistro

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdRegistro	PK,FK	int	[0-9]	Not allowed	Identificador del Registro que esta asociado al reporte de una herramienta utilizada o propuesta para desarrollar un proyecto.

IdAreaTematica	PK,FK	tinyint	[0-9]	Not allowed	Identificador del área temática a la que pertenece la herramienta indicada.
EstatusHerramienta		char(1)	P = Propuesta U = Propuesta y utilizada N = No propuesta y utilizada	Not allowed	Estatus que indica si la herramienta fue propuesta, si fue propuesta y utilizada o si fue una herramienta que no se propuso en el plan de trabajo y que se utilizó.

Table: Idioma

Columns	KEY	Data type	Value/Range	Allow NULLs	Descrption
IdIdioma	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica a cada uno de los idiomas que están registrados en el sistema.
NombreIdioma		varchar(50)	Todos los valores	Not allowed	Es el nombre que identifica al idioma.
EstatusIdioma		bit	0 = Inactivo 1 = Activo	Not allowed	Estado que indica si el idioma está activo o no.

Table: Institucion

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdInstitucion	PK	tinyint identity	[0-9]	Not allowed	Clave de la Institución a la que pertenece una Escuela o Facultad.
NombreInstitucion		varchar(255)	Todos los valores	Not allowed	Nombre completo de la Institución Educativa.
EstatusInstitucion		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus de la Institución, indica si esta activa (1) o no (0).
NombreCortoInst		varchar(15)	Todos los valores	Not allowed	NombreCortoInst son las siglas de la Institución o el nombre abreviado.

Table: NivelAcademico

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdNivelAcademico	PK	tinyint identity	[0-9]	Not allowed	Clave asignada a la etapa académica.
NombreNivelAcademico		varchar(100)	Todos los valores	Not allowed	Nombre de la etapa académica, las cuáles pueden ser: Técnico, Bachillerato, Licenciatura, Posgrado, etcétera.
EstatusNivelAcademico		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si el nivel académico esta activo (1) o no (0).

Table: NivelBeca

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdNivel	PK	smallint identity	[0-9]	Not allowed	Clave que identifica los niveles para las becas existentes en DGSCA.

DescripcionNivel		varchar(100)	Todos los valores	Not allowed	Describe cada nivel de beca que se ofrece.
MontoNivel		smallmoney	[0-9]	Not allowed	Indica el monto fijado para cada nivel.
NombreBeca		varchar(50)	Todos los valores	Not allowed	Es el nombre con el que se hace referencia a una beca.
EstatusNivel		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si un nivel de beca esta activo o no.

Table: NivelIdioma

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdAlumno (FK)	PK,FK	int	[0-9]	Not allowed	Identificador del Alumno.
IdIdioma (FK)	PK,FK	tinyint	[0-9]	Not allowed	Identificador del Idioma que maneja un Alumno.
PorcentajeHabla		varchar(10)	Alto Medio Bajo	Allowed	Nivel que el Alumno habla del idioma seleccionado.
PorcentajeLee		varchar(10)	Alto Medio Bajo	Allowed	Nivel que el Alumno lee del idioma seleccionado.
PorcentajeTraduce		varchar(10)	Alto Medio Bajo	Allowed	Nivel que el Alumno traduce del idioma seleccionado.

Table: Perfil

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdPerfil	PK	tinyint identity	[0-9]	Not allowed	Clave que se le asignara a cada perfil para controlar acciones permitidas.
DescripcionPerfil		varchar(100)	Todos los valores	Not allowed	Descripción que identifica el perfil del usuario, los cuáles pueden ser: Administrador, Alumno, Candidato o Responsable.
EstatusPerfil		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si el perfil se encuentra activo o no.
ClavePerfil		varchar(2)	A = Administrador RS = Responsable de Servicio RA = Responsable de Área RP = Responsable de Prestador C = Candidato P = Prestador	Not allowed	Clave asignada al perfil, abreviatura.

