UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

CENTRO DE INGENIERÍA DE SOFTWARE Y SISTEMAS

Desarrollo de una Herramienta de Software Libre que Permita el Diseño de Centros de Datos

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por el

Br. Ronald, D. Arias M.
CI 18.841.034

Para optar por el título de
Licenciado en Computación

Tutor: Prof. Andrés Castro

13 de Octubre de 2010

DEDICATORIA

Quiero dedicar este Trabajo Especial de Grado a las dos mujeres que más amo en este mundo, mi abuela Marcia y mi madre Haydeé quienes han sacrificado mucho para que sea quien soy ahora y a mi padre Antonio por estar cuando lo he necesitado.

Una vida entera no será suficiente para retribuirles.

Los Amo.

AGRADECIMIENTOS

En principio agradecer a Dios Todopoderoso, que me ha brindado fuerza y esperanza a lo largo del camino y que siempre ha guiado mis pasos hacia el mejor sendero.

A mi novia Annalicia, quien me ha dado consejos oportunos y ha sabido darme esas palabras de aliento que se necesitan para seguir adelante.

A mis Tios Yoel, Jose Luis y Wilfredo, quienes han sido padres para mí, siempre enseñándome la diferencia entre el bien y el mal.

A mi Abuelo Jose, quien a pesar de testarudo, ha sabido brindarme palabras sabias en momentos de oscuridad.

A mi pequeño Hermano Alexander, con quien siempre peleo solo por el hecho de que nos queremos y con quien seguiré retribuyéndole a la familia todo lo que nos han dado.

A mi mejor amigo de toda la vida y hermano del alma Fernando, 18 años de amistad y los que nos faltan amigo.

A mis hermanos Junior y Michelle, quienes a pesar que no los veo mucho, se que están ahí y de una u otra forma me han ayudado a crecer como persona.

A mi madrina Fátima, que siempre ha estado para ayudar y hacerme un hombre de bien. A Carmen Mata que trae siempre alegría y optimismo a mi persona.

A mi tutor Andrés Castro, por confiar en mi talento y ayudarme a materializar mis ideas desde que fui su alumno hasta este momento.

Al profesor Andrés Sanoja, quien ha sido un gran mentor, del que he aprendido innumerables cosas y con quien he compartido más que lo académico, una amistad.

A la profesora Jossie Zambrano, por siempre estar al tanto de mis cosas y abrirme tantas puertas que me permitieron alcanzar muchos éxitos.

Al profesor Trino Gómez por guiarme en el camino de los maratones y darme ánimos hasta el último momento de las competencias.

A mi grupo más apreciado de amigos, Gilberto "Tico" Ovalles, Gino Di Paolo, Nevily "La Negra" Aular, Francisco Suarez, Antonio "Toño" Dos Santos, Sergio Escalante, Dayana Torres, María "Maga" De Freitas, Enrique "kike" Buono, Alexandra Arreaza y todos aquellos que en este momento escapan de mi mente. Cada uno aportó un poco para este logro.

A los profesores que contribuyeron a mi formación tanto académica como personal Antonio Silva, Robinson Rivas, Eugenio Scalise, Marcos Raydan, Luis Manuel Hernandez, Jesus Lares, Guy Bernaez, Ronald Pietri, Rafael Angulo, Antonio Leal, Sergio Rivas, Edgar Gonzalez, Tina Di Vasta, Claudia León, Ernesto Coto, Eleonora Acosta y Esmitt Ramirez.

A Todas aquellas personas que no han sido nombradas pero que de alguna manera han estado presentes a lo largo de mi vida. Gracias.

RESUMEN

El presente Trabajo Especial de Grado consiste en el análisis, diseño *y desarrollo de una herramienta de software libre que permita el diseño de centros de datos, para así poder llevar un inventario de los equipos, reflejar las interconexiones de los mismos y visualizar las aplicaciones que en éstos se ejecutan.*

Normalmente en un centro de datos no se tiene un inventario actualizado de los recursos de hardware y software, es decir, que la mayoría de los centros de datos operan sin llevar un control específico de los componentes, perfil del uso de los equipos, estado de uso y obsolencia en que se encuentran y aplicaciones que se encuentran en las instalaciones. Esto acarrea que la solicitud de nuevos equipos, mantenimiento y producción se vean limitados.

El objetivo del presente Trabajo Especial de Grado resolver los problemas antes mencionados y optimizar las tareas de los operadores y administradores de centros de datos, y por ende el rendimiento del mismo.

El sistema desarrollado proporciona interfaces de diseño y carga de datos que permiten a los usuarios incorporar al sistema toda la información relacionada con el centro de datos a diseñar, así como el posicionamiento e interconexiones de los elementos e igualmente el software que en ellos se ejecuta. De la misma manera, el sistema proporciona interfaces que permiten al usuario visualizar el centro de datos una vez diseñado, o incluso parcialmente diseñado. Una vez que se guarda un estado del centro de datos en el diseñador, inmediatamente se encuentra disponible para su visualización.

Las tecnologías y herramientas utilizadas más relevantes que contribuyeron al desarrollo exitoso de la aplicación fueron las siguientes: Ruby on Rails, framework para el desarrollo de aplicaciones Web bajo el

esquema MVC; ActionScript 3, lenguaje de programación de Adobe que cuenta con una amplia librería de funciones gráficas; Action Message Format 3, formato de intercambio de mensajes que permite una comunicación eficiente entre un servidor de aplicaciones y ActionScript; AJAX, enfoque de desarrollo basado en un conjunto de tecnologías ya existentes, agrupadas para presentar información e interactuar dinámicamente, de manera asíncrona, con un servidor Web; Prototype, librería de JavaScript que provee de múltiples funcionalidades y utilidades tales como interacción con AJAX, efectos gráficos, entre otros; Swift3D, suite de diseño de elementos 3D con capacidades de exportación al formato SWF de Flash; Suite Adobe Flash CS4, entorno de desarrollo que provee múltiples herramientas que facilitan e incrementan el tiempo de desarrollo; Distributed ruby, tecnología de comunicaciones distribuidas para el lenguaje ruby.

Términos Clave: *Centro de Datos, Interfaz Enriquecida o entorno RIA, Framework de desarrollo, Servidor de aplicaciones, Drag and Drop, REST, DRB, Sensors, Base de datos.*

Índice General

Índice de Figuras

INTRODUCCIÓN

El propósito fundamental del presente Trabajo Especial de Grado, es mostrar las habilidades y capacidades adquiridas a lo largo de la carrera, así como el empleo de los conocimientos teóricos y del instrumental tecnológico y metodológico.

El rápido y continuo crecimiento de Internet y con ello, el número de usuarios conectados a éste, así como la gran cantidad de redes implementadas a nivel mundial, ha sugerido la necesidad de desarrollar herramientas que faciliten los trabajos de administración de redes, creación y administración de centros de datos al igual que nuevas y mejores alternativas de almacenamiento y procesamiento de datos, poniendo a su alcance sistemas informáticos que permitan el control y el monitoreo de dichas actividades; detectando eventos anómalos, prevención de posibles fallas y optimización de procesos como auditorías de seguridad y monitoreo.

Sin embargo, la mayoría de los centros de datos operan sin llevar un control específico de los componentes y aplicaciones que se encuentran en las instalaciones. Esto acarrea que la solicitud de nuevos equipos, mantenimiento y producción se vean limitados. Adicionalmente, no se tiene un perfil del uso de los equipos, por lo que se hace difícil saber el estado de uso y obsolencia en que se encuentran.

En base a dichos problemas, surge la necesidad de contar con un sistema que automatice dichos procesos de forma rápida, eficiente y fácil de usar. Es decir, un software que contenga los planos de todo el centro de datos y además permita actualizarlos rápida y fácilmente, para así generar gran facilidad en la administración y optimización del establecimiento.

Debido a esto, el objetivo que motiva la realización del presente Trabajo Especial de Grado, será por tanto, d*iseñar y desarrollar una herramienta de software libre que permita el diseño de centros de datos, para así poder llevar un inventario de los equipos, reflejar las interconexiones de los mismos y visualizar las aplicaciones que en éstos se ejecutan.*

Asimismo, el alcance del presente documento será la elaboración de una herramienta de software que contenga los planos de un centro de datos, permita modificarlos y administrarlos, y además de contar con el monitoreo de los recursos que en este se encuentran.

De acuerdo a las necesidades descritas previamente y para concretar lo antes planteado, se dará una visión general de cada uno de los capítulos que estructuran el presente trabajo Especial de Grado.

a) Capítulo 1: Problema de la Investigación.

Se plantea la necesidad de d*iseñar y desarrollar una herramienta de software libre que permita el diseño de centros de datos, para así poder llevar un inventario de los equipos, reflejar las interconexiones de los mismos y visualizar las aplicaciones que en estos se ejecutan. De igual forma se plantea el contexto del problema y la solución propuesta para el mismo, el objetivo general y los objetivos específicos del presente Trabajo Especial de Grado, así como la justificación e importancia de su realización. Por último, se describe la metodología de desarrollo aplicada.*

b) Capítulo 2: Marco Conceptual.

Se describen los tópicos más relevantes que se encuentran estrechamente relacionados con el problema de investigación, abarcando los siguientes aspectos:

- *Centros de datos, definición y descripción de los mismos, componentes que lo conforman y tipos más comunes.*
- *Interfaces enriquecidas, definición y características de las mismas. Entorno RIA, definición y características, descripción y principales tecnologías.*
- *Tecnologías de desarrollo, definición, funcionamiento y descripción de cada una de las siguientes: Ruby on Rails, ActionScript 3, Action Message Format 3, Asynchronous JavaScript And Xml, Prototype y Swift 3D.*

c) Capítulo 3: Marco Aplicativo.

Se describe de manera detallada cada uno de los pasos de la metodología de desarrollo, aplicadas al caso de estudio en una adaptación de UP ágil denominado "Desarrollo de una herramienta de software libre que permita el diseño de centros de datos". Dicha adaptación de la metodología consta de tres fases: Fase de Iniciación, Fase de Elaboración y Fase de Construcción las cuales son aplicadas en tres iteraciones sobre el desarrollo del Trabajo Especial de Grado.

Luego se presentan las conclusiones, consideraciones y recomendaciones del presente Trabajo Especial de Grado.

Finalmente, se presentan las referencias bibliográficas y digitales.

CAPÍTULO I

PROBLEMA DE LA INVESTIGACIÓN

En el siguiente capítulo se plantea el escenario sobre el cual se desarrolla el presente Trabajo Especial de Grado.

1.1. Problema de investigación

Normalmente en un centro de datos no se tiene un inventario actualizado de los recursos de hardware y software, es decir, que la mayoría de los centros de datos operan sin llevar un control específico de los componentes y aplicaciones que se encuentran en las instalaciones. Esto acarrea que la solicitud de nuevos equipos, mantenimiento y producción se vean limitados.

Otro de los problemas que existen es que no se tiene un perfil del uso de los equipos, por lo que se hace difícil saber el estado de uso y obsolencia en que se encuentran. Conociendo esta información se facilita el reemplazo de los equipos, la utilidad que puede tener algún equipo y así mismo sacar el mayor provecho del mismo.

Por último, la falta de conocimiento de las aplicaciones que se encuentran ejecutándose en los servidores ralentiza y dificulta los diagnósticos y auditorias de seguridad sobre centro de datos.

1.2. Solución

De acuerdo al problema planteado, la solución propuesta es implementar un software óptimo y eficiente que permita el diseño de centros de datos, para así poder llevar un inventario de los equipos, reflejar las

interconexiones de los mismos y visualizar las aplicaciones que en éstos se ejecutan.

1.3. Objetivo General

D*iseñar y desarrollar una herramienta de software libre que permita el diseño de centros de datos, para así poder llevar un inventario de los equipos, reflejar las interconexiones de los mismos y visualizar las aplicaciones que en éstos se ejecutan.*

1.4. Objetivos específicos

1. *Realizar un estudio de las operaciones comunes de los centros de datos.*
2. *Planificar, diseñar e implementar un sistema de carga de datos, un sistema de diseño y un sistema de visualización de los componentes del centro de datos.*
3. *Implementar el mínimo de funcionalidades básicas que provean a la aplicación de similitudes con aplicaciones de uso común. Este mínimo sería:*
   - *Editar propiedades de los componentes.*
   - *Arrastrar elementos de una biblioteca al sitio de diseño.*
   - *Personalización de dichos elementos.*
4. *Analizar la posibilidad de incluir un sistema de monitoreo de temperatura mediante hardware especializado que se conecte a la aplicación.*
5. *Realizar mediciones de la usabilidad y accesibilidad de la aplicación.*
6. *Analizar la extensibilidad y escalabilidad de la solución implementada, para permitir su implantación en múltiples centros de datos.*

1.5. Justificación e importancia

Debido a la importancia que tiene para un administrador de centros de datos el mantenimiento de las instalaciones de hardware y software, así como la facilitación de sus procesos y la posible optimización de sus equipos, surge la necesidad de contar con un sistema que automatice dichos procesos de forma rápida, eficiente y fácil de usar.

Contar con un software que contenga los planos de todo el centro de datos y además permita actualizarlos rápida y fácilmente generará gran facilidad para la administración y optimización del establecimiento.

1.6. Metodología de trabajo

Para el desarrollo del presente Trabajo Especial de Grado se utiliza una variación de la metodología Agile Unified Process (UP ágil).

Según Bryan Campbell, Agile UP es una versión simplificada de Rational Unified Process (RUP). Este describe un enfoque simple y fácil de entender para el desarrollo de software usando técnicas y conceptos que aún se mantienen vigentes en RUP. Los enfoques aplican técnicas ágiles incluidas en el Desarrollo Dirigido por Pruebas (TDD), Desarrollo Dirigido por Modelado Ágil (AMDD), administración de cambios ágil, y refactorización de bases de datos para mejorar la productividad.

La Figura 1.1 representa el ciclo de vida de UP ágil.

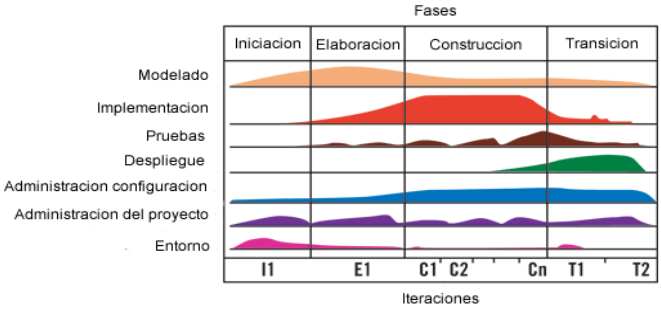

Figura 1.1 Actividades en UP ágil

En dicha figura la naturaleza serial de UP Ágil es capturada en cuatro fases:

a) *Iniciación: El objetivo es identificar el alcance inicial del proyecto, una arquitectura potencial de su sistema, y obtener la financiación inicial del proyecto y la aceptación del involucrado.*
b) *Elaboración: El objetivo es mejorar la arquitectura del sistema.*
c) *Construcción: El objetivo es construir software funcional en una base regular e incremental, la cual cumpla con las necesidades de prioridad más alta de los involucrados del proyecto.*
d) *Transición: El objetivo es validar y desplegar el sistema en el ambiente de producción.*

Por otro lado, las disciplinas son ejecutadas en una manera iterativa, definiendo las actividades, las cuales los miembros del equipo ejecutan para construir, validar y liberar software funcional que cumpla con las necesidades.

La metodología ha sido modificada para adaptarse a las necesidades de desarrollo de este Trabajo Especial de Grado. En principio se utilizarán solo las fases de iniciación, elaboración y construcción ya que no está establecido como objetivo la puesta en producción de la aplicación (Fase de transición). Asimismo las disciplinas a utilizar serán las de modelado, implementación, pruebas y administración del proyecto.

De esta manera el proceso está organizado en cinco iteraciones, la primera para el desarrollo del sistema cargador de datos, la segunda iteración cubrirá el sistema diseñador, la tercera la resolución de problemas y refinamiento del cargador de datos y diseñador, la cuarta iteración abarcará el sistema visualizador y refinamiento de las 2 primeras iteraciones, además de las pruebas necesarias para garantizar su correcto funcionamiento y la quinta y última abarcará el sistema de monitoreo del centro de datos.

A continuación se explica detalladamente cada una de las fases de desarrollo de la metodología propuesta que serán aplicadas a cada iteración definida. Es importante señalar que las iteraciones bien definidas y los artefactos que se desean obtener se encuentran detalladas en el marco aplicativo.

a) Fase de iniciación

Realizar el levantamiento de información. Analizar los requerimientos de la iteración y los artefactos que deben ser generados.

b) Fase de elaboración

El propósito de esta fase incursionar en el dominio del problema y realizar el diseño de la solución, además de la planificación de la generación de los artefactos de la iteración.

c) Fase de construcción

Esta fase consiste en la implementación de los componentes de la aplicación de la iteración actual. Asimismo la realización de los artefactos y demás diagramas necesarios para la realización de la aplicación.

CAPITULO II

MARCO CONCEPTUAL

En este capítulo se describen los tópicos que están estrechamente relacionados con los componentes de centros de datos. Asimismo se puntualizan los aspectos más relevantes las interfaces enriquecidas y todo lo referente al desarrollo de aplicaciones enriquecidas de internet (RIA) y finalmente se describen el conjunto de tecnologías utilizadas para el desarrollo del presente Trabajo Especial de Grado.

2.1. Centros de Datos

Los centros de datos han existido desde los inicios de la computación, desde aquellas estructuras gigantes difíciles de operar y mantener hasta los actuales que requieren de poco espacio y proveen gran cantidad de herramientas para su administración. Sus principales objetivos siempre han sido ejecutar aplicaciones para administrar lógica de negocio, cálculos para investigaciones académicas y almacenar grandes volúmenes de datos.

Básicamente un centro de datos, también conocido como centro de cómputo o centro de cálculo, según [U-DC-WK] es un edificio o sala de mediano o gran tamaño usada para mantener en ella una gran cantidad de equipo de procesamiento, interconexión y almacenamiento de datos. Usualmente estos almacenan gran cantidad de computadoras con altas capacidades de procesamiento y almacenamiento, conectadas mediante dispositivos de red de alta velocidad.

Los centros de datos son ambientes críticos para cualquier compañía y la necesidad de su correcto diseño y administración es un parámetro

sumamente importante a considerar. Igualmente parámetros como la seguridad de los equipos e integridad de los datos que este almacena.

En resumen, según [U-CPD-WK], entre las principales funciones que puede cumplir un centro de datos destacan: (a) Almacenar, procesar e intercambiar información digital (b) Proveer aplicaciones y servicios, administrando varios servicios tales como alojamiento web, intranet, servidores de correos, servidores de aplicaciones, etc. La razón de esto radica en que usualmente son creados y mantenidos por grandes compañías tales como bancos y empresas financieras, universidades, entidades gubernamentales y otras empresas que manejan o procesan grandes volúmenes de datos y todas estas con objetivos en común entre los cuales destacan el acceso y procesamiento rápido sobre los datos y su disponibilidad continua.

2.1.1. Componentes

Los centros de datos constituyen una parte crítica en cualquier organización, y con ésta surgen necesidades fundamentales de comunicación y cooperación de todos los factores para su correcto diseño, funcionamiento y administración.