Table: PreguntaEvaluacion

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdPregunta	PK	tinyint identity	[0-9]	Not allowed	Clave asociada a cada una de las preguntas incluidas en la evaluación.

TextoPregunta		varchar(255)	Todos los valores	Not allowed	Texto de cada pregunta.
EstatusPregunta		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus de la pregunta, indica si esta activa (1) o no (0) en el cuestionario de evaluación.
Opcion1		varchar(80)	Todos los valores	Allowed	Es la primera opción que aparecerá como posible respuesta de la pregunta.
Opcion2		varchar(80)	Todos los valores	Allowed	Es la segunda opción que aparecerá como posible respuesta de la pregunta.
Opcion3		varchar(80)	Todos los valores	Allowed	Es la tercera opción que aparecerá como posible respuesta de la pregunta.
Opcion4		varchar(80)	Todos los valores	Allowed	Es la cuarta opción que aparecerá como posible respuesta de la pregunta.
Opcion5		varchar(80)	Todos los valores	Allowed	Es la quinta opción que aparecerá como posible respuesta de la pregunta.
Opcion6		varchar(80)	Todos los valores	Allowed	Es la sexta opción que aparecerá como posible respuesta de la pregunta.
IdSeccion	FK	smallint	[0-9]	Allowed	Identificador de la sección a la que pertenece una pregunta.

Table: ProgramaServicio

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdProgramaServicio	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica a cada uno de los programas de servicio que se tienen registrados en el sistema.
NombreProgServicio		varchar(255)	Todos los valores	Not allowed	Nombre del Programa de Servicio.
DescripcionProgServicio		varchar(255)	Todos los valores	Not allowed	Breve descripción del Programa de Servicio.
Indicador		char(1)	I = Programa de servicio interno E = Programa de servicio externo	Not allowed	Indica si el programa de servicio pertenece a DGSCA o bien si es de alguna otra institución u organismo gubernamental.
Periodo		nvarchar(25)	Todos los valores	Not allowed	Clave con la cuál se tiene registrado el Programa de Servicio Social ante la DGOSE, dicha clave se genera anualmente. En el caso de las becas se

					tiene la clave correspondiente al periodo, por ejemplo 2007-2
FechaInicioRegistro		smalldatetime	AAAAMMDD	Not allowed	Fecha de inicio en la que se puede solicitar un servicio de becario o prebecario.
FechaFinRegistro		smalldatetime	AAAAMMDD	Not allowed	Fecha fin en la que se puede solicitar un servicio de becario o prebecario.
FechaInicioServicio		smalldatetime	AAAAMMDD	Not allowed	Fecha a partir de la cual se inicia la vigencia del periodo de servicio.
FechaFinServicio		smalldatetime	AAAAMMDD	Not allowed	Fecha a partir de la cual termina la vigencia del periodo de servicio.
Estatus		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus del Programa de Servicio, que indica si esta activo o no.
IdTipoServicio	FK	tinyint	[0-9]	Not allowed	Clave que identifica a un Tipo de Servicio.

Table: Proyecto

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdProyecto	PK	int identity	[0-9]	Not allowed	Clave que se le asigna a cada proyecto registrado en el sistema.
NombreProyecto		varchar(255)	Todos los valores	Not allowed	Nombre del proyecto.
Descripcion		varchar(255)	Todos los valores	Not allowed	Descripción del proyecto.
Objetivo		varchar(255)	Todos los valores	Not allowed	Objetivo del proyecto.
ProyectoPermanente		bit	0 = Temporal 1 = Permamente	Not allowed	Indica si el proyecto es de forma permanente o si es temporal.
FechaInicioProyecto		smalldatetime	AAAAMMDD	Not allowed	Indica la fecha en la que se inicio el proyecto.
FechaFinProyecto		smalldatetime	AAAAMMDD	Not allowed	Indica la fecha en la que se termina el proyecto.
EstatusProyecto		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si el proyecto esta activo o no.
FechaAltaProy		smalldatetime	AAAAMMDD	Not allowed	Fecha de alta del proyecto en el sistema.
IdResponsable	FK	int	[0-9]	Allowed	Identificador del Responsable a cargo del proyecto.

Table: Puesto

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdPuesto	PK	tinyint identity	[0-9]	Not allowed	Identificador que corresponde a cada uno de los puestos que existen en DGSCA.

NombrePuesto		varchar(150)	Todos los valores	Not allowed	Nombre o descripción que corresponde a cada uno de los puestos de DGSCA.
EstatusPuesto		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si el puesto esta activo o inactivo.

Table: Registro

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdRegistro	PK	int identity	[0-9]	Not allowed	Clave consecutiva para llevar el control de los registros de servicio de un prestador.
IdAlumno	FK	int	[0-9]	Not allowed	Identificador del Alumno al que pertenece el registro.
Promedio		float	[0-9], .	Not allowed	Promedio escolar del prestador, obtenido hasta el momento del registro.
PorcentajeCreditos		float	[0-9], .	Not allowed	Porcentaje de créditos acumulados al momento del registro.
SemestreActual		tinyint	[1-9]	Allowed	Semestre que esta cursando el prestador al momento del registro.
FechaAltaRegistro		smalldatetime	AAAAMMDD	Not allowed	Fecha en la que se registra en el sistema un prestador con un tipo de servicio.
FechaInicioServicio		smalldatetime	AAAAMMDD	Allowed	Fecha en la que inicia el tipo de servicio.
FechaFinServicio		smalldatetime	AAAAMMDD	Allowed	Fecha en la que termina el tipo de servicio.
HorarioInicio		varchar(5)	HH:MM	Allowed	Hora a partir del cuál puede asistir el alumno a DGSCA. Se llena cuando se solicita el servicio por primera vez
HorarioFin		varchar(5)	HH:MM	Allowed	Hora hasta la que puede asistir el alumno a DGSCA. Se llena cuando se solicita el servicio por primera vez.
EstatusRegistro		char(1)	A = Asignado S = Solicitud	Not allowed	Indica si el registro actual es de un alumno que ya fue Asignado a un servicio o bien si realizo una nueva Solicitud.
NombreRespInforme		varchar(255)	Todos los valores	Allowed	Nombre del Responsable Directo que aprobó el Informe de trabajo de un prestador.
NombreRespPlan		varchar(255)	Todos los valores	Allowed	Nombre del Responsable Directo que aprobó el Plan de trabajo de un prestador.

AutorizaRespDirecto		char(1)	I = Aprobó informe P = Aprobó plan	Allowed	Estatus que indica si el Responsable Directo aprobó el Informe y/o el Plan de trabajo.
AutorizaRespArea		char(1)	I = Aprobó informe P = Aprobó plan	Allowed	Estatus que indica si el Responsable de Área aprobó el Informe y /o el Plan de trabajo.
ComentarioDesempeno		text	Todos los valores	Allowed	Comentarios que un Responsable puede hacer respecto al desempeño de un prestador
AutorComentario		varchar(255)	Todos los valores	Allowed	Nombre del Responsable que realiza los comentarios de desempeño del prestador.
ObservacionesBeca		varchar(500)	Todos los valores	Allowed	Observaciones realizadas por el Comité respecto a una solicitud de Beca.
MontoBeca		money	[0-9], .	Allowed	Monto de la beca que fue otorgado al prestador.
EstatusBeca		char(1)	A = Beca otorgada R = Beca rechazada	Allowed	Campo para poner si se le otorgó o no la beca.
Renovacion		bit	0 = No renovación 1 = Renovación	Allowed	Indica si se trata de una renovación de beca para un prestador o no.
Periodo		nvarchar(15)	Todos los valores	Allowed	Clave con la cuál se tiene registrado el Programa de Servicio Social ante la DGOSE, dicha clave se genera anualmente.
ResultadosObtener		varchar(255)	Todos los valores	Allowed	Descripción de los resultados que se esperan obtener en el proyecto con la participación del prestador.
AvanceEsperado		int	[0-9]	Allowed	Porcentaje de avance en la elaboración del proyecto que se espera alcanzar con la participación del prestador.
ResultadosObtenidos		varchar(255)	Todos los valores	Allowed	Descripción de los resultados obtenidos en el proyecto con la participación del prestador.
AvanceLogrado		int	[0-9]	Allowed	Porcentaje de avance en la elaboración del proyecto que se logro con con la participación del prestador.