En este sentido, los componentes de un centro de datos constituyen su parte física y su parte lógica, ya que al hablar de componentes se hace referencia no sólo a la infraestructura física sino también a las interconexiones y a la infraestructura tecnológica y de aplicaciones, las cuales representan la lógica de negocio de la organización. Así mismo, según [U-DC-EF], en la manera de organizar dichos componentes resaltan tres categorías importantes: (a) Componentes físicos (b) Componentes de Red y (c) Aplicaciones.

Los componentes físicos son todos aquellos elementos eléctricos, electrónicos, mecánicos y electromecánicos que integran la infraestructura

del centro de datos; sus cables, gabinetes o cajas, torres, periféricos de todo tipo y cualquier otro elemento físico involucrado; contrariamente al software el cual es un soporte lógico e intangible.

Como describe [U-RED-ML], Los componentes de red, a pesar de que son compontes físicos también, se diferencian del hardware común porque cumplen una función muy particular, la cual es interconectar dichos elementos de hardware común. No obstante, interconectar componentes de hardware no es sino la función principal de estos componentes, además de esta, cumplen con muchas más funciones vitales para el desarrollo de un centro de datos.

Por último Las aplicaciones son todo el componente lógico de un centro de datos, es decir, estas llegan a ser literalmente "el cerebro" de la instalación. Los antes mencionados componentes físicos y de red constituyen el caparazón o estructura del centro de datos, en cambio las aplicaciones son la lógica del centro, aquellas funciones para las cuales existen los componentes físicos.

De acuerdo con esta categorización podríamos resumir que el hardware de un centro de datos se encuentra en componentes físicos, las conexiones entre dicho hardware y otras redes en componentes de red y la lógica de negocios y servicios en la parte de aplicaciones.

2.2. Interfaces Enriquecidas

La tendencia a desarrollar aplicaciones en Internet viene creciendo a un ritmo muy acelerado y la mayoría de las organizaciones están empezando a aprovechar los beneficios que ésta brinda [Duhl, 2003].

A menudo, la falta de atención en asuntos relacionados con la navegación de un sitio resulta en la frustración del usuario, ya que no puede encontrar lo que está buscando. Y con esta experiencia, se reduce la posibilidad de convertir a dicho usuario en cliente.

En general, para construir un buen sitio es necesario comprender las necesidades de su audiencia. Y hay que tener en cuenta dos aspectos fundamentales: la usabilidad y la accesibilidad. La usabilidad, para un sitio web, mide la facilidad con la que un usuario realiza determinadas acciones en él, a través de algunos parámetros como navegación intuitiva, disponibilidad de la información, posibilidad de memorizar opciones seleccionadas, eficiencia del sitio, y satisfacción en general.

Hasta ahora, las aplicaciones de escritorio permitían una mayor riqueza gráfica y mejor respuesta en la interacción con el usuario en comparación con las aplicaciones Web. Se pensaba que las aplicaciones Web nunca alcanzarían la capacidad de interacción de las aplicaciones de escritorio. En efecto, ante cada acción del usuario que requería al servidor, se debía recargar la página Web utilizada o dirigirse a una nueva, perjudicando el tiempo de respuesta. Según Bradbury (2006) [P-RIA-DB] esto conduce a usuarios insatisfechos, lo cual redunda en pérdida de ingresos para las organizaciones.

Esta brecha de desempeño se está cerrando y cada vez más las aplicaciones Web se acercan a las aplicaciones de escritorio; por ejemplo, se puede citar a Google Suggest [Google, 2008b], Google Maps [Google, 2008a] y Flickr [Yahoo, 2008] entre los casos más conocidos.

El uso de Internet está cambiando junto con las expectativas de los usuarios. La proliferación del ancho de banda, la demanda de los consumidores y la competencia de las organizaciones por llegar a nuevos mercados, crea la necesidad de impulsar nuevas tecnologías de desarrollo de aplicaciones que tengan el alcance de las aplicaciones Web, pero que tengan además el desempeño y la calidad de interacción de las aplicaciones de escritorio.

Según [U-RIA-SID], las interfaces o aplicaciones enriquecidas de internet se encuentran a la vanguardia de este cambio, ofreciendo una serie

de ventajas y tecnologías que prometen mejorar el desempeño y economía de las empresas, así como facilitar y mejorar la interacción con los usuarios. El objetivo de estas es ofrecer un conjunto de tecnologías sobre Web que permitan crear aplicaciones tan rápidas, llamativas y eficientes como las aplicaciones de escritorio, pero aprovechando además todas las ventajas de una aplicación Web.

2.2.1. Entorno RIA

Rich Internet Applications (RIAs) o aplicaciones de internet enriquecidas, básicamente se trata de un nuevo paradigma de desarrollo de aplicaciones web que está emergiendo actualmente con mucha fuerza en el mundo de las Tecnologías de la Información y los negocios. La mejor manera de comprender lo que son las RIAs es poniéndolas en el contexto de otras tecnologías. Para esto debemos pensar las soluciones tecnológicas en términos de dos características: alcance y riqueza.

Riqueza es la habilidad para incorporar interactividad e interfaces de usuario intuitivas en el cliente, y alcance es la habilidad de la aplicación para estar disponible para cualquier usuario en el lugar del planeta en el que éste se encuentre.

Así mismo, una aplicación RIA es un tipo de aplicación Web que provee una mayor riqueza interactiva que las aplicaciones Web tradicionales, incorporando características muy similares a las que poseen las aplicaciones de escritorio, las cuales proveen una interacción dinámica y rica en elementos de interfaz.

Como se mencionó anteriormente, el modelo tradicional de aplicaciones Web tiene una serie de desventajas, como la poca capacidad multimedia que posee y la recarga continua de páginas. Todo esto se debe a que el cliente en las aplicaciones Web tradicionales solo se limita a desplegar el contenido HTML. En cambio, las RIAs incorporan un motor como una

nueva capa del lado del cliente que sirve como intermediaria entre la interacción del cliente con el servidor y tiene la responsabilidad de realizar los cambios sobre la interfaz de usuario. En síntesis, las RIAs plantean un nuevo enfoque que logrará muchas contribuciones al entorno Web y con ello una mejora de la interacción entre los internautas y las aplicaciones.

El enfoque del presente Trabajo Especial de Grado será el desarrollo de una aplicación RIA donde se aproveche todo el potencial y beneficios de las mismas.

2.3. Tecnologías de Desarrollo

En el presente Trabajo Especial de Grado, con la finalidad de construir una aplicación con tecnologías emergentes en el mundo Web y en el enfoque de desarrollo RIA, se utilizaron gran cantidad tecnologías complejas que garantizaran la calidad y eficiencia de la aplicación. A continuación se describen cada una de ellas.

2.3.1. Ruby on Rails

Según Ruby on Rails Org [U-RoR] Ruby on Rails es un entorno de desarrollo web de código abierto que está optimizado para satisfacción de los programadores y de la productividad. Te permite escribir un buen código favoreciendo la convención antes que la configuración.

En este concepto es importante definir la separación entre los dos entes importantes que conforman este entorno, ruby y rails. Por su parte, según informit [U-INFIT] Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre. Según su creador, Ruby está diseñado para la productividad y la diversión del desarrollador, siguiendo los

principios de una buena interfaz de usuario. Sostiene que el diseño de sistemas necesita enfatizar las necesidades humanas más que las de la máquina.

Por otro lado, Rails, según Ruby on Rails Org, es un completo entorno para desarrollar aplicaciones web con base de datos de acuerdo con la estructura Modelo-Vista-Controlador (MVC). Desde el Ajax en la vista, a la petición y respuesta en el controlador, hasta el modelo, Rails te da un entorno de desarrollo de Ruby. Para probarlo, solo se necesita una base de datos y un servidor web.

El desarrollo sobre este entorno está basado en dos filosofías, "no te repitas" (del inglés Don't repeat yourself, DRY) y "convención sobre configuración". DRY significa que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework de pila completa, los componentes están integrados de manera que no hace falta establecer puentes entre ellos. Mientras que convención sobre configuración significa que el programador sólo necesita definir aquella configuración que no es convencional. Ruby on Rails se ha convertido en un entorno muy poderoso para el desarrollo de aplicaciones Web y que cada día toma más auge dentro del mundo del desarrollo Web.

Ruby on Rails funciona bajo el paradigma Modelo-Vista-Controlador (MVC), [Steve Burbeck; 1992] el cual es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, el cual aplicado a web la vista es un página HTML, o html.erb en caso de rails, el cual contiene el HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Asimismo Rails permite la integración con aplicaciones de desarrolladores externos llamadas plugins, las cuales proveen funcionalidades extra a las que provee dicho framework y que contribuyen de manera notable al desarrollo, incrementando las funcionalidades y disminuyendo los tiempos.

Los plugins utilizados en el presente Trabajo Especial de Grado se nombran a continuación:

- *SWF_FU: Es un plugin que se encarga de lidiar con los archivos SWF de animación de flash proveyendo al desarrollador poder manipular los SWF de la misma forma que utiliza las imágenes, hojas de estilo y demás archivos públicos en rails. SWF_FU utiliza SWFObject (Librería oficial de JavaScript para la inclusión de archivos SWF en documentos HTML) para embeber los objetos de flash en las vistas y soportar todas las opciones del mismo.*
- *RubyAMF: Es un plugin que funciona como Gateway remoto para flash en rails. Integra directamente dentro de un controlador las capacidades para resolver serializaciones AMF y establecer comunicaciones del mismo modo.*
- *PDF Writer: Este plugin provee una forma rápida y fácil de usar para la creación de documentos PDF. Provee una interfaz para la inserción de elementos mediante coordenadas, incluyendo entre ellos desde texto simple hasta imágenes y gráficos.*

2.3.2. Action Script 3.0

Según Adobe Systems incorporated [A-Adobe-as3], ActionScript(AS) es un lenguaje de programación orientado a objetos (OOP), utilizado en especial en aplicaciones web animadas realizadas en el entorno Adobe Flash, FLEX o AIR, así como en entornos de aplicaciones RIA. AS fue lanzado con la versión 4 de Flash, y desde entonces, ha ido ampliándose

poco a poco, hasta llegar a niveles de dinamismo y versatilidad muy altos en la versión 10 (Adobe Flash CS4) de Flash.

ActionScript es un lenguaje de script y está basado en especificaciones de estándar de industria ECMA-262, un estándar para JavaScript, de ahí que ActionScript tenga similitud en su sintaxis con JavaScript.

La versión más extendida actualmente es ActionScript 3.0, que significo una mejora en el manejo de programación orientada a objetos al ajustarse mejor al estándar ECMA-262 y es utilizada en la última versión de Adobe Flash y Flex. Desde la versión 2 de Flex viene incluido ActionScript 3, el cual mejora su rendimiento en comparación de sus antecesores, además de incluir nuevas características como el uso de expresiones regulares y nuevas formas de empaquetar las clases.

Según [U-AS3-TC], ActionScript 3.0 ofrece un modelo de programación robusto que resultará familiar a los desarrolladores con conocimientos básicos sobre programación orientada a objetos. Algunas de las principales funciones de ActionScript 3.0 son:

a) *Una nueva máquina virtual ActionScript, denominada AVM, que utiliza un nuevo conjunto de instrucciones de código de bytes y proporciona importantes mejoras de rendimiento.*
b) *Una base de código de compilador más moderna, que se ajusta mejor al estándar ECMAScript (ECMA 262) y que realiza mejores optimizaciones que las versiones anteriores del compilador.*
c) *Una interfaz de programación de aplicaciones (API) ampliada y mejorada, con un control de bajo nivel de los objetos y un auténtico modelo orientado a objetos.*
d) *Un núcleo del lenguaje basado en el próximo borrador de especificación del lenguaje ECMAScript (ECMA-262) edición 4.*

e) *Una API XML basada en la especificación de ECMAScript para XML (E4X) (ECMA-357 edición 2). E4X es una extensión del lenguaje ECMAScript que añade XML como un tipo de datos nativo del lenguaje.*
f) *Un modelo de eventos basado en la especificación de eventos DOM (modelo de objetos de documento) de nivel 3.*

2.3.3. Action Message Format 3

De acuerdo con la especificación para AMF 3 [P-AMF-AD], Adobe Systems Incorporated lo define como un formato binario compacto que es usado para serializar objetos de ActionScript. Una vez serializado el objeto puede ser usado para persistir y obtener el estado público de una aplicación a través de una sesión o para permitir la comunicación entre dos puntos a través de una red para intercambiar datos estructurados. Asimismo, AMF es internamente usado por el flash player en muchas situaciones como por ejemplo para representar datos binarios almacenados usando la clase ByteArray.

AMF, en el ámbito de este Trabajo Especial de Grado, propone una mejora en las comunicaciones entre el servidor de aplicaciones y el cliente que se ejecuta en el flash player hecho en ActionScript. Las comunicaciones fueron optimizadas para permitir sustituir grandes documentos en formato XML o JSON al formato binario que propone AMF. En consecuencia, los datos en la transmisión cliente servidor son más ligeros y pueden viajar más rápido. Además se agrega un nivel de seguridad a las comunicaciones ya que se cambia el texto plano a datos binarios, lo cual implica que cualquier interceptor de las comunicaciones debe además interpretar los datos binarios para poder visualizarlos.

Para el uso de AMF se necesita un decodificador en el lado del servidor que pueda interpretar los tipos de datos enviados y transformarlos

al lenguaje nativo. Por ser el contexto de esta aplicación basado en el lenguaje del lado del servidor ruby, se utiliza rubyAMF como conversor y el mapeo de los tipos de datos desde ActionScript a Ruby es el siguiente:

- *undefined -> nil*
- *null -> nil*
- *false -> false*
- *true -> true*
- *Number -> Fixnum*
- *int -> Integer*
- *String -> String*
- *XML -> String (serializado listo para usar la función "from_xml")*
- *Array -> Array*
- *MixexArray -> Hash*
- *Object -> Hash*
- *Custom Class -> Ruby Class*

La principal utilidad que tiene AMF, como ya se ha mencionado además de serializar tipos de datos en ActionScript, es que puede ser usado para comunicaciones asíncronas en invocaciones de servicios remotos. Una estructura de mensajes simple es usada para enviar un conjunto de solicitudes a un punto remoto. De esta forma el cliente en ActionScript se convierte en un consumidor de servicios remotos, aprovechando además las características asíncronas de la comunicación.

2.3.4. Asynchronous Javascript And Xml

Según González y Moreno [González-Moreno; 2006], AJAX (acrónimo para Asynchronous JavaScript And XML. JavaScript y XML Asíncronos), es un enfoque de desarrollo basado en un conjunto de tecnologías ya

existentes, agrupadas para presentar información e interactuar dinámicamente, de manera asíncrona, con un servidor Web.

Entre las tecnologías que agrupa AJAX se destacan las siguientes como las principales:

a) *HTML y CSS: para la presentación, estructuración y formato del contenido.*
b) *DOM (Document Object Model): Con el modelo de objetos del documento se logra obtener la estructura del documento HTML. Utilizando esta estructura se pueden agregar, eliminar y modificar, de manera dinámica, elementos de la página mediante el uso de la tecnología JavaScript.*
c) *XML: Para el intercambio de datos entre el cliente (navegador Web) y el servidor.*
d) *JavaScript: Mediante esta tecnología del lado del cliente se realizan las peticiones de manera asíncrona y junto con el manejo del DOM, se logra la interacción dinámica con el usuario.*

A diferencia de las aplicaciones Web clásicas, las aplicaciones AJAX pueden enviar peticiones al servidor sin interrumpir la interacción entre el usuario y la aplicación. Estas aplicaciones mantienen la página actual cargada y evitan el típico tiempo de espera entre peticiones, brindándole al usuario la posibilidad de continuar interactuando con la aplicación mientras se realiza la petición, por ejemplo, llenando otros campos de un formulario. Esto es posible gracias al uso de peticiones en segundo plano (peticiones asíncronas).

2.3.5. Prototype

Según Prototype Org [U-PT-ORG], Prototype es un framework que promueve el desarrollo fácil de aplicaciones Web dinámicas. Asimismo cuenta con herramientas únicas y fáciles de usar para el desarrollo y con

las mejores librerías para la utilización de AJAX. No obstante, posee también integración con el framework de desarrollo rails lo cual potencia y extiende sus capacidades de una manera abrumadora.

En el ámbito de este Trabajo Especial de Grado se utiliza la integración de los frameworks rails y Prototype para maximizar las funcionalidades que requieren AJAX así como los efectos especiales incluidos en la interfaz grafica de usuario.

Entre las principales bondades de Prototype aprovechadas se encuentran el uso de AJAX, la fácil y rápida manipulación de elementos del DOM, la serialización de formularios y elementos de los mismos y las capacidades de ejecución de funciones mediante la captura de eventos, como por ejemplo cambios en los campos de un formulario.

2.3.6. Swift 3D

De acuerdo con sus creadores, electric Rain [U-SW-ER], Swift 3D es una aplicación de software que permite a los desarrolladores crear e importar modelos 3D, así como animarlos, manipularlos y exportarlos para su uso en múltiples plataformas como documentos de Flash y PaperVision3D.

Swift3D provee gran cantidad de funcionalidades interesantes y sin embargo es mucho más ligero que otros programas de edición 3D como 3Dstudio. De esta forma se destaca un importante punto, el cual es la integración con la suite de animaciones Flash.

Debido a que permite el trabajo de diseño y además la creación de animaciones pudiendo editar frame a frame y la exportación al formato de animaciones de adobe (SWF), swift3D se convirtió en una opción crucial para el diseño de los elementos del centro de datos a ser utilizados en el presente Trabajo Especial de Grado.

La figura 2.1 muestra la interfaz grafica de usuario de la aplicación, donde se puede apreciar el entorno completo de desarrollo y el mismo utilizado en el presente trabajo para realizar las animaciones con las cuales el usuario trabajará constantemente.

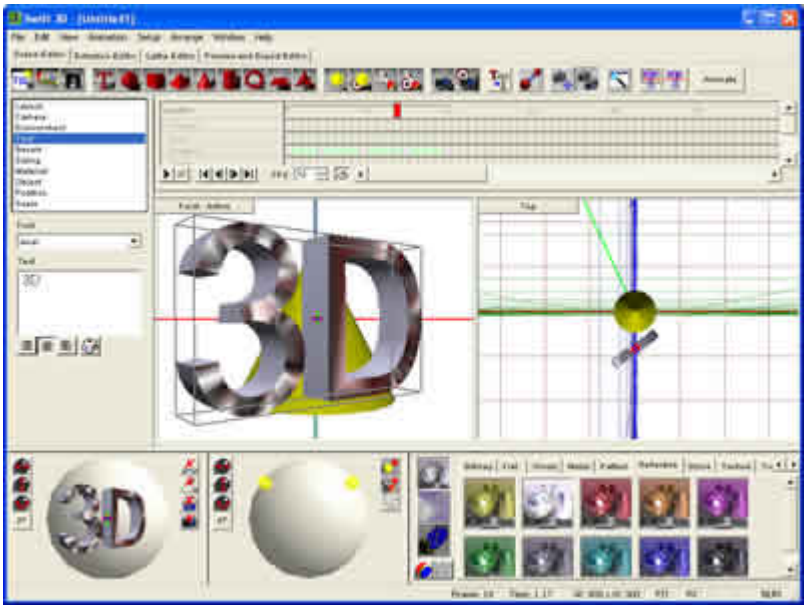

Figura 2.1 Interfaz gráfica de usuario Swift3D

2.3.7. Distributed Ruby (DRB)

DRB es una librería de desarrollo del lenguaje de programación Ruby, que permite a programas escritos en dicho lenguaje comunicarse de manera remota o en la misma máquina con otros programas que soporten DRB. Esta librería utiliza RMI (Remote Method Invocation – Llamadas a procedimientos remotos) para pasar comandos y parámetros entre los procesos.