NombreDireccion		varchar(150)	Todos los valores	Allowed	Nombre de la Dirección de DGSCA a la cuál esta asignado un Alumno.
NombreCentroExtension		varchar(150)	Todos los valores	Allowed	Nombre del Centro de Extensión de DGSCA al cuál esta asignado un Alumno.
NombreUnidadDGSCA		varchar(150)	Todos los valores	Allowed	Nombre de la Unidad Administrativa de DGSCA al cuál esta asignado un Alumno.
IdNivel	FK	smallint	[0-9]	Allowed	Identificador del Nivel de Beca asociado al Registro.
IdProyecto	FK	int	[0-9]	Allowed	Identificador del Proyecto asociado al Registro.
IdProgramaServicio	FK	tinyint	[0-9]	Allowed	Identificador del Programa de Servicio asociado a un Registro.
IdTipoServicio	FK	tinyint	[0-9]	Allowed	Identificador del Tipo de Servicio asociado a un Registro.

Table: Responsable

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdResponsable	PK,FK	int	[0-9]	Not allowed	Clave que identifica a un Responsable, corresponde al identificador de Usuario.
IdResponsableSuperior		int	[0-9]	Allowed	Por medio de esta clave se podrá tener el registro de quién es el Responsable superior de un Responsable.
IdTipoResponsable	FK	int	[0-9]	Allowed	Identificador de Tipo Responsable.
IdPuesto	FK	tinyint	[0-9]	Allowed	Identificador del Puesto asignado al Responsable, de acuerdo a la estructura organizacional de DGSCA.
IdDireccion	FK	tinyint	[0-9]	Allowed	Identificador de la Dirección en la cuál colabora el Responsable.
IdUnidad	FK	tinyint	[0-9]	Allowed	Identificador de la Unidad en la cuál colabora el Responsable.
IdCentro	FK	tinyint	[0-9]	Not allowed	Identificador del Centro de Extensión al que pertenece un Responsable.
EstatusResponsable		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus del Responsable, indica si esta activo o no.

Table: Seccion

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdSeccion	PK	smallint identity	[0-9]	Not allowed	Clave de la sección a la que pertenece cada pregunta, las secciones son: Asistencia, Compromiso, Actitud
NombreSeccion		varchar(150)	Todos los valores	Not allowed	Nombre de la sección mediante las cuáles se clasifican las preguntas
EstatusSeccion		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si la sección esta activa o no.

Table: TipoAlumno

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdTipoAlumno	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica al tipo de alumno.
DescTipoAlumno		varchar(150)	Todos los valores	Not allowed	Descripción del tipo de alumno.
ClaveTipoAlumno		varchar(2)	P = Prestador C = Candidato	Not allowed	Indica si se trata de un alumno que es Prestador o Candidato.
EstatusAlumno		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si un tipo de alumno esta activo o no .

Table: TipoCurso

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdTipoCurso		tinyint identity	[0-9]	Not allowed	Clave que identifica al Tipo de Curso.
DescTipoCurso		varchar(100)	Todos los valores	Not allowed	Descripción del Tipo Curso.
ClaveTipoCurso		varchar(2)	I = Interno E = Externo	Not allowed	Clave alfabética que corresponde al curso.
EstatusTipoCurso		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus que indica si un tipo de curso esta activo o no.

Table: TipoResponsable

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdTipoResponsable	PK	int identity	[0-9]	Not allowed	Clave que identifica a un Tipo de Responsable.
DescTipoResponsable		varchar(150)	Todos los valores	Not allowed	Descripción del Tipo de Responsable
ClaveResponsable		varchar(2)	RS = Responsable de Servicio RA = Responsable de Área RP = Responsable de Prestador	Not allowed	Es una clave alfabética para identificar al Tipo de Responsable
EstatusResponsable		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si un Responsable esta activo o no.

Table: TipoServicio

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdTipoServicio	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica a cada uno de los tipos de servicio registrados en el sistema.
NombreServicio		varchar(50)	Todos los valores	Not allowed	Nombre que describe a cada uno de los servicios que pueden ser solicitados: Becario, Prebecario, Servicio Social y Apoyo.
ClaveServicio		varchar(2)	SA = Servicio Apoyo SS = Servicio Social B = Becario PB = Prebecario	Not allowed	Clave que permite identificar el tipo de servicio mediante una abreviación, por ejemplo servicio social será "SS".
EstatusServicio		bit	0 = Inactivo 1 = Activo	Not allowed	Estatus del tipo de servicio indica si esta activo o no.

Table: UnidadDGSCA

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdUnidad	PK	tinyint identity	[0-9]	Not allowed	Clave que identifica a cada una de las unidades existentes en DGSCA, las cuales pueden ser: subdirección, coordinación, departamento o área
NombreUnidad		varchar(150)	Todos los valores	Not allowed	Nombre de la Unidad de DGSCA
EstatusUnidad		bit	0 = Inactivo 1 = Activo	Not allowed	Indica si la Unidad de DGSCA esta activa o no.

Table: Usuario

Columns	KEY	Data type	Value/Range	Allow NULLs	Description
IdUsuario	PK	int identity	[0-9]	Not allowed	Clave que se le asignará a cada Usuario para tener un registro en el sistema.
DescUsuario		nvarchar(20)	Todos los valores	Not allowed	Nombre o login que utiliza un Usuario para acceso al sistema
Contrasena		nvarchar(10)	Todos los valores	Not allowed	Clave de confirmación de un Usuario para acceso al sistema.
EstatusUsuario		bit	0 = Inactivo 1 = Activo	Not allowed	Bandera que sirve para identificar si el usuario está activo o no.
NombreUsuario		varchar(55)	Todos los valores	Not allowed	Nombre completo del Usuario.
ApePaternoUsuario		varchar(55)	Todos los valores	Not allowed	Apellido Paterno del Usuario

ApeMaternoUsuario		varchar(55)	Todos los valores	Not allowed	Apellido Materno del Usuario.
EmailUsuario		varchar(50)	Todos los valores	Not allowed	Email del Usuario.
FechaRegistro		smalldatetime	AAAAMMDD	Not allowed	Fecha de Registro de un Usuario en el sistema.
IdPerfil	FK	tinyint	[0-9]	Not allowed	Identificador de perfil de usuario