2.3.8. Threads

En un sistema operativo, un Thread, hilo de ejecución o subproceso es una característica que permite a una aplicación realizar varias tareas a

la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

Un hilo es básicamente una tarea que puede ser ejecutada en paralelo con otra tarea.

Los hilos de ejecución que comparten los mismos recursos, sumados a estos recursos, son en conjunto conocidos como un proceso. El hecho de que los hilos de ejecución de un mismo proceso compartan los recursos hace que cualquiera de estos hilos pueda modificar éstos. Cuando un hilo modifica un dato en la memoria, los otros hilos acceden a ese dato modificado inmediatamente.

2.3.9. Sensors

Sensors es una librería que trabaja sobre la plataforma UNIX/Linux que permite conocer con exactitud el nivel de las temperaturas de trabajo de los componentes internos de un componente de hardware. Con esto podemos prevenir problemas de sobrecalentamiento y desgaste efectivo del hardware por un mal funcionamiento del sistema.

Para realizar esta labor, sensors se encarga de administrar los sensores incluidos en el hardware y desde los que se pueden monitorizar las temperaturas. El nombre especifico de esta librería es "lm-sensors". En resumen, provee información de ciertos sectores de la placa, además de los rangos efectivos de funcionamiento y alarmas en caso de que dichos recursos no se encuentren dentro de los rangos establecidos como normales.

CAPÍTULO III

MARCO APLICATIVO

Dando continuidad al desarrollo del presente Trabajo Especial de Grado y para lograr los objetivos planteados, en el siguiente capítulo se describe la aplicación del método de desarrollo propuesto en el capítulo uno.

En ese sentido, debido a que el método de desarrollo propuesto es iterativo e incremental, se proponen cinco iteraciones compuestas por las diferentes tareas de desarrollo, donde cada iteración va a estar estructurada y guiada por las fases que se realizan en dicha metodología, las cuales son Inicio, Elaboración y construcción.

El número de iteraciones y las tareas a realizar en cada una se determinaron por las divisiones que hay entre los diferentes módulos de la aplicación de diseño de centros de datos. Es decir, las tareas están bien definidas y diferenciadas entre ellas y son encapsuladas en tres grandes módulos. Asimismo en cada iteración se realizan modificaciones de las iteraciones anteriores con la finalidad de corregir errores y agregar nuevas funcionalidades a los módulos previos. Dichas iteraciones se definen a continuación:

- *Iteración 1: Esta iteración alcanza el análisis, diseño y desarrollo del módulo cargador de datos del sistema el cual permite la carga de todos los componentes del centro de datos y permite la inclusión de características y software asociado a dichos componentes.*

Además de esta iteración se obtendrá la primera versión del modelo de la base de datos.

- *Iteración 2: Esta iteración plantea el análisis, diseño y desarrollo del módulo más grande de la aplicación, el cual es el modulo de diseño. Además se obtiene el refinamiento del modelo de datos y se realiza una revisión del modulo cargador de datos para corregir posibles errores o agregar funcionalidades no contempladas en la primera iteración.*
- *Iteración 3: Esta iteración plantea el refinamiento de las funcionalidades desarrolladas en las iteraciones uno y dos. Cabe destacar que esta iteración solo consta de fase de elaboración y fase de construcción, ya que el objetivo es resolver problemas y refinar soluciones a requerimientos de los módulos cargador de datos y diseñador, así como agregar, modificar o eliminar funcionalidades a los mismos para optimizar su funcionamiento.*
- *Iteración 4: En esta iteración se realiza el análisis, diseño y desarrollo el modulo visualizador del centro de datos, el cual recopila la información del cargador de datos y del diseñador y las muestra para que el usuario pueda visualizar los elementos del centro y además ver la información que ellos contienen. Además en esta iteración se obtiene el modelo de datos final de la aplicación. Por último en esta iteración se agregan funcionalidades generales del sistema como autenticación, manejo de sesiones, adición de nuevos usuarios, cambio de contraseñas y se realiza una revisión completa del sistema una vez integrado además de las pruebas necesarias para verificar su correcto funcionamiento.*
- *Iteración 5: En esta última iteración se realiza el análisis, diseño y desarrollo del componente para el monitoreo del hardware del centro de datos. Dicho componente se encargará de obtener*

información sobre temperatura, velocidad de ventiladores, memoria y demás información de hardware que se considere crítica para el correcto funcionamiento de los elementos del centro de datos. Es necesario destacar que este componente obtendrá datos en tiempo real de dichos elementos de hardware.

Asimismo, según la metodología de desarrollo, cada iteración se ha dividido en 3 fases de desarrollo las cuales se definen a continuación:

- *Fase de Iniciación: Esta fase consiste en determinar el alcance del desarrollo y la comprensión de los requerimientos del sistema, para así poder obtener un buen funcionamiento de la iteración que se desea desarrollar. Para llevar a cabo las tareas relacionadas con esta fase se realizó lo siguiente:*
  - *Análisis de la iteración actual y estudio de la iteración anterior con el fin de realizar el acoplamiento y las adaptaciones necesarias para el correcto funcionamiento del sistema.*
  - *Levantamiento de la información de la iteración actual.*
  - *Revisión y corrección de iteraciones anteriores.*
- *Fase de Elaboración: En esta fase se realiza el análisis de requerimientos y se diseña la solución del problema. Además se realiza la generación de los artefactos de la iteración. Para llevar a cabo las tareas relacionadas con esta fase se realizó lo siguiente:*
  - *Diseño del modelo de datos correspondiente a la iteración actual.*
  - *Establecimiento de la arquitectura de la aplicación para la iteración actual.*
  - *Realización de los diagramas de casos de uso de la iteración con sus respectivas descripciones.*
  - *Definición del esquema de comunicaciones de la aplicación, en caso de haberlas.*

- *Elaboración de los prototipos de interfaz necesarios para la interacción con el usuario final del sistema en el módulo correspondiente a la iteración actual.*

➢ *Fase de construcción: En esta fase se elabora la codificación de un módulo operativo, tomando en cuenta el diseño que se elaboró en la fase anterior. Para llevar a cabo las tareas relacionadas con esta fase se realizó lo siguiente:*

- *Definición de los modelos necesarios para la correspondencia objeto relacional de la iteración actual.*
- *Elaboración de la arquitectura tecnológica que soportará la iteración actual y posteriormente la aplicación.*
- *Definición de los controladores necesarios para el manejo del flujo de información entre las interfaces de usuario y los modelos de persistencia y lógica de negocios de la iteración actual.*
- *Se desarrolla código legible y documentado, que permitirá facilitar los cambios y mejoras en nuevas versiones de la aplicación e iteraciones posteriores.*
- *Integración de la fase anterior con la fase actual.*
- *Realización de pruebas de la iteración actual y la integración con fases anteriores.*

3.1. Iteración I

Durante la primera iteración se realizó el análisis, diseño y desarrollo del módulo cargador de datos del sistema el cual permite la carga de todos los componentes del centro de datos. Además se realizó el diseño inicial del modelo de datos del sistema, necesario para el módulo antes mencionado.

3.1.1. Fase de Iniciación

En esta fase se realizaron las tareas de levantamiento de información y definición de requerimientos necesarios para el desarrollo del sistema cargador de datos.

En principio, la recopilación de requerimientos de software para sistema cargador fueron los siguientes:

- *Una interfaz rápida y eficiente que permita la edición de información de elementos del centro de datos.*
- *Funcionalidad que permita la adición de nuevos elementos a la lista de elementos disponibles.*
- *Funcionalidad que permita remover elementos de la lista de elementos disponibles.*
- *Capacidades de diferenciar entre elementos de hardware y software*
- *Capacidad de asociar y desasociar software a elementos de hardware.*
- *Funcionalidad de búsqueda que facilite la selección de los elementos.*
- *Permitir la agregación de nuevas características a cada elemento en particular para su personalización.*

3.1.2. Fase de Elaboración

En base a los requerimientos definidos en la fase anterior se seleccionó una arquitectura MVC para el diseño de la solución, la cual tiene como objetivo principal separar la lógica de negocios de la lógica de diseño y de la capa de datos. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y en caso de posibles cambios en el sistema sólo se modifica la capa requerida sin afectar rigurosamente las demás capas. Además permite la fácil integración de los

módulos del sistema ya que las funcionalidades están debidamente separadas y organizadas.

En la figura 3.1 se muestra el modelo vista controlador utilizado en el presente trabajo especial de grado, donde los modelos representan la persistencia de datos, los controladores se encargan de distribuir el flujo de ejecución entre modelos y vistas, además de ejecutar cierta lógica de aplicación y las vistas se encargan de proveer una interfaz al usuario de la aplicación mediante el browser.

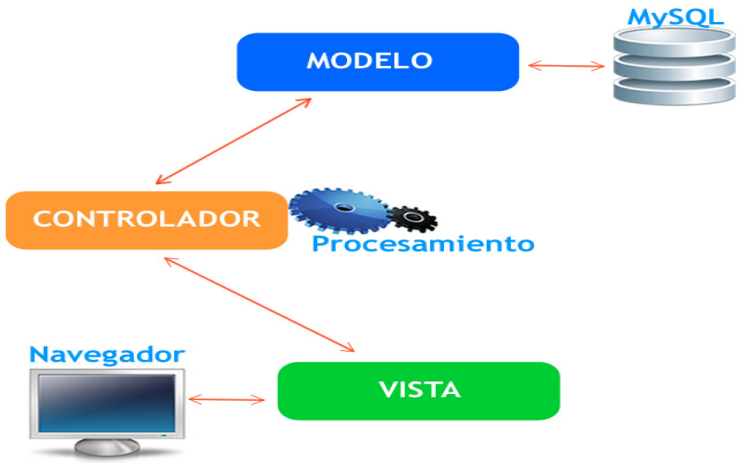

Figura 3.1 Arquitectura Modelo-Vista-Controlador

En esta fase se diseñó y desarrollo el modelo de datos del módulo cargador de datos, el cual consta de nueve estructuras que reflejan las necesidades de almacenamiento de dicho módulo. En la figura 3.2 Se muestra el diseño parcial (sólo la parte correspondiente al cargador de datos).

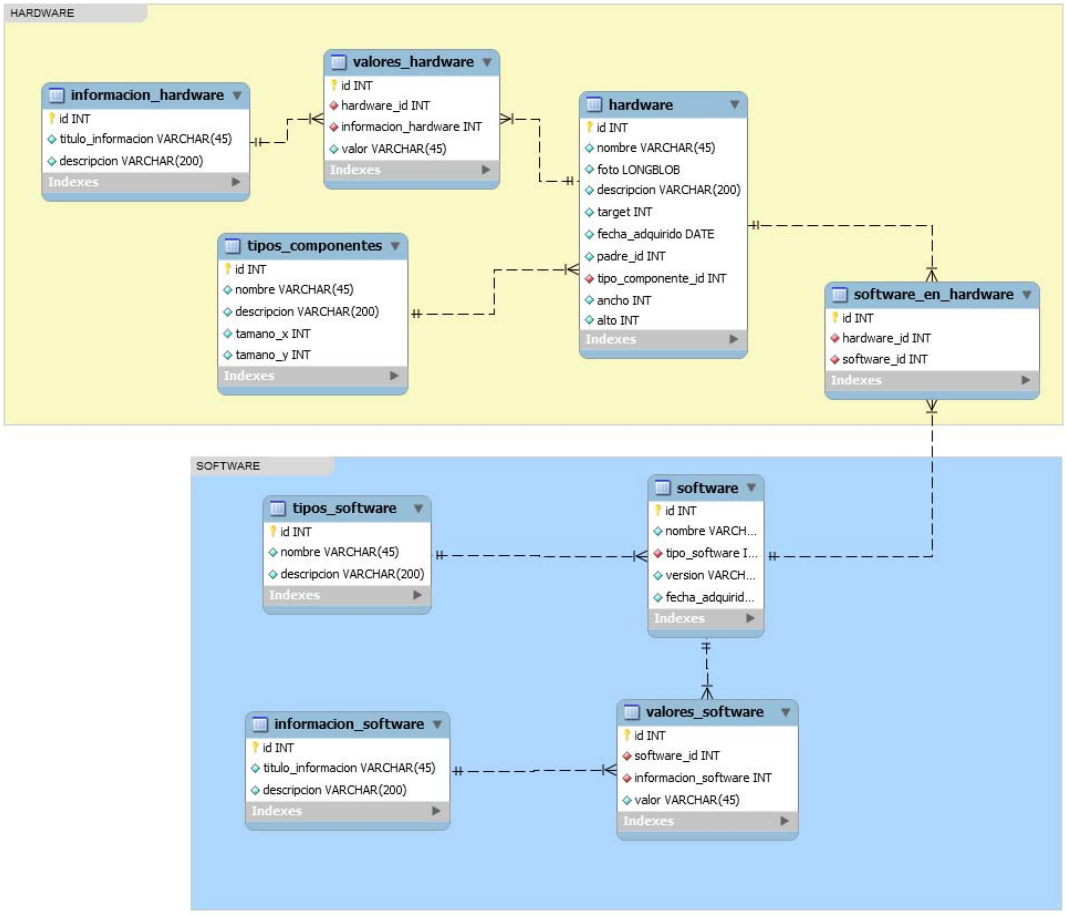

Figura 3.2 Diseño del modelo de datos para el módulo cargador de datos

De acuerdo a los requerimientos de este módulo, tanto el hardware como el software deben tener un conjunto de características asociadas y deben poder ser agregadas por el usuario del sistema, es decir, dichas características no pueden estar fijas como atributos de una estructura sino que deben separarse en otra estructura y luego asociarse posteriormente como se realiza en "hardware" e "informacion_hardware", con una estructura adicional "valores_hardware" que soporta la multiplicidad n:m de esta relación, además de contener el atributo "valor" que define el valor de la característica del hardware. Análogamente ocurre con las estructuras "software", "informacion_software" y "valores_software".

Otro de los requerimientos de este módulo es la posibilidad de enlazar software al hardware lo cual se refleja en la relación "software_en_hardware" del modelo. Por último las estructuras de "tipos_componentes" y "tipos_software" definen el tipo de hardware y software respectivamente, de los elementos del centro, por ejemplo tipos de componentes como enrutadores, torres de servidores y software como servidores de aplicaciones, monitores de red, etc.

Continuando con el diseño del módulo cargador de datos, se definieron y diseñaron los diagramas de casos de uso que permitirán reflejar las interacciones entre los usuarios y las funcionalidades del sistema. A continuación se presentan los casos de uso para el módulo cargador de datos divididos en tres niveles:

- **Nivel 0:** *Muestra de una manera general la interacción de usuario con el sistema. Concretamente, permite conocer a los actores que interactúan con el sistema, en este caso el módulo cargador de datos. Los casos de uso de nivel 0 del cargador se muestran en la figura 3.3.*

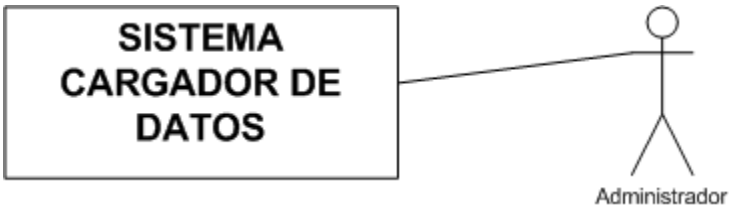

Figura 3.3 Diagrama de casos de uso del cargador – Nivel 0

- **Nivel 1:** *En este nivel podemos apreciar un mayor nivel de detalle, ya que se muestra qué funcionalidades existen dentro del sistema. Los casos de uso de nivel 1 se muestran en la figura 3.4.*

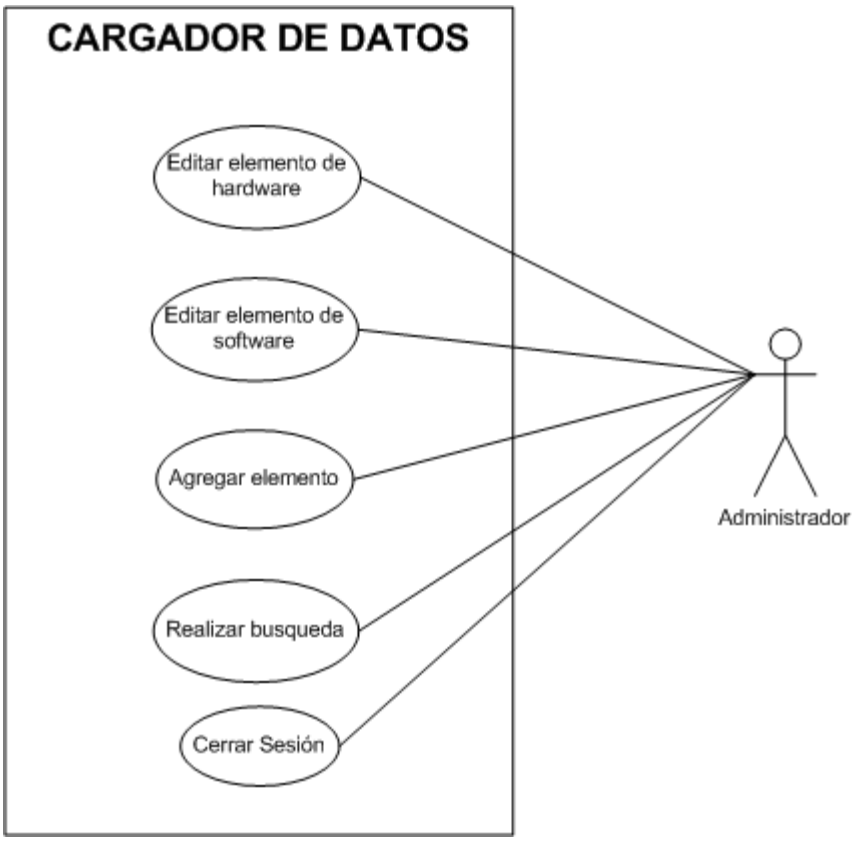

Figura 3.4 Diagrama de casos de uso del cargador – Nivel 1

Además del diagrama de casos de uso de la figura 3.4, a continuación se describen cada uno de los casos de uso que en él se encuentran:

- **Editar elemento de hardware:** *Permite editar la información de un elemento de hardware que haya sido cargado en el sistema*

  Pre-condición: *El usuario debe haber agregado y luego seleccionado el elemento de hardware.*

  Post-condición: *Los cambios realizados por el usuario deben reflejarse en el sistema.*

- **Editar elemento de software:** *Permite editar la información de un elemento de software que haya sido cargado en el sistema.*
  Pre-condición: *El usuario debe haber agregado y luego seleccionado el elemento de software.*
  Post-condición: *Los cambios realizados por el usuario deben reflejarse en el sistema.*

- **Agregar elemento:** *Permite al usuario agregar un elemento de hardware o software al sistema, pudiendo asociar además características a dichos elementos.*
  Pre-condición: *Ninguna.*
  Post-condición: *Los cambios realizados por el usuario deben reflejarse en el sistema.*

- **Realizar búsqueda:** *Permite al usuario realizar búsqueda de un elemento en la lista de la barra lateral izquierda.*
  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento.*
  Post-condición: *El elemento que coincida con el patrón de búsqueda debe ser resaltado.*

- **Cerrar Sesión:** *Permite al usuario salir del sistema correctamente eliminando los datos de su sesión actual.*
  Pre-condición: *El usuario debe haber iniciado una sesión en el módulo previo al cargador de datos.*
  Post-condición: *El sistema debe permitir la salida del usuario y terminar la sesión.*

Por ser una funcionalidad importante dentro del modulo cargador, la edición de características será mejor explicada mediante el diagrama de secuencia de la figura 3.5 que describe la interacción entre los componentes que forman dicha funcionalidad.

El diagrama de secuencia indica cuales son las clases involucradas en la ejecución de dicha funcionalidad, así como el flujo de ejecución de la misma. La mayoría de las funcionalidades del sistema siguen un flujo similar, basadas cada una en las respectivas interacciones mostradas en los diagramas de clase de cada iteración. Asimismo, las interfaces utilizadas para esta funcionalidad serán definidas posteriormente en la fase de construcción.

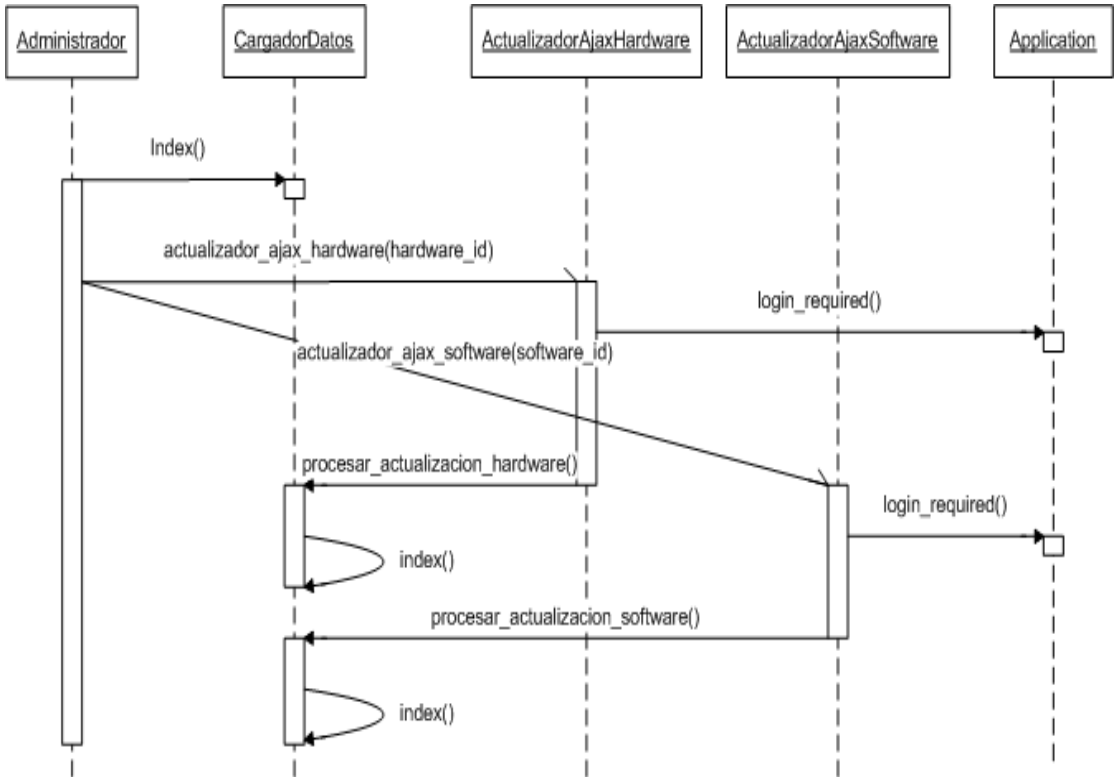

Figura 3.5 Diagrama de secuencia para la edición de características en el cargador de datos

- **Nivel 2:** *En este nivel se tiene un mayor grado de refinamiento, ya que se describen en detalle ciertos casos de uso en específico de nivel 1 que aún pueden descomponerse en otros casos de uso adicionales. A continuación se pueden apreciar los casos de uso de nivel dos desde la figura 3.6 a la 3.8.*

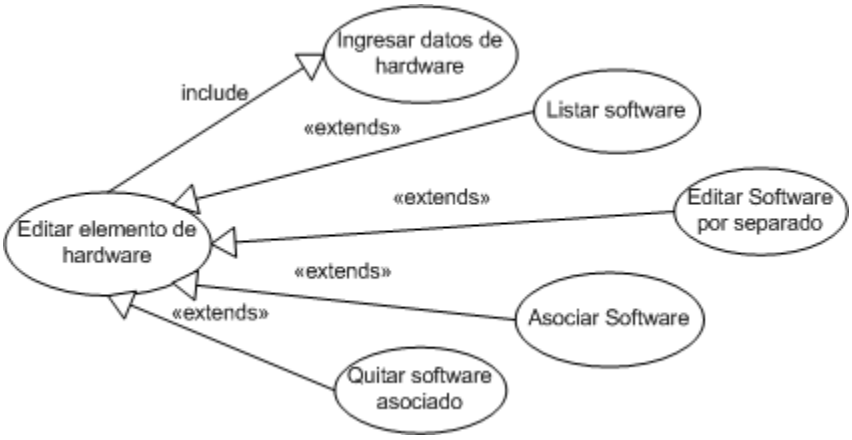

Figura 3.6 Diagrama de caso de uso (Editar elemento de hardware) – Nivel 2

Figura 3.7 Diagrama de casos de uso (Editar elemento de software) – Nivel 2

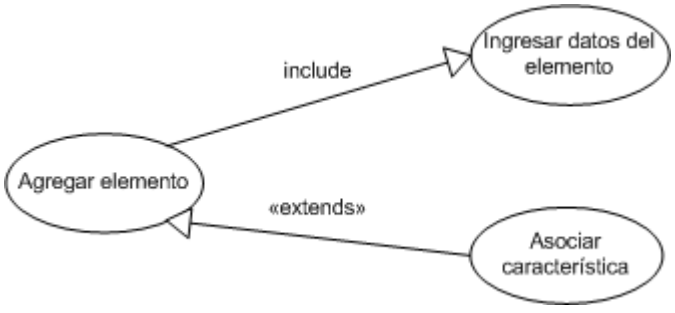

Figura 3.8 Diagrama de casos de uso (Agregar elemento) – Nivel 2

Igualmente a continuación se presentan las descripciones de los casos de uso de las figuras 3.6 a la 3.8.

- **Ingresar datos de hardware:** *Permite al usuario ingresar los valores de las características de un elemento de hardware.*
  Pre-condición: *El usuario debe haber agregado previamente el elemento de hardware y haberlo seleccionado.*
  Post-condición: *Los cambios realizados por el usuario deben reflejarse en el sistema.*

- **Listar software:** *Permite al usuario ver una lista de todo el software asociado a un hardware.*
  Pre-condición: *El usuario debe haber agregado previamente el elemento de hardware.*
  Post-condición: *Debe desplegarse una lista de software en caso de haber alguno asociado.*

- **Editar Software por separado:** *Permite al usuario editar el software asociado a un hardware.*
  Pre-condición: *El usuario debe haber agregado previamente el elemento de hardware y haberlo seleccionado.*
  Post-condición: *Se debe mostrar la interfaz de asociar software.*

- **Asociar software:** *Permite al usuario asociar nuevo software a un elemento de hardware seleccionado.*
  Pre-condición: *El usuario debe haber agregado previamente el elemento de hardware y haberlo seleccionado. Debe existir al menos un elemento de software.*

Post-condición: *Se debe asociar el software al hardware seleccionado.*

- **Quitar software asociado:** *Permite al usuario quitar la asociación entre un software y un elemento de hardware.*
  Pre-condición: *El usuario debe haber agregado previamente una asociación entre el hardware seleccionado y el software a desasociar.*
  Post-condición: *Se debe quitar la asociación entre el hardware y software seleccionado.*

- **Ingresar datos de software:** *Permite al usuario ingresar los valores de las características de un elemento de software.*
  Pre-condición: *El usuario debe haber agregado previamente el elemento de software y haberlo seleccionado.*
  Post-condición: *Los cambios realizados por el usuario deben reflejarse en el sistema.*

- **Ingresar datos del elemento:** *Permite al usuario ingresar los valores de las características de un elemento al momento de su creación.*
  Pre-condición: *Ninguna.*
  Post-condición: *Se debe agregar la característica al elemento.*

- **Asociar Característica:** *Permite al usuario asociar una característica a un elemento de hardware o software.*
  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento y haberlo seleccionado.*

Post-condición: *La característica debe adjuntarse al elemento seleccionado.*

Luego de las definiciones de los casos de uso, ya teniendo las funcionalidades que debe proveer el sistema, se realizó el diagrama de clases del módulo cargador de datos, el cual refleja las clases utilizadas, así como los atributos, métodos y relaciones entre las mismas. Estas clases son explicadas en detalle en la sección 3.1.3 del presente documento.

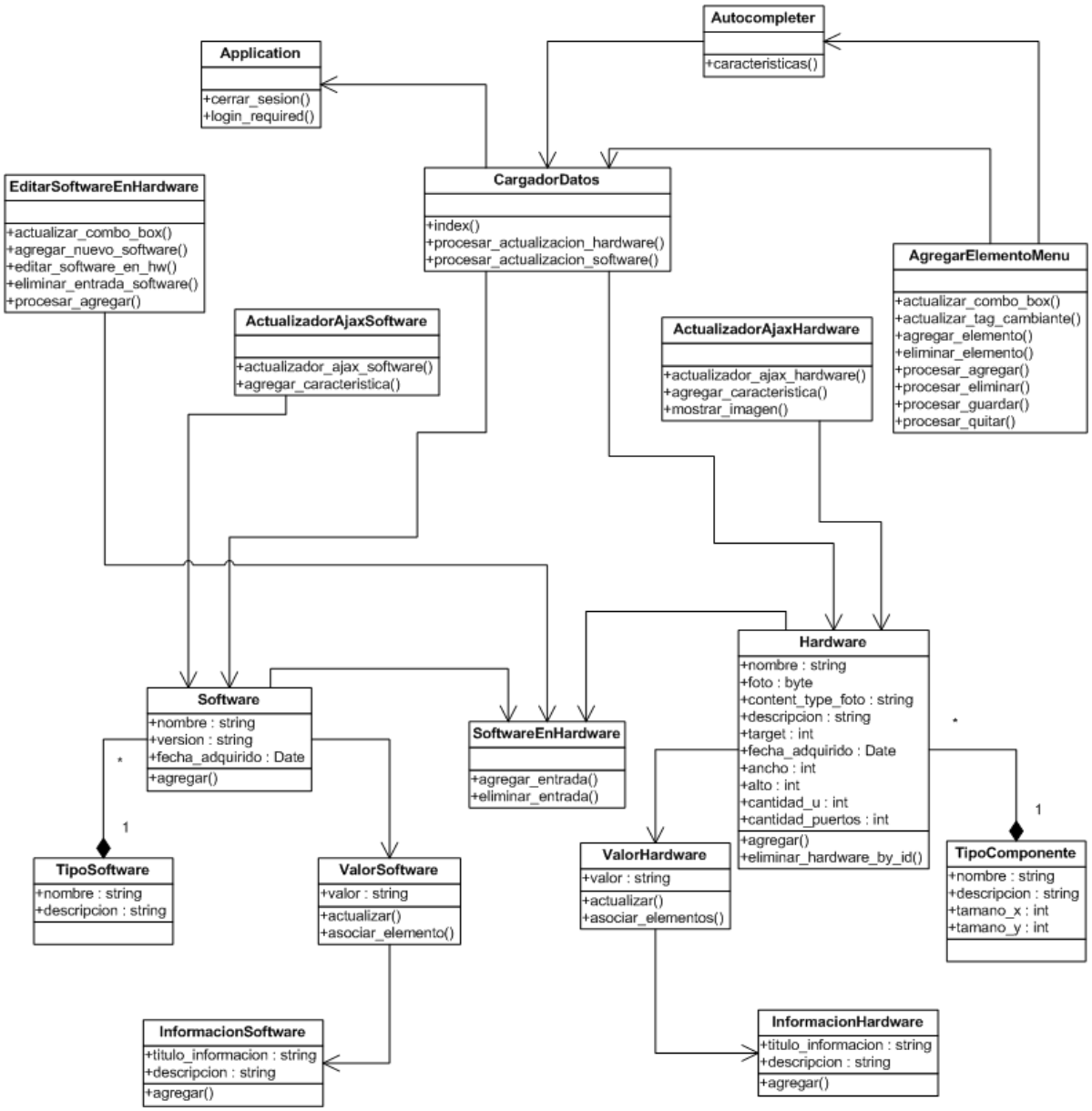

Figura 3.9 Diagrama de clases del cargador de datos

Debido a su complejidad, a continuación se presenta el diagrama de secuencia de la funcionalidad de agregación de nuevos elementos en la figura 3.10.

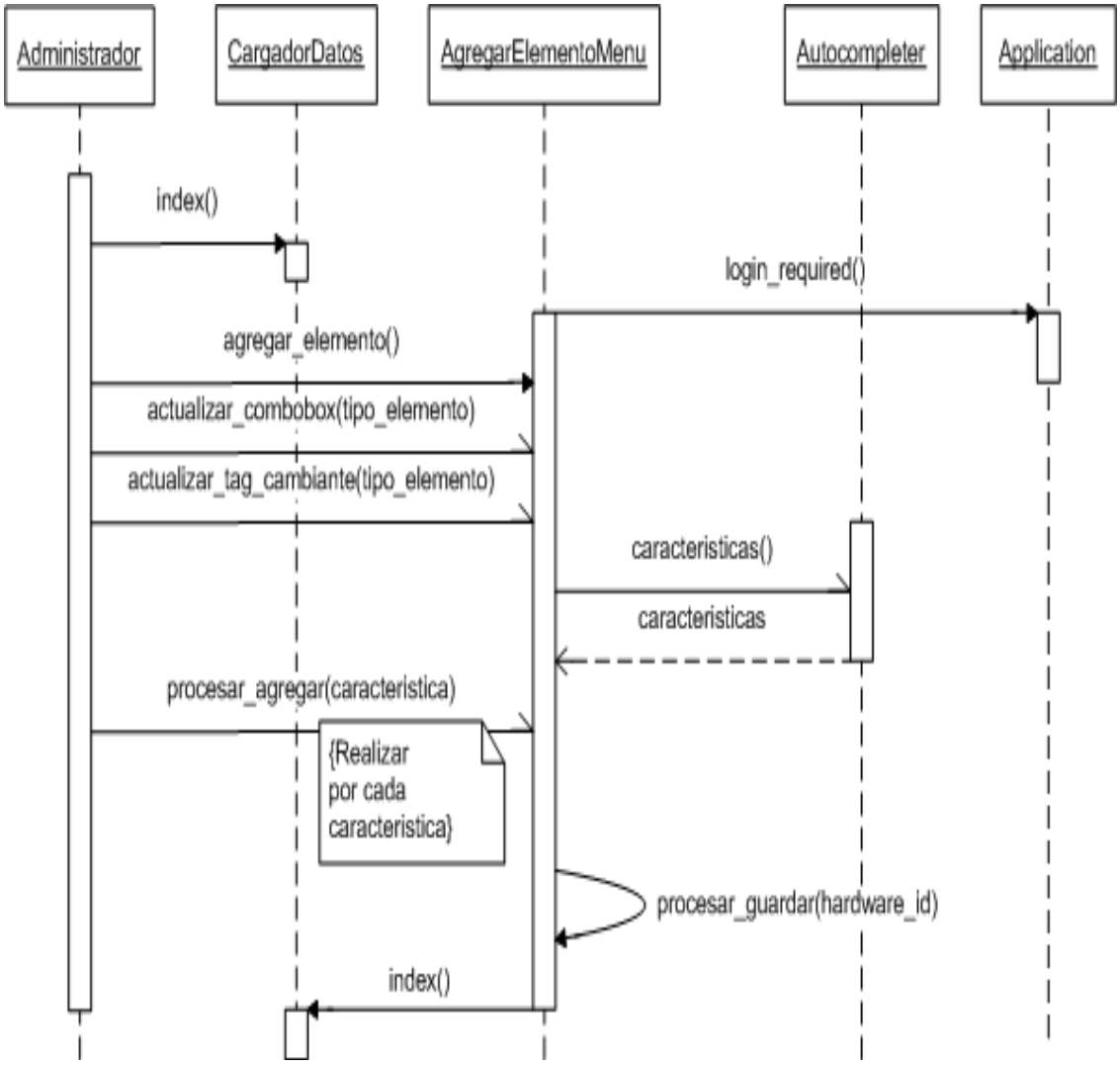

Figura 3.10 Diagrama de secuencia para agregar un elemento al sistema mediante el cargador de datos

Además del diagrama de clases y los diagramas de secuencia, se realizó el primer diseño de la interfaz del cargador de datos, la cual se muestra en la figura 3.11. Esta incluyó una barra lateral que listara los elementos agregados, con una caja de búsqueda y un área donde se pudiera editar la información de los elementos. Asimismo, cuenta con diferentes enlaces a opciones contempladas en los casos de uso.

Figura 3.11 Interfaz de usuario inicial del cargador de datos

Por último, en esta fase se realizó el diagrama o esquema de los elementos del cargador de datos. Este esquema permite tener mucho más claro la interacción de los elementos dentro del módulo de una manera más técnica y funcional. La figura 3.12 muestra este diagrama, en el cual se pueden apreciar todos los objetos que van a estar interactuando en el

módulo cargador de datos y que posteriormente en la fase de construcción van a ser explicados en detalle.

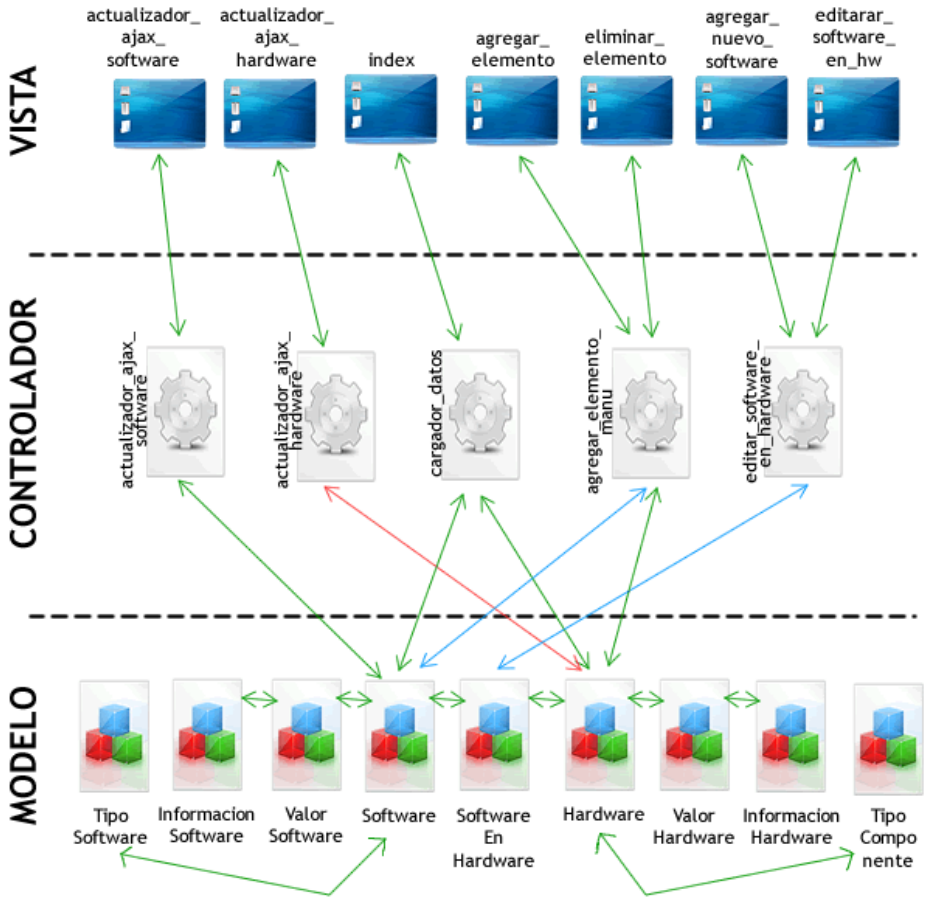

Figura 3.12 Esquema de archivos del cargador de datos

3.1.3. Fase de Construcción

En la presente fase se procede al desarrollo del diseño propuesto en las fases de inicio y elaboración del módulo cargador de datos. En este se desarrollaron los modelos necesarios para dar soporte a la persistencia de datos e interacción con la base de datos. La cantidad de modelos y como se corresponden con la arquitectura MVC se puede apreciar en la figura 3.12.

Los modelos en el módulo cargador de datos funcionan como interfaz entre la aplicación y la base de datos (Cómo se ilustra en la figura 3.1), permitiendo así la fácil interacción con la misma. A continuación se definen cada uno de los modelos, elaborados según las especificaciones del framework Ruby on Rails, del módulo cargador de datos propuestos en la figura 3.12:

- **Hardware**
- **ValorHardware**
- **InformacionHardware**
- **TipoComponente**
- **SoftwareEnHardware**
- **Software**
- **ValorSoftware**
- **InformacionSoftware**
- **TipoSoftware**

En resumen, el cuerpo de cada modelo se estructura en tres partes. La primera sería la definición de la clase, donde se especifica el nombre del modelo y de acuerdo con las convenciones del framework, estos se relacionan directamente con la tabla correspondiente a dicho modelo. Es de notar que todos los modelos deben heredar de "ActiveRecord::Base" para poder implementar estas características debido a la convención de modelos en rails. Una segunda parte serían las definiciones de relaciones con los otros modelos, las cuales reflejan las relaciones entre las tablas de la base de datos, y luego las validaciones sobre los datos del modelo. Por último, se consiguen las definiciones de las funciones asociadas a cada modelo, donde se implementa parte de la lógica de negocio del presente módulo. Es necesario destacar que cada modelo se corresponde con una parte del diagrama de clases realizado en la fase de elaboración.

A continuación, con la construcción del módulo cargador de datos, se procede a definir cada uno de los controladores propuestos en la figura 3.10, los cuales son los encargados de interactuar entre las vistas y los modelos, así como dirigir el flujo de ejecución en la aplicación. Estas clases complementan el diagrama de clases de la fase de elaboración. Los controladores, junto con los modelos en dicho diagrama, reflejan la interacción completa entre las clases del módulo cargador de datos.

- **ApplicationController:** *En este controlador se encuentran todas las funciones globales de la aplicación, es decir, que pueden ser accedidas desde cualquier otro controlador, modelo o vista. En él se realizan las validaciones de sesión de usuario para que sean accesibles en toda la aplicación.*

- **CargadorDatosController:** *Este es el controlador principal del módulo, el cual se encarga de procesar las solicitudes de actualización de los elementos de hardware y software. Las vistas del módulo cargador están estructuradas de tal forma que solo se actualice la parte central mediante Ajax y se mantenga el layout de la aplicación; esto hace que el rendimiento de la aplicación incremente dándole más rápida respuesta al usuario. La vista principal del cargador de datos se muestra en la figura 3.11.*

- **ActualizadorAjaxHardwareController:** *Este controlador es el encargado de la gestión de actualización del hardware. Mediante él se redirige a la vista de edición de los elementos de hardware y se realiza el procesamiento de agregación de características a dichos elementos. En la figura 3.13 se puede apreciar la vista de edición de características de un elemento de hardware que posteriormente invoca a la función "procesar_actualizacion_hardware" del controlador "CargadorDatosController" para culminar.*

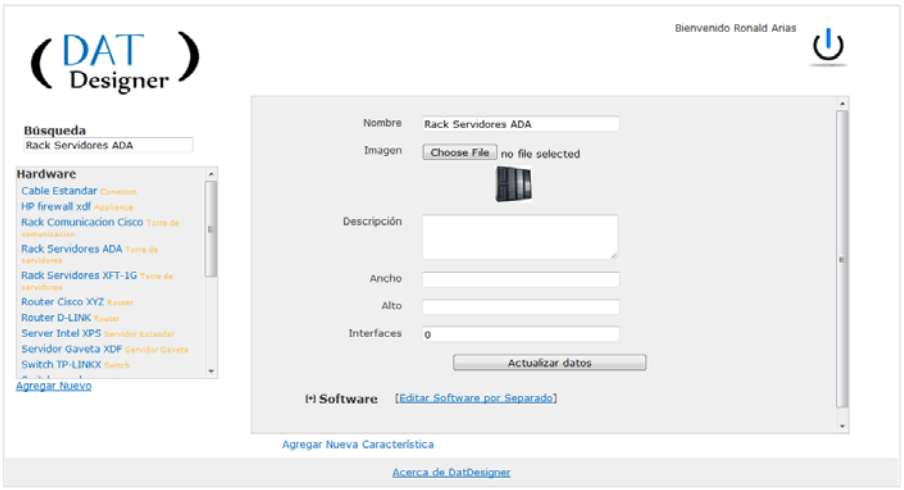

Figura 3.13 Interfaz de usuario ActualizadorAjaxHardwareController

- **ActualizadorAjaxSoftwareController:** *Este controlador es el encargado de la gestión de actualización del software. Mediante él se redirige a la vista de edición de los elementos de software y se realiza el procesamiento de agregación de características a dichos elementos. La vista asociada con este controlador sigue el mismo formato que la vista de la figura 3.13, salvo que no hay software asociado con los propios elementos de software. A continuación se presenta la implementación del presente controlador.*

- **AgregarElementoMenuController:** *Este controlador se encarga de la inclusión de nuevos elementos al sistema así como de la lógica para eliminarlos del mismo. El objetivo primordial es adicionar o sustraer elementos del centro de datos. Asimismo, esta parte del módulo cargador de datos tiene la particularidad de contener muchas interacciones asíncronas con el servidor. Como muestra de esta característica se puede apreciar en el código siguiente que las funciones "actualizar_tag_cambiante", "actualizar_combo_box_ajax", "procesar_agregar" y "procesar_*

quitar" son utilizadas solo para dichas interacciones asíncronas. Estas se encargan respectivamente de actualizar las listas desplegables de "Tipo de elemento" y "Categoría", e incluir o quitar elementos del área de "Características agregadas" que se muestran en la figura 3.14. Cabe destacar que estas características asíncronas mejoran el rendimiento de la aplicación e incrementan la satisfacción del usuario.

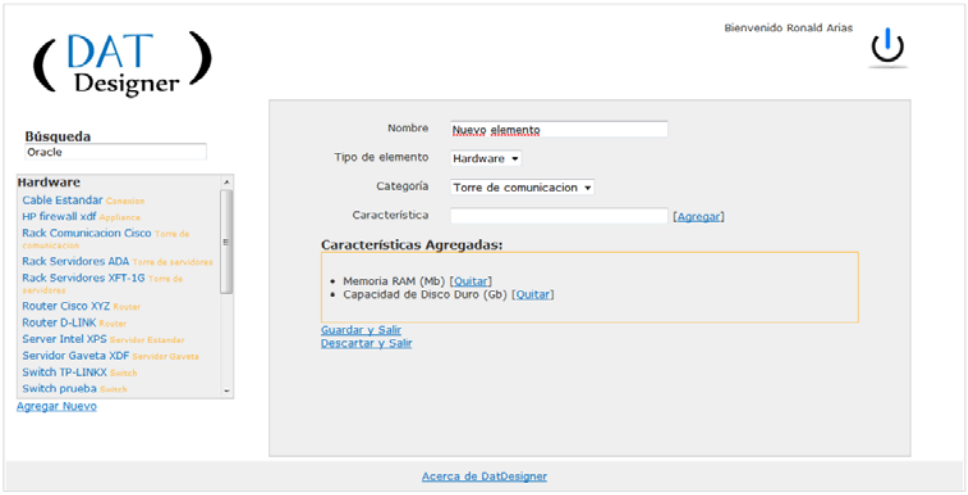

Figura 3.14 Interfaz de usuario AgregarElementoMenuController

- **EditarSoftwareEnHardwareController:** *Este controlador provee de las funcionalidades para asociar o desasociar software a los elementos de hardware. Su composición es sencilla aunque sigue la misma tendencia de solicitudes asíncronas al servidor que el controlador "AgregarElementoMenuController".*

- **AutoCompleterController:** *Este controlador se encarga de proveer a las vistas una funcionalidad de autocompletado para permitir al usuario agregar características a los elementos de manera fácil y rápida. Su labor consiste en listar, dependiendo de si*

el elemento es hardware o software, el conjunto de características disponibles que coincidan con un patrón de búsqueda del usuario. En caso de no haber ninguna que coincida, una nueva posibilidad de característica será agregada y estará disponible para su utilización posteriormente. La implementación se muestra a continuación y además, la figura 3.15 ejemplifica como se realiza el despliegue del menú de autocompletado.

```
class AutocompleterController < ApplicationController
   def caracteristicas
         @caracteristicas_hardw = InformacionHardware.find(:all)
         @caracteristicas_softw = InformacionSoftware.find(:all)
         render :layout=> false
   end

end
```

Figura 3.15 Interfaz de la función autocompletar

3.2. Iteración II

Durante la segunda iteración se realizó análisis, diseño y desarrollo del módulo diseñador del sistema el cual permite realizar el diseño y organización de todos los elementos del centro de datos. Cabe destacar que este es el módulo más grande y complejo del sistema. Asimismo, se refinó el modelo de datos del sistema, se integró la iteración número uno con la actual y se hicieron revisiones sobre el resultado de la iteración uno.

3.2.1. Fase de Iniciación

En esta fase se realizaron las tareas de levantamiento de información y definición de requerimientos necesarios para el desarrollo del sistema diseñador de centros de datos. Asimismo se realizó un análisis de la iteración anterior para facilitar la integración con la iteración actual y se realizó un listado de las modificaciones de la iteración uno a realizarse en la iteración actual.

La recopilación de requerimientos de software necesarios para sistema diseñador en esta fase fueron los siguientes:

- *Interfaz dinámica y fácil de usar, que permita al usuario arrastrar elementos de una lista a un área de diseño donde el usuario pueda organizar lógicamente el centro de datos.*
- *Motor de búsqueda que facilite la selección de los elementos.*
- *Barra de herramientas que provee de las funcionalidades de edición sobre los elementos del área de diseño.*
- *Capacidad de arrastrar elementos para posicionarlos en un lugar seleccionado del área de diseño.*
- *Capacidad para el anidamiento de elementos, es decir, permitir incluir elementos dentro de otros elementos según ciertas restricciones.*
- *Herramienta que permita realizar conexiones lógicas entre los elementos del centro de datos.*
- *Herramienta que permita eliminar elementos del área de diseño.*
- *Herramientas de "zoom in" y "zoom out" para permitir acercar o alejar respectivamente el visor del área de diseño.*
- *Opción que permita reflejar el área de diseño y las interconexiones de los elementos de manera persistente en la base de datos del sistema.*

- *Herramienta que permita eliminar interconexiones entre elementos del área de diseño.*

3.2.2. Fase de Elaboración

Para esta segunda iteración, se decidió utilizar el lenguaje ActionScript 3 sobre la plataforma Adobe Flash para darle dinamismo y presentación al módulo diseñador de centros de datos. En base a los requerimientos de interfaz gráfica del módulo se analizaron diferentes tecnologías, sin embargo todo ha convergido a la antes mencionada debido a las capacidades gráficas y facilidades que presta la suite CS4 de adobe Systems incorporated para el desarrollo.

Toda esta plataforma se incorporó sobre el Framework de desarrollo Ruby on Rails mediante el formato de intercambio que provee Flash, llamado SWF. Asimismo, por existir una comunicación directa de la aplicación en Flash con el framework y la base de datos, se estableció un esquema de comunicaciones que permitiera el intercambio de datos de manera rápida y eficiente.

La primera opción estudiada fue el intercambio de datos mediante mensajes XML, pero en vista de lo ineficiente de las comunicaciones ante la cantidad de datos a intercambiar y la falta de seguridad de dicha comunicación, se optó por la búsqueda de una mejora. Como mejora se propuso realizar las comunicaciones mediante Action Message Format (AMF – Descrito en el Marco Teórico), el cual establece que las comunicaciones entre el servidor de aplicaciones y el reproductor Flash se realizan en formato binario, lo cual aumenta considerablemente la eficiencia de las comunicaciones y se presta para un intercambio más seguro. La figura 3.16 muestra el esquema de comunicaciones implantado para el módulo diseñador.

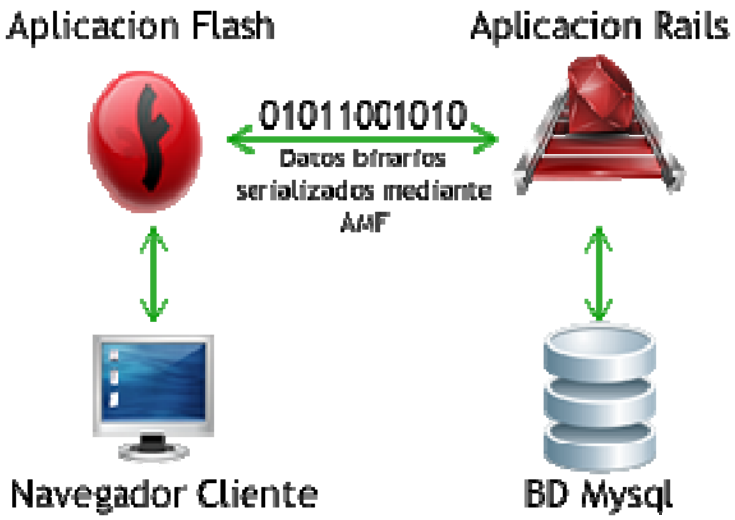

Figura 3.16 Esquema de comunicaciones del diseñador

En este esquema de comunicaciones podemos apreciar que el intercambio de datos se produce entre el servidor de aplicaciones de Rails y el documento flash de formato de archivo SWF. A pesar que este documento está integrado en el servidor no significa que el intercambio de información entre ellos sea implícito. Para realizar las comunicaciones, se implementó una interfaz REST (Representational State Transfer – Descrito en el Marco Teórico) en rails que permitiera el consumo y obtención de recursos de la aplicación, los cuales son extraídos por el servidor desde la base de datos. Adicional a esto, en la aplicación Flash se realizó una capa de comunicaciones con ActionScript 3 que permitió utilizar las capacidades internas de AMF para establecer la comunicación y permitir la invocación "remota" de recursos en el servidor. Esto incluso pudiera permitir que la aplicación flash fuese incrustada en una aplicación diferente y aun así poder consumir los recursos del servidor de manera remota.

Asimismo en esta fase, se realizó el diseño y desarrollo del modelo de datos necesario para soportar las actividades del diseñador de centros de datos. La interacción con la base de datos por parte de este módulo

permite llevar el área de diseño a una instancia persistente de la aplicación. Dicho modelo de persistencia se muestra en la figura 3.17.

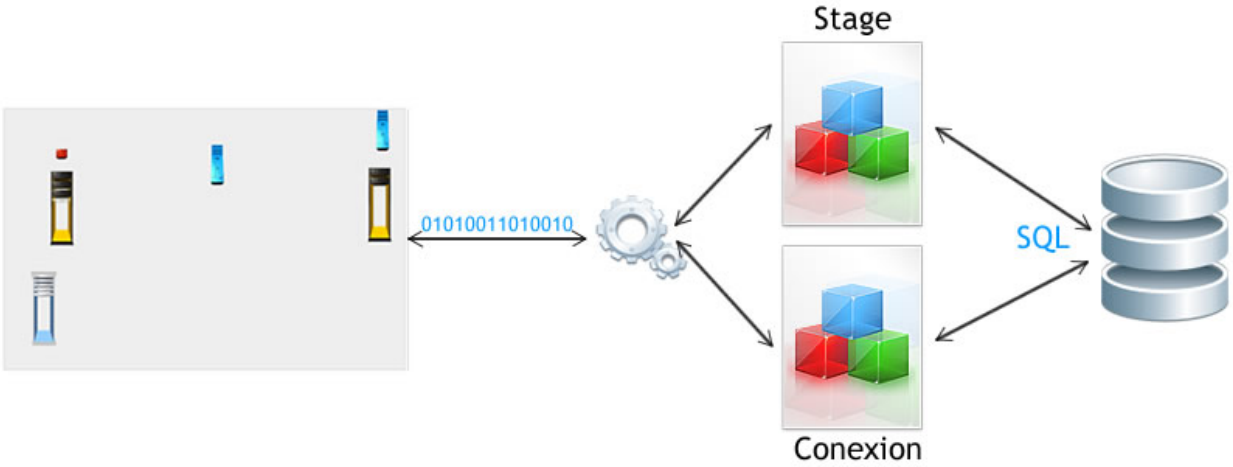

Figura 3.17 Persistencia del área de diseño a la base de datos

Debido a esto, al modelo de datos obtenido en la iteración anterior se le añaden dos estructuras adicionales que reflejan el área de diseño y las conexiones entre los elementos de dicha área (stage y conexiones respectivamente). En la figura 3.18 se pueden apreciar las estructuras antes mencionadas con el conjunto de atributos respectivos necesarios de cada elemento en el área de diseño. Entre los más resaltantes podemos apreciar las posiciones "x" y "y" que deben ser almacenadas ya que ActionScript maneja los elementos sobre un eje de coordenadas y el tipo de símbolo que representa la clase que debe ser instanciada al momento de recrear el elemento en el área de diseño. Asimismo, se puede apreciar en la estructura de conexiones que cada conexión consiste de una clave múltiple entre los elementos interconectados y además las interfaces de red por las cuales se interconectan.

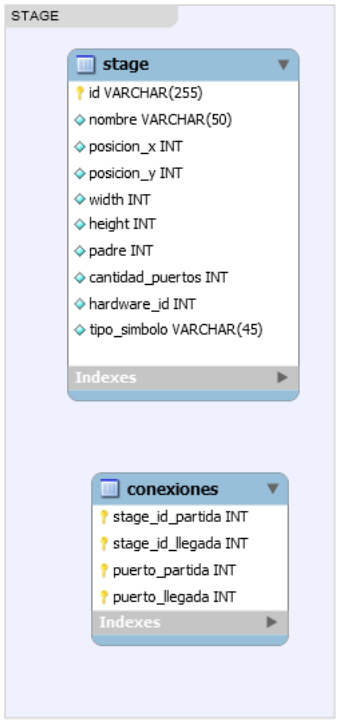

Figura 3.18 Diseño del modelo de datos del módulo diseñador

A continuación, como parte del diseño del módulo diseñador de centros de datos, se definieron y diseñaron los diagramas de casos de uso que permitirán reflejar las interacciones entre los usuarios y las funcionalidades del sistema. A continuación se presentan los casos de uso para el módulo diseñador de centros de datos divididos en tres niveles:

- **Nivel 0:** *Muestra de una manera general la interacción de usuario con el sistema. Los casos de uso de nivel 0 del módulo cargador de datos se muestran en la figura 3.19.*

Figura 3.19 Diagrama de casos de uso del diseñador – Nivel 0

- **Nivel 1:** *En este nivel podemos apreciar mayor detalle, ya que se muestra qué funcionalidades existen dentro del módulo diseñador. Los casos de uso de nivel 1 se muestran en la figura 3.20.*

Figura 3.20 Diagrama de casos de uso del diseñador – Nivel 1

Cabe destacar que este módulo es el más complejo y el que presenta mayor cantidad de funcionalidades en el sistema. Además del diagrama de casos de uso de la figura 3.20, a continuación se describen cada uno de los casos de uso que en él se encuentran y no fueron descritos anteriormente:

- **Agregar elemento al área de diseño:** *Permite al usuario arrastrar un elemento del panel lateral izquierdo al área de diseño.*

  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento.*

  Post-condición: *El elemento debe agregarse al área de diseño.*

- **Eliminar elemento del área de diseño:** *Permite al usuario eliminar un elemento del área de diseño.*
- **Pre-condición:** *El usuario debe haber agregado previamente al menos un elemento al área de diseño.*
- **Post-condición:** *El elemento debe ser removido del área de diseño, junto con todas sus conexiones e hijos.*

- **Agregar conexión:** *Permite al usuario agregar una conexión lógica entre dos elementos del área de diseño.*

  Pre-condición: *El usuario debe haber agregado previamente al menos dos elementos al área de diseño y estos deben poseer una interfaz de red disponible cada uno.*

  Post-condición: *Se debe realizar la conexión lógica entre ambos elementos y se debe mostrar un mensaje de éxito.*

- **Eliminar conexiones:** *Permite al usuario eliminar una conexión entre dos elementos del área de diseño.*
  Pre-condición: *El usuario debe haber agregado previamente dos elementos al área de diseño, y además deben tener una conexión entre ellos.*
  Post-condición: *Se debe eliminar la conexión y mostrar un mensaje de éxito.*

- **Buscar elemento:** *Permite al usuario realizar búsqueda de un elemento en la lista de la barra lateral izquierda.*
  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento.*
  Post-condición: *Se debe resaltar el elemento que coincida con el patrón de búsqueda.*

- **Realizar zoom-in:** *Permite al usuario acercar el área de diseño de manera que se puedan apreciar con más detalle los elementos del área de diseño.*
  Pre-condición: *El nivel de zoom debe estar uno por encima del máximo.*
  Post-condición: *Se debe acercar el área de diseño.*

- **Realizar zoom-out:** *Permite al usuario alejarse del área de diseño para tener una vista más general del mismo.*
  Pre-condición: *El nivel de zoom debe estar uno por encima del zoom mínimo.*
  Post-condición: *Se debe alejar el área de diseño.*

- **Moverse a través del área de diseño:** *Permite al usuario trasladarse de un lado a otro por el área de diseño.*
  Pre-condición: *El nivel de zoom debe estar al menos uno por encima del nivel mínimo.*
  Post-condición: *Se debe mover el foco del área de diseño.*

- **Salvar estado del área de diseño:** *Permite al usuario salvar el estado del área de diseño, y con ello la posición, tamaño, conexiones y demás datos de los elementos que en esta se encuentren.*
  Pre-condición: *Ninguna.*
  Post-condición: *Se debe reflejar persistentemente el área de diseño en la base de datos.*

➢ **Nivel 2:** *En este nivel se tiene el mayor grado de refinamiento, ya que se describen en detalle ciertos casos de uso en específico de nivel 1 que aún pueden descomponerse en otros casos de uso adicionales. A continuación se pueden apreciar los casos de uso de nivel dos en las figuras 3.21 y 3.22.*

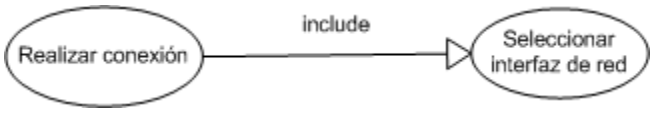

Figura 3.21 Diagrama de casos de uso (Realizar conexión) – Nivel 2

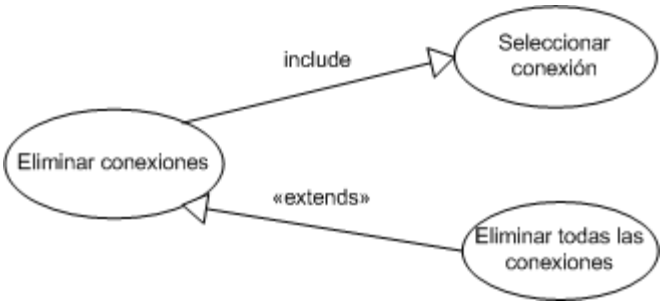

Figura 3.22 Diagrama de casos de uso (Eliminar conexiones) – Nivel 2

Igualmente a continuación se presentan las descripciones de los casos de uso de las figuras 3.21 y 3.22.

- **Seleccionar interfaz de red:** *Permite al usuario seleccionar la interfaz de red o puerto por el cual desea realizar la conexión.*

  Pre-condición: *El elemento seleccionado debe tener al menos una interfaz de red disponible.*

  Post-condición: *Se debe seleccionar la interfaz de red.*

- **Seleccionar conexión:** *Permite al usuario seleccionar la conexión que desea eliminar.*

  Pre-condición: *Debe existir al menos una conexión entre el elemento seleccionado y algún otro del área de diseño.*

  Post-condición: *Se debe seleccionar la conexión.*

- **Eliminar todas las conexiones:** *Permite al usuario eliminar todas las conexiones de un elemento del área de diseño seleccionado.*

  Pre-condición: *Debe existir al menos una conexión entre el elemento seleccionado y algún otro del área de diseño.*

Post-condición: *Se deben eliminar todas las conexiones entre el elemento seleccionado y aquellos con los que conecte.*

Asimismo, una vez obtenidas las funcionalidades requeridas y los casos de uso del sistema se puede definir el diagrama de clases del módulo diseñador de centros de datos, cuyas clases son explicadas en la fase de elaboración de la presente iteración.

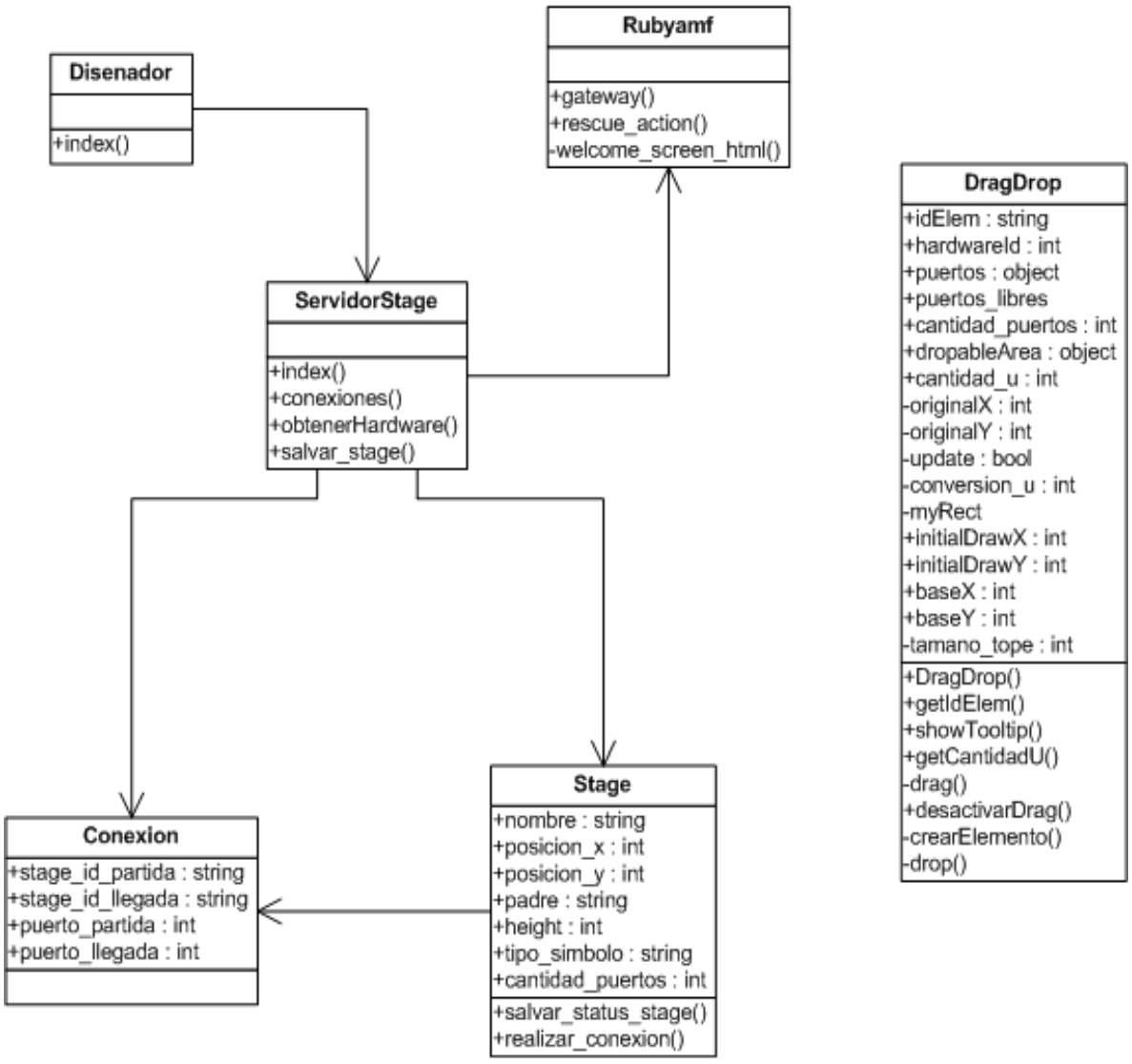

Figura 3.23 Diagrama de clases del diseñador de centros de datos

Dicho diagrama contiene las clases asociadas al módulo diseñador, no obstante, este no incluye toda su implementación, esto decir, que existe también desarrollo no orientado a objetos dentro de este módulo.

Para explicar mejor la interacción entre la clase DragDrop y el área de diseño, a continuación se muestra un diagrama de secuencia que ilustra la secuencia para agregar un nuevo elemento al área de diseño.

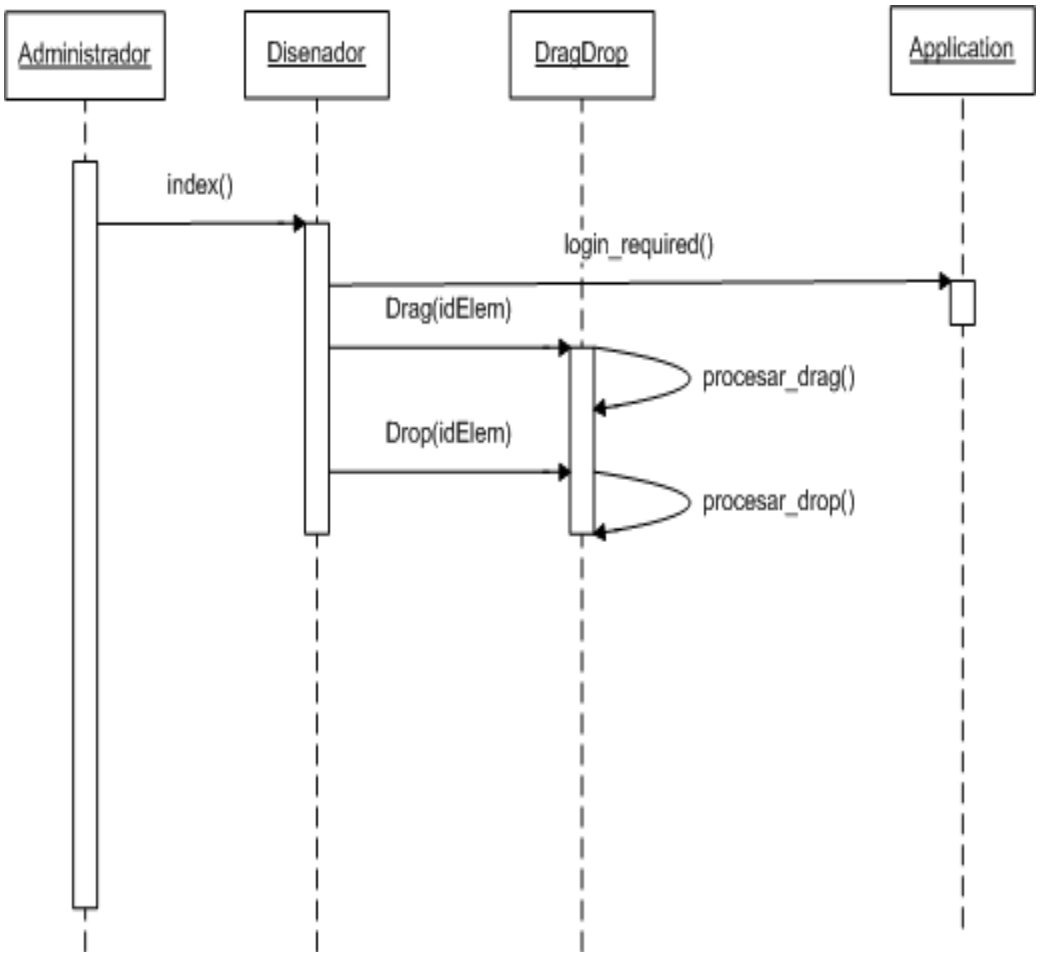

Figura 3.24 Diagrama de secuencia para insertar un elemento en el área de diseño del módulo diseñador de centros de datos

Posterior a la definición del diagrama de clases, se realizó el diseño de la interfaz del diseñador de centros de datos, la cual se muestra en la figura 3.25. Con la finalidad de mantener una sintonía en el diseño de la interfaz se trató de realizar este diseño lo más parecido posible del diseño de la interfaz de la iteración uno del el cargador de datos. Cabe destacar que el diseño de la interfaz no solo contó con la diagramación de la página sino que incluyó un conjunto de elementos de interfaz importantes como los iconos de la barra de herramientas y de las opciones de salvar y cerrar sesión e incluso de cada tipo elemento del centro de datos, los cuales fueron realizados en 3D con la aplicación Switf3D y serán mostrados con detalle en la siguiente fase. Asimismo, este diseño incluyó una barra lateral que listara los elementos agregados, con una caja de búsqueda y un área de diseño que representa el centro de datos en sí. Por último cuenta con las diferentes opciones contempladas en los casos de uso distribuidas entre la barra de herramientas y demás componentes del diseñador.

Figura 3.25 Interfaz de usuario inicial del diseñador

Por último, se definió el esquema a trabajar en el diseñador. La manera en que se dispuso el código en ActionScript fue por capas, es decir,

cada parte lógica del módulo diseñador fue distribuida en una capa distinta para darle mejor organización al código y hacerlo más legible y fácil de identificar. Sin embargo, todas las instancias de los elementos en cada capa son visibles por las demás, lo cual facilita enormemente el desarrollo. Además se desarrolló una clase llamada "DragDrop", de la cual hereda cada elemento del centro de datos, y que encapsula todas las propiedades y métodos de cada elemento. La figura 3.26 muestra el esquema antes descrito.

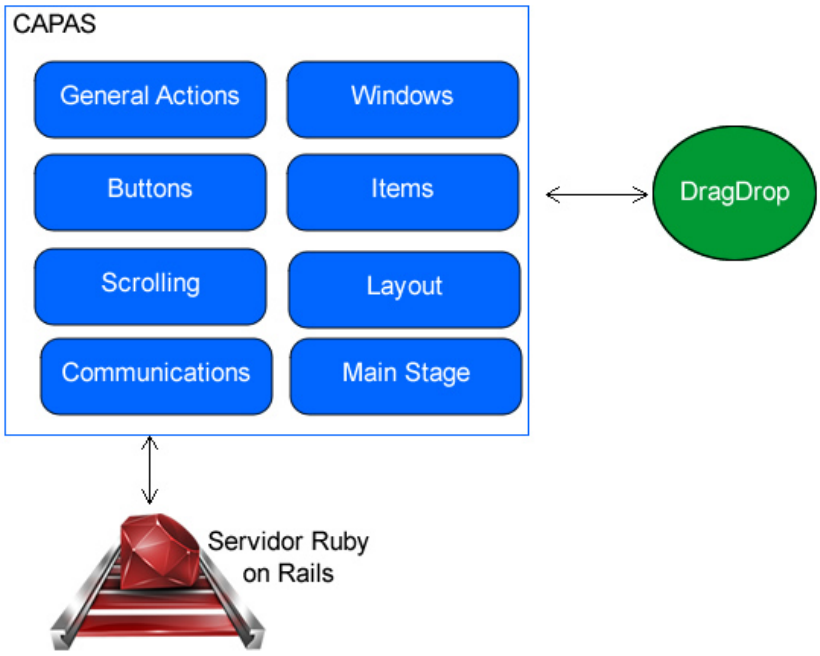

Figura 3.26 Esquema de capas del diseñador para ActionScript

Este esquema en capas puede ejemplificarse mediante un diagrama de clases. A continuación se presenta dicho diagrama para las capas, que dará mejor entendimiento del módulo diseñador. La definición de las clases que se encuentran en dicho diagrama, así como la definición de las capas, se encuentra en la fase de elaboración de la presente iteración.

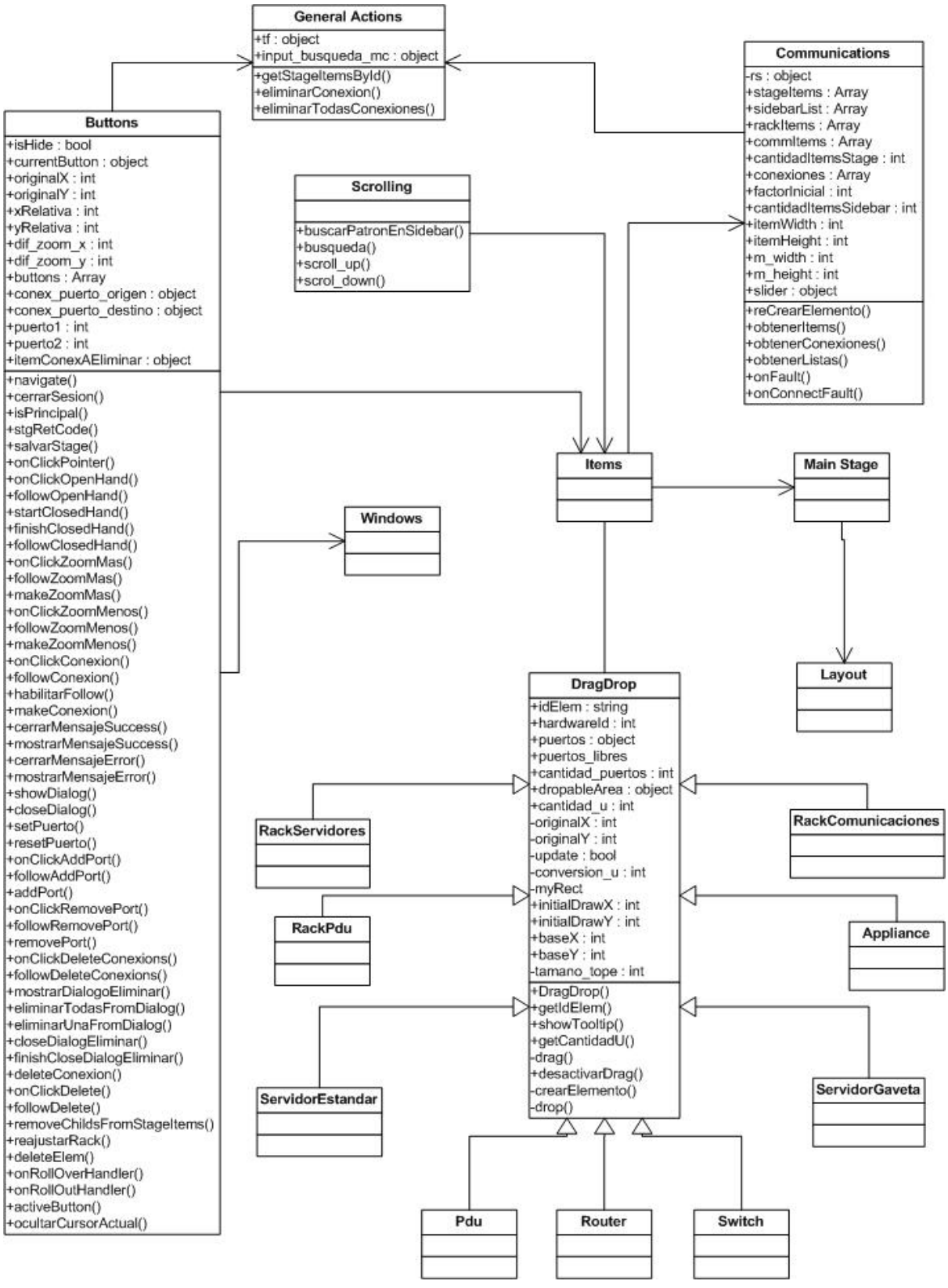

Figura 3.27 Diagrama de clases del diseñador, tomando capas como clases

Por último, para la fase de elaboración de la presente iteración, se presenta una manera de entrelazar la lógica expuesta en las figuras 3.16, 3.17 y 3.26, la cual es un diagrama de secuencia que permita apreciar los elementos que intervienen en la comunicación entre el servidor rails y ActionScript mediante AMF. A continuación se presenta dicho diagrama.

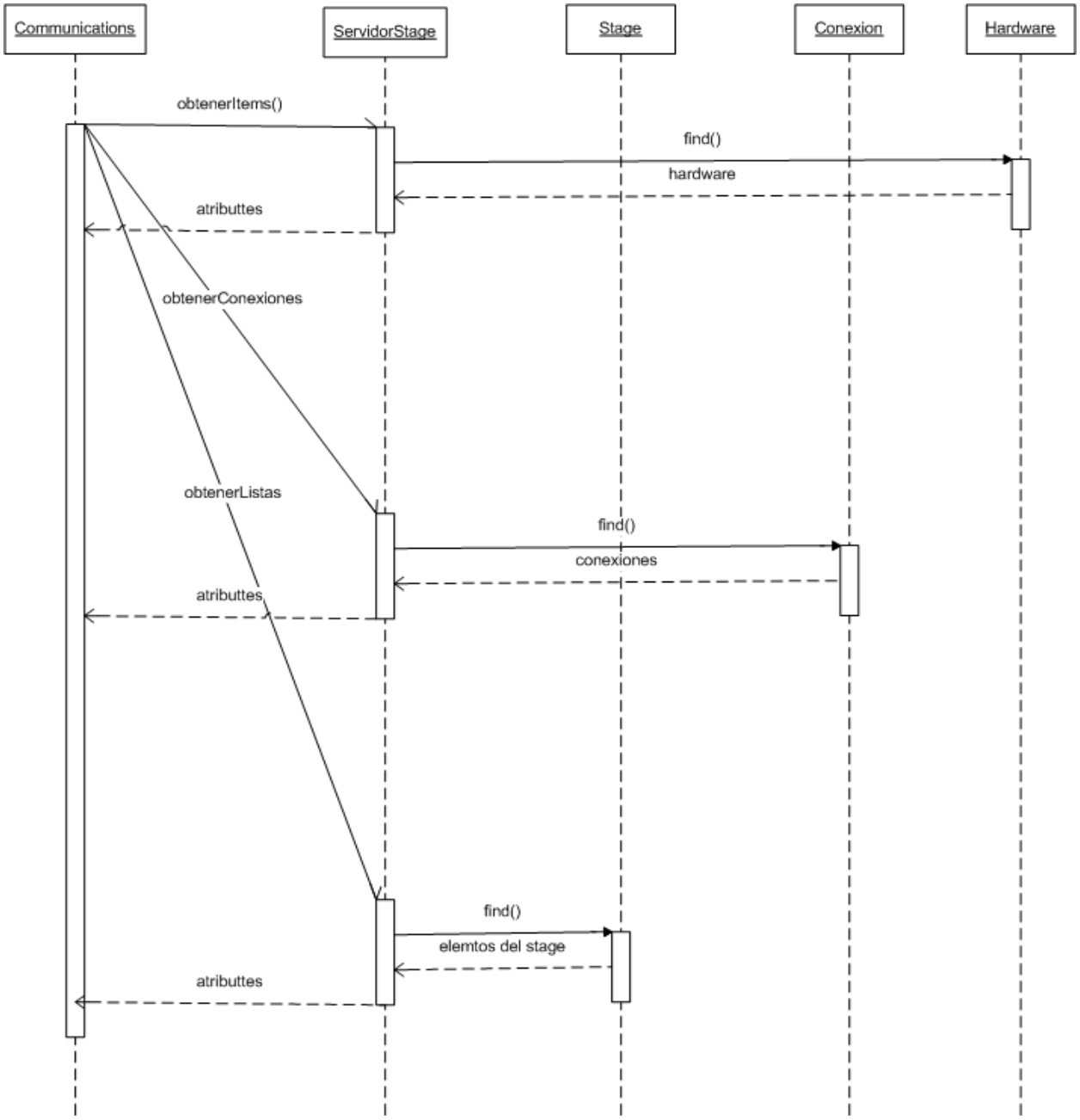

Figura 3.28 Diagrama de secuencia de las comunicaciones AMF

3.2.3. Fase de Construcción

En la presente fase se procede al desarrollo del diseño propuesto en las fases de inicio y elaboración del módulo diseñador de centros de datos. En principio se desarrollaron los modelos necesarios para dar soporte a la persistencia de datos e interacción con la base de datos. Según el esquema planteado en la figura 3.16, el módulo diseñador utiliza como interfaz la aplicación sobre Ruby on Rails para interactuar con la base de datos. Para dicha interacción es necesaria la implementación de dos nuevos modelos para la aplicación, los cuales se agregan a los demás del cargador de datos tal como se plantea en la figura 3.1. Estos modelos servirán de interfaz para interactuar con las estructuras de la figura 3.18 y su implementación se define a continuación. "Stage" provee la persistencia para el área de diseño y "Conexion" para las conexiones entre elementos.

En resumen, el modelo "Stage" se encarga de reflejar el estado del área de diseño en la base de datos una vez que el usuario invoca la funcionalidad de "salvar estado del área de diseño" mientras que el modelo "Conexion" a pesar de que no contiene código alguno, permite que desde los controladores se pueda acceder a los datos que la tabla "conexiones" posee debido a las características que provee ActiveRecord.

Continuando con la construcción del módulo diseñador, se procede a definir el controlador que sirve de interfaz entre Flash y Ruby on Rails como propone la figura 3.16.

- **ServidorStageController:** *Este controlador provee de la interfaz que utilizará la capa de comunicaciones de flash para interactuar con Rails. Provee cuatro funcionalidades principales las cuales se encargan de retornar todos los elementos que se encuentran actualmente en el área de diseño (index), todas las conexiones que hay entre los elementos del área de diseño (conexiones) y el hardware que se listará en el panel lateral izquierdo como*

elementos de diseño disponibles (obtenerHardware) así como proveer la funcionalidad de salvar el estado del área de diseño (salvar_stage). Es importante destacar que estos métodos serán invocados "remotamente" por los servicios de la capa de comunicaciones y que los tipos de retorno son serializados a formato AMF mediante la clase "Rubyamf", definida previamente en el diagrama de clases que se encuentra en la fase de elaboración de esta iteración. La forma de invocación a dicha serialización se define a continuación.

```
@stage = Stage.find(:all).sort_by{|x| x.padre}

respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @stage.to_xml }
      format.amf { render :amf => @stage }
end
```

Donde "@stage" es un objeto de tipo "Stage", el cual es serializado y renderizado mediante "format.amf" dentro del bloque de respuesta de la función ruby.

Una vez culminada la construcción de la plataforma de Rails que dará soporte a la interacción con flash mediante AMF, lo siguiente es continuar con la implementación de las capas del diseñador propuestas en la figura 3.26 y a la clase "DragDrop" de la misma figura.

- **General Actions:** *Esta capa contiene las funciones generales que pertenecen a la lógica general del módulo diseñador. Entre ellas destacan las funciones de recorrido sobre los elementos del área de diseño y las de eliminación de conexiones entre elementos.*

- **Windows:** *Esta capa no contiene código asociado pero alberga el diseño de las ventanas que muestran mensajes e interactúan con el usuario. Los cuatro tipos de ventanas se ilustran en la figura 3.29.*

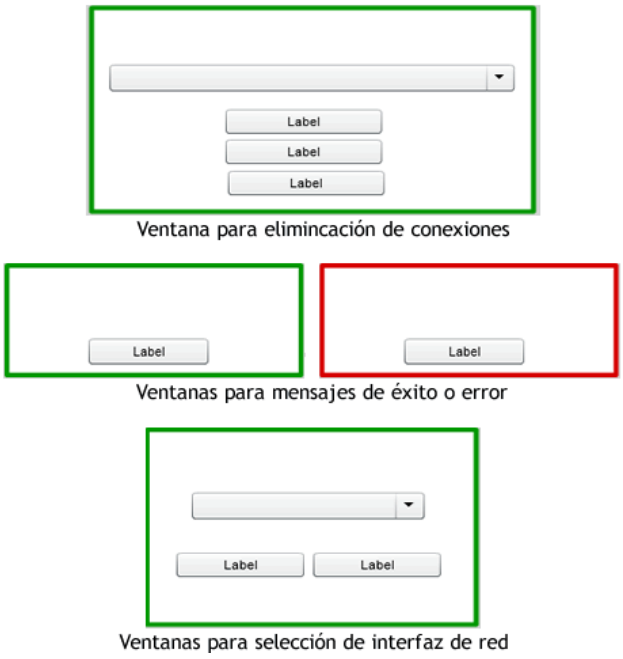

Figura 3.29 Modelo de ventanas emergente del diseñador

- **Communications:** *Esta capa contiene toda la lógica de comunicaciones con el servidor rails, creación de las listas de elementos disponibles y recreación de los elementos en el área de diseño. Esta capa contiene, en principio, la declaración de las variables necesarias para albergar los elementos del área de diseño, así como del panel lateral y otras estructuras auxiliares. Luego de esto, la creación del servicio remoto mediante la clase "RemotingService" y la invocación de los métodos que ahí se encuentran y que fueron descritos previamente en el controlador "ServidorStageController". Para la creación de la lista de elementos disponibles del panel lateral se utiliza la función "obtenerListas" en la cual dependiendo del tipo de símbolo se instancia la clase del ítem que corresponda (Los ítems serán descritos en la capa Items). Es importante destacar que todos los ítems heredan de la clase DragDrop para obtener sus características y funcionalidades. A continuación se muestra la manera de invocar el servicio remoto.*

```
import org.rubyamf.remoting.ssr.*;

var rs:RemotingService;

/*Creacion del servicio remoto*/
rs = new RemotingService("http://DOMINIO:PUERTO/rubyamf/gateway",
"ServidorStageController");
rs.addEventListener(FaultEvent.CONNECTION_ERROR, onConnectFault);
rs.addHeader('recordset_format',false,'fl9');
rs.index([], obtenerItems, onFault);
/*llamadas a los metodos remotos*/
rs.obtenerHardware([], obtenerListas, onFault);
rs.conexiones([], obtenerConexiones, onFault);
```

Luego de que se instancia la clase "RemotingService", se realiza la llamada a los métodos definidos en el servidor como se indicó en el párrafo anterior y los valores de retorno de dichos métodos son capturados como parámetros en las funciones respectivas del parámetro dos de la llamada, es decir, que la invocación a "rs.index" invoca al método "index" del controlador rails mediante AMF y los valores de retorno de dicha función son capturados por "ObternerItems", la cual es la función que se encuentra como segundo parámetro de la invocación. Es necesario destacar que esta invocación se realiza de manera asíncrona, lo cual mejora el rendimiento de la aplicación notablemente, así como también es relevante indicar que esta es solo la lógica de invocación; la lógica de serialización y comunicación se encuentra en el paquete "org.amf.remoting.ssr" que se importa en la primera línea del fragmento de código anterior.

Otro bloque a resaltar es la creación del "slider", el cual le da sentido y funcionalidad de lista al panel lateral. Esta parte de la capa es importante porque ni Flash ni ActionScript dan soporte nativo a paneles deslizantes o "Scrollpanels" para elementos que no sean componentes de texto. Por ende, la lista de elementos, que son un conjunto de MovieClips, no entra en esta categoría y no existe soporte para el Scroll de este tipo. La solución propuesta

para este problema fue utilizar un elemento de la clase "Slider", editar su apariencia y realizar cálculos matemáticos en función de las salidas que este provee a la hora de que se realiza un movimiento del mismo. A continuación se encuentra el fragmento de código que provee dicha solución.

```
var s:Slider = new Slider();
/*colocando items_mc del tamaño adecuado para la cantidad de
elemntos*/
var itemsPerBox:Number = items_mc.height/(itemHeight + espaciado);
var newItemsHeight:Number =
(items_mc.height*cantidadItems)/(itemsPerBox);
if(newItemsHeight > items_mc.height){
  items_mc.height = newItemsHeight - (items_mc.height/2);
  /*SE modifica la escala de los elementos del panel*/
}

var max:Number = items_mc.height-m_height;

/*Si hay suficientes items para hacer scroll*/
  s.maximum=1;
  if(max!=0){
    s.maximum = items_mc.height/10;
  }
  s.direction = SliderDirection.VERTICAL;
  s.rotationY = 180;
  s.name = "slider_mc";
  s.setSize(m_width,m_height-40);
  s.move(m_x + m_width-10,m_y + m_height-20);
  s.liveDragging = true;
  s.addEventListener(SliderEvent.CHANGE, announceChange);
  addChild(s);

/*END SLIDER*/

/*Actualizacion del panel lateral al nuevo valor indicado por el
usuario*/
function announceChange(e:SliderEvent):void {
  items_mc.y = (-e.target.value*10) + factorInicial;
}
```

Por último, otra función a destacar es "recrearElemento", la cual es encargada de recrear los ítems en el área de diseño cuando se carga el módulo diseñador. Esta función obtiene los valores que están guardados en la base de datos y reconstruye el área de

diseño, dejándola como se diseñó originalmente. Las demás funciones siguen la misma línea de las antes explicadas.

- **Buttons:** *Esta capa provee las funcionalidades necesarias para la que el usuario pueda utilizar las herramientas y opciones del módulo. Además del código, contiene todos los elementos como botones, opciones y cursores utilizados en el diseñador. La implementación de esta capa resulta muy engorrosa y con partes de código que a simple vista son ilegibles. Sin embargo, se trató de documentar la mayor cantidad de código para facilitar su lectura. Esta capa cuenta con una implementación de 903 líneas de código que no serán incluidas en el presente documento por motivos de pertinencia. Sin embargo la figura 3.30 muestra los botones que se encuentran en la presente capa divididos en tres grupos (barra de herramientas, opciones y cursores) y que son nombrados a continuación de izquierda a derecha y de arriba hacia abajo según la ilustración 3.30: mouse simple, mano, zoom-in, zoom-out, agregar conexión, eliminar conexiones, eliminar elemento, cursor mano abierta, cursor mano cerrada, cursor zoom-in, cursor zoom-out, cursor agregar conexión, cursor, eliminar conexión, cursor eliminar elemento, opción guardar y opción cerrar sesión.*

Figura 3.30 Elementos de la capa Buttons del diseñador

- **Scrolling:** *Esta capa contiene la lógica de operación del panel lateral izquierdo. Esta se encarga de hacer funcionar el buscador de elementos y los botones arriba y debajo de la barra de desplazamiento. La parte más importante de esta capa es la implementación de la búsqueda sobre los elementos del panel lateral, los cuales son MovieClips y solo son ubicables por sus coordenadas y propiedades. El desarrollo de la búsqueda consta de dos funciones, las cuales se muestran a continuación.*

```
/*search events*/

function buscarPatronEnSidebar(patron:String):Object{
  for each(var it:Object in sidebarList){
    var str:String = it.name.toLowerCase();
   /*Se hace la busqueda por el nombre del item que concuerde con el
   patrón y se retorna dicho item*/
    if (str.search(patron.toLowerCase()) != -1){
      return it;
      break;
    }
  }
  return null;
}

function busqueda(event:KeyboardEvent):void{
  var itTemp:Object = buscarPatronEnSidebar(event.target.text);
  if(itTemp != null){
   /*Si se encuentra algún ítem se actualiza el valor del sidebar*/
    s.value = (itTemp.y*items_mc.scaleY)/10;
    items_mc.y = (-s.value*10) + factorInicial;
  }else{
    trace("no match");
  }
}
```

En resumen, la función "búsqueda" se ejecuta cuando ocurre el evento de presionar una tecla en la barra de búsqueda del diseñador, luego se procesa la búsqueda en la función "buscarPatronEnSidebar", buscando el elemento o MovieClip que coincida con lo que introdujo el usuario y por último se obtienen las

coordenadas donde se encuentra dicho elemento para modificar los valores del scroll y posición del panel lateral en función de estas.

- **Items:** *Esta capa contiene todos los elementos y clases disponibles para el diseño del centro de datos. Para esta capa se realizó el diseño de los elementos con la herramienta de diseño Swift 3D como se indicó en la fase de elaboración. La figura 3.31 muestra el resultado del diseño de los tipos de elementos que pueden ser instanciados y utilizados para el diseño del centro de datos (De izquierda a derecha: Torres de comunicaciones con switches y routers dentro, torres de servidores con servidores corredizos dentro, torres de PDU con PDUs dentro, mesas, vidrios y paredes).*

Figura 3.31 Tipos de elementos disponibles para el diseño

- **Layouts:** *Esta capa contiene los gráficos de la diagramación del módulo diseñador tal cual como se muestra en la figura 3.25.*
- **Main Stage:** *Esta capa contiene los gráficos del área de diseño.*

- **DragDrop:** *Esta clase contiene la lógica interna de cada elemento instanciado en módulo diseñador. Lógicamente, cada elemento hereda de la clase DragDrop para poder adquirir los atributos y métodos que en esta se encuentran. La clase DragDrop contiene toda la lógica para las funcionalidades de arrastrar y soltar elementos al área de diseño, agregarlos a la misma y todas las operaciones y atributos que lógicamente contiene un elemento del centro de datos. De esta lógica es importante destacar tres funciones en particular; la primera es la función "Drag", la cual se ejecuta cuando se dispara un evento de "MouseDown", el cual consiste en presionar el botón principal del mouse, y es la encargada de la lógica necesaria para mover un elemento de un lado a otro. Es relevante mencionar que por cada elemento, al heredar de esta clase, ya provee sus atributos, métodos y el manejo de eventos se realiza sobre cada uno de ellos. La siguiente función a destacar es "Drop", la cual se ejecuta cuando ocurre un evento de "MouseUp" sobre un elemento, el cual consiste en soltar el botón primario del mouse. Ambas funciones son el núcleo de la clase y están compuestas de lógica de programación optimizada y validaciones que permiten su correcto funcionamiento. Por último, la función de creación de elementos completa el núcleo principal de esta clase; dicha función se encarga de la creación de los elementos en el área de diseño, una vez que se ejecuta la función "Drop". Esta función recrea la instancia del MovieClip que es arrastrado desde el panel lateral en el área de diseño y le asigna valores a sus propiedades. A continuación se muestra la implementación de dicha función.*

```
private function crearElemento(nombreClase:String,
destiny:MovieClip):MovieClip{
```

```
        var definicionClase:Class =
        Class(getDefinitionByName(nombreClase));

        /*Se obtiene la definicion de la clase del padre y se
        instancia dinámicamente el nuevo elemento*/
        var item:MovieClip = new definicionClase();
                ...
          /*Aqui se actualizan las propiedades del elemento*/
                ...
        var now:Date = new Date();
        item.idElem = now.getTime();/*id del elemento*/

                ...
          /*Aqui se asignan los puertos libres del elemento*/
                ...

        item.addEventListener(MouseEvent.MOUSE_OVER, showTooltip);

                ...
          /*Aqui se agrega el elemento en su contenedor destino*/
                ...

        destiny.addChild(item);
        return item;
    }
```

Como se menciono anteriormente, para optimizar el desarrollo, se organizó el código en capas, de tal manera que cada capa actuara como un objeto, en donde tuviera sus propios atributos y métodos y que interactuaran entre sí cuando se crucen lógicas de aplicación que requieran cooperación de varios elementos del módulo. Por esta razón, es posible ejemplificar las capas como clases en un diagrama de clases y de esta manera comprender mejor su funcionamiento. Es importante recalcar que las capas no son objetos ni clases dentro del lenguaje de programación, no obstante, lógicamente si se comportan como clases. Dicho diagrama de clases se encuentra en la figura 3.27 de la fase de elaboración.

3.3. Iteración III

En la tercera iteración se realizó una revisión a las primeras dos iteraciones, correspondientes al desarrollo del sistema cargador y diseñador de centros de datos. El motivo por el cual se realiza esta iteración es para

asegurar el correcto funcionamiento de los dos módulos más grandes del sistema.

Es importante destacar que en esta iteración, por tratarse de modificaciones a iteraciones anteriores, no se requiere de un levantamiento de requerimientos como tal, ya que en las iteraciones uno y dos ya se realizaron los levantamientos de requerimientos respectivos a los módulos cargador y diseñador. Por tal razón, la presente iteración consta sólo de fase de elaboración y fase de construcción, de acuerdo con la adaptación realizada en la metodología de desarrollo.

3.3.1. Fase de Elaboración

En la presente fase se realiza el análisis y diseño de las modificaciones requeridas para el módulo cargador de datos y para el módulo diseñador de centros de datos (iteraciones uno y dos respectivamente).

El primer análisis está orientado al modelo de datos, el cual debe dar soporte a las nuevas funcionalidades requeridas en los módulos antes mencionados. Comenzando con el sistema cargador de datos, este requiere de modificaciones que complementen ciertas funcionalidades del sistema que ya se encuentran desarrolladas. La primera es el complemento de la funcionalidad "agregar_elemento", mediante la cual se incorporan elementos al sistema; Un nuevo análisis indica que el módulo cargador no provee soporte en caso de que un elemento sea desincorporado del centro de datos, lo cual nos lleva a requerir una nueva funcionalidad, "eliminar_elemento". Para esta nueva funcionalidad no se requiere realizar modificaciones sobre el modelo de datos, ya que mediante ella se eliminan entradas de la estructura antes diseñada.

Continuando con el análisis, se encontró que el soporte para la agregación de características dinámicamente a los elementos no está

completo, ya que se requiere de la interfaz necesaria para la agregación de características. Para esta funcionalidad tampoco se requieren modificaciones sobre el modelo de datos.

Una vez analizado el modelo de datos del cargador, se realizó el análisis del modelo del diseñador, en el cual tampoco presenta cambios de estructura, por lo que las dos nuevas funcionalidades requeridas en el diseñador no lo requieren.

El segundo análisis se realizó sobre los diagramas de caso de uso del sistema cargador, los cuales fueron modificados para dar soporte a las dos nuevas funcionalidades que se deben adicionar. A continuación se presentan los casos de uso de nivel 1 del cargador de datos, con las nuevas funcionalidades incluidas.

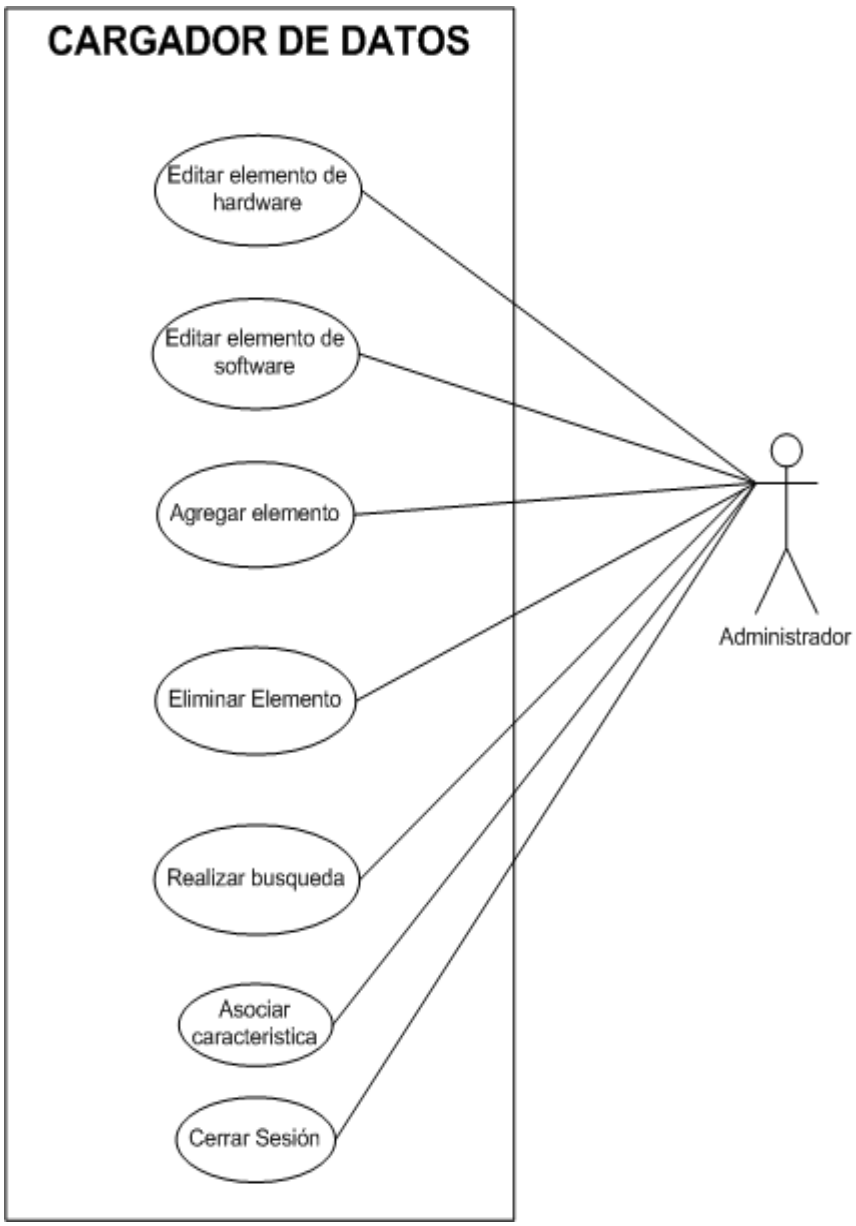

Figura 3.32 Diagrama final de casos de uso del cargador– Nivel 1

En la figura 3.32 se muestra el diagrama final de casos de uso del cargador de datos, donde se puede apreciar que se agregaron las funcionalidades "eliminar_elemento" y "asociar_caracteristica". Asimismo a continuación se presenta la descripción de dichos casos de uso.

- **Eliminar elemento:** *Permite al usuario eliminar un elemento del sistema.*

  Pre-condición: *El usuario debe haber agregado previamente el elemento.*

  Post-condición: *Los cambios realizados por el usuario deben reflejarse en el sistema.*

- **Asociar característica:** *Permite al usuario asociar una característica a un elemento de hardware o software.*

  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento y haberlo seleccionado.*

  Post-condición: *La característica debe adjuntarse al elemento seleccionado.*

El último análisis de la presente iteración se realizó sobre las funcionalidades y casos de uso del módulo diseñador. A dicho diagrama se le adicionaron dos nuevas funcionalidades en el nivel uno. El análisis indicó que es necesario reflejar en el área de diseño la capacidad que tienen ciertos elementos de hardware de variar en su cantidad de interfaces de red, a pesar de ser el mismo modelo, es decir, es importante poder agregar nuevas interfaces de red a los elementos del área de diseño. Asimismo, es importante contar con una funcionalidad complemento que permita quitar interfaces de red a dichos elementos. Todo esto con la finalidad de asemejar el diseño del centro de datos lo más q se pueda a la realidad. El diagrama final de casos de uso de nivel uno del diseñador de centros de datos se encuentra a continuación.

Figura 3.33 Diagrama final de casos de uso del diseñador – Nivel 1

Asimismo la descripción de las nuevas funcionalidades incluidas en el módulo se encuentra a continuación.

- **Agregar interfaz de red:** *Permite al usuario adicionar una interfaz de red o puerto a algún elemento del área de diseño.*
- **Pre-condición:** *El usuario debe haber agregado previamente al menos un elemento al área de diseño.*
- **Post-condición:** *Se debe adicionar un puerto disponible al elemento y mostrar un mensaje de éxito.*

- **Eliminar interfaz de red:** *Permite al usuario sustraer la última interfaz de red o puerto de algún elemento del área de diseño.*
- **Pre-condición:** *El usuario debe haber agregado previamente al menos un elemento al área de diseño y este elemento debe poseer la última interfaz de red disponible.*
- **Post-condición:** *Se debe sustraer la interfaz de red o mostrar un mensaje de error en caso de que el procedimiento no pueda realizarse.*

Por último, se realizaron modificaciones al diagrama de clases del módulo cargador de datos (descrito previamente en la fase de elaboración de la iteración uno), específicamente a las clases "AgregarElementoMenu", "ActualizadorAjaxHardware" y "ActualizadorAjaxsoftware" para agregar las funcionalidades antes mencionadas en los casos de uso.

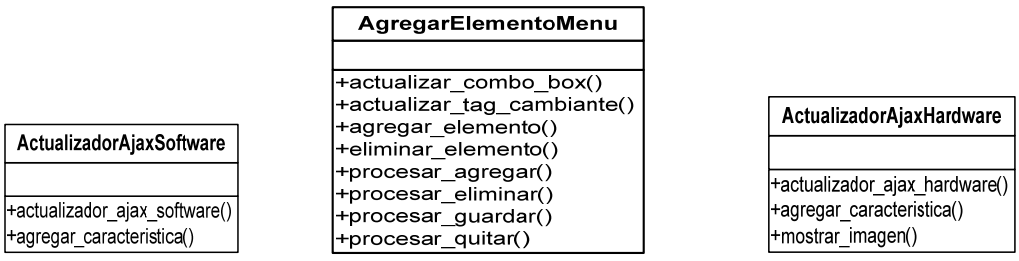

Figura 3.34 Modificaciones sobre las clases de la iteración dos

3.3.2. Fase de Construcción

En la presente fase se procede al desarrollo del diseño propuesto en la fase elaboración, es decir, al desarrollo de las nuevas funcionalidades e interfaces de los módulos cargador de datos y diseñador de centros de datos.

Por no haber modificaciones en el modelo de datos, en esta iteración no se desarrollaron nuevos modelos en la aplicación. No obstante, se agregó el desarrollo de las funcionalidades propuestas en la fase de elaboración a los controladores previamente creados en las iteraciones uno y dos, así como se realizaron las modificaciones necesarias a la interfaz de usuario que dirige al uso de dichas funcionalidades.

Comenzando por el cargador de datos, la funcionalidad de "eliminar_elemento" se adicionó al controlador "AgregarElementoMenu", en donde, para dar soporte completo, se adicionaron un par de funciones llamadas "eliminar_elemento" y "procesar_elemento" que se encargan de realizar la lógica descrita en el caso de uso final de la fase de elaboración.

Seguidamente, en los controladores "ActualizadorAjaxHardware" y "ActualizadorAjaxSoftware" se agregaron las funciones necesarias para dar soporte al caso de uso de "asociar_caracteristica". Ambas funciones ejecutan una lógica parecida pero sobre elementos distintos del centro, es decir, unas agregan características a los elementos de software y otras a los elementos de hardware; por ello el motivo de la separación en dos partes.

Por último, en el cargador de datos, se realizaron las modificaciones a la interfaz para agregar acceso a las nuevas funcionalidades del sistema.

Una vez modificado el cargador de datos se procedió a realizar las inclusiones necesarias en el módulo diseñador. Para esto hizo falta agregar la lógica necesaria dentro de la capa "Buttons" que se encargara de

implementar los manejadores de eventos y funciones necesarias para la adición y sustracción de interfaces de red a los elementos.

La interfaces de usuario añadida para el uso de las funcionalidades de agregar y eliminar interfaces de red fueron dos entradas en la barra de herramientas del diseñador, así como dos cursores adicionales. Tanto las entradas como los cursores se muestran en la figura 3.35.

Figura 3.35 Interfaz de nuevas funciones en el diseñador

3.4. Iteración IV

Durante la cuarta iteración se realizó el análisis, diseño y desarrollo del módulo visualizador del sistema, el cual permite ver el producto de las actividades realizadas con el módulo cargador y el módulo diseñador. Dicho producto es la organización de todos los elementos del centro de datos, sus características e interconexiones. Este módulo fue integrado en el sistema para que interactuara con el resultado de las dos primeras iteraciones. Asimismo, se refinó el modelo de datos del sistema y se hicieron revisiones sobre el resultado de las iteraciones uno y dos. Por último, en esta iteración se realizó la interfaz general del sistema, la cual permite al usuario autenticarse y elegir entre las opciones que desea realizar en el sistema, así como todas las validaciones de seguridad necesarias en la aplicación. Cabe destacar que se realizaron pruebas de integración sobre toda la aplicación,

resultado de la integración de las tres primeras iteraciones de la metodología de desarrollo.

3.4.1. Fase de Iniciación

En esta fase se realizó un análisis de las iteraciones anteriores para facilitar la integración con la iteración actual. Así mismo, se realizaron las tareas necesarias de levantamiento de información y definición de requerimientos necesarios para el desarrollo del sistema visualizador de centros de datos. Además de esto, se realizó revisión adicional a todos los módulos del sistema, integrados y funcionales con el fin de garantizar el funcionamiento correcto del sistema final.

La recopilación de requerimientos de software necesarios para sistema visualizador y funcionalidades añadidas de autenticación, agregación de usuarios y cambio de contraseñas en esta fase fueron los siguientes:

- *Interfaz que permita al usuario seleccionar los elementos del centro de datos para visualizar sus características y conexiones.*
- *Se deben poder visualizar también los elementos anidados dentro de otro elemento.*
- *Interfaz con caja de autenticación donde el usuario pueda ingresar su nombre de usuario y contraseña para ingresar al sistema.*
- *Interfaz que liste todas las funcionalidades del sistema, donde el usuario pueda seleccionar el módulo al cual desea acceder o la funcionalidad que desea utilizar.*
- *Funcionalidades de agregar usuarios, donde un usuario de rol administrador pueda agregar otros usuarios al sistema, con su respectivo nombre de usuario y contraseña.*
- *Funcionalidad de cambiar contraseña donde el usuario, una vez habiendo ingresado al sistema, puede introducir su contraseña*

anterior y una nueva contraseña de acceso al sistema para reemplazarla la anterior.

- *Agregar seguridad al almacenamiento de contraseñas mediante la encriptación de las contraseñas de usuario en la base de datos.*
- *Interfaz que provee información general acerca del sistema.*
- *Manejo de roles de usuario en la aplicación que contemple dos tipos de usuario: Administrador y usuario común. Donde el administrador podrá realizar todas las operaciones del sistema mientras el usuario común solo podrá utilizar el módulo visualizador y cambiar su contraseña en el sistema.*

3.4.2. Fase de Elaboración

En esta cuarta iteración se desarrolló el módulo visualizador, así como revisiones de las primeras dos iteraciones y funcionalidades extras para unificar los tres grandes módulos del sistema en una misma aplicación. Esta fase se dedica al diseño de la arquitectura del visualizador el cual, al igual que el diseñador, está implementado en lenguaje ActionScript con el esquema de comunicaciones sugerido en la figura 3.13, el cual plantea una serialización de los datos a comunicar mediante AMF y el intercambio de datos binarios entre