9. Glosario

Abstracción	Es el proceso de extraer las características esenciales o generales de alguna cosa u objeto.
Actor	En UML, es un rol que lleva a cabo una persona, que es usuaria de un sistema o bien otro sistema.
Applets	Subprograma invocado por un navegador Web desde una página en internet, comúnmente muestra gráficos, animaciones, emite sonidos.
Artefacto	Es algún documento, informe o ejecutable que se produce o manipula, durante la metodología de desarrollo de software.
Base de Datos	Conjunto de información almacenada y organizada sistemáticamente, la cual pertenece a un mismo contexto.
Desviación estándar	La desviación estándar es una medida del grado de dispersión de los datos con respecto al valor promedio, es el "promedio" o variación esperada con respecto a la media aritmética.
Framework	Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.
HTTP	Protocolo de transferencia de hipertexto. El propósito del protocolo HTTP es permitir la transferencia de archivos (principalmente, en formato HTML) entre un navegador (el cliente) y un servidor web localizado mediante una cadena de caracteres denominada dirección URL.
Independencia física	En bases de datos, significa que la información no depende del medio de almacenamiento.
Independencia lógica	En bases de datos, hace referencia a que cuando se actualiza un dato, el resto de la información almacenada no debe sufrir alteraciones por dicho cambio.
Normalización	Proceso de descomposición si pérdida de información para obtener nuevas relaciones que satisfagan las formas normales, evitando problemas de actualización y redundancia.
Objeto	Representa un elemento identificable que está constituido por ciertas características o atributos y es capaz de realizar un conjunto de acciones u operaciones.
Paradigma	Modelo o ejemplo a seguir, por una comunidad científica, es un ejemplo de los problemas que tiene que resolver y del modo como se van a dar las soluciones.
Patrón	En el desarrollo de software, es la descripción de un problema y de la solución a dicho problema, la cual ha sido implementada y probada anteriormente.

PERT	Siglas de la técnica PERT que significan Program Evaluation and Review Technique. Dicha técnica es utilizada para evaluar los tiempos que se deberán considerar para cada tarea en un proyecto.
Riesgo	Evento incierto que podría afectar de manera negativa a los factores críticos de éxito, si llega a ocurrir.
RUP	Rational Unified Process. Proceso de desarrollo de software.
Severidad del riesgo	Es el resultado de multiplicar el impacto por la probabilidad.
Smalltalk	Es considerado como el primer lenguaje orientado a objetos.
Stakeholder	Es cualquier involucrado dentro del ámbito de un proyecto.
Struts	Herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC. Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation.
TCP/IP	Siglas de Protocolo de Control de Transmisión/Protocolo de Internet (en inglés Transmission Control Protocol/Internet Protocol).
Thread	Un hilo o flujo de control del programa. Representa un proceso individual ejecutándose en un sistema.

10. Bibliografía y Referencias

Bibliografía

- Cavaness, Chuck; et.al. (2003). *Jakarta Struts Pocket Reference*. Surrey: O'reilly & Associates.
- De Miguel, Adoración; et.al. (1993). *Concepción y diseño de bases de datos, del modelo E/R al modelo relacional*. USA: Addison-Wesley Iberoameriacana.
- Horine, Gregory M. (2005). *Gestión de Proyectos*. Madrid: Anaya Multimedia.
- Husted, Ted. (2003). *Struts In Action: A Practical Guide To The Leading Java Web Frame Work*. London: Manning Publications CO.
- Larman, Criag. (2003). *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Madrid: Pearson Education.
- Schach, Stephen. (2004). *Análisis y Diseño OO con UML y el proceso unificado*. MacGraw-Hill.
- Schmuller, Joseph. (2000). *Aprendiendo UML en 24 horas*. México: Prentice – Hall.
- Silberschatz, Abraham; et.al. (2007). *Fundamentos de Diseño de Bases de Datos*. Madrid: McGraw-Hill/Interamericana.
- Stelting, Stephen. (2003). *Patrones de Diseño aplicados a JAVA*. Madrid: Pearson Education.

Referencias Electrónicas

- IBM Corporation. (2000, 2006). *Rational Method Composer*. Version 7.1. http://www.ibm.com/developerworks/downloads/r/rup/ Fecha de consulta: Agosto 2007-Febrero 2009
- The Apache Software Foundation. (2000-2009). *Struts*. http://struts.apache.org/. Fecha de consulta: Agosto 2007-Febrero 2009
- Sun Microsystems, Inc. (1994-2009). *Java SE APIs & Documentation*. http://java.sun.com/javase/. Fecha de consulta: Agosto 2007-Febrero 2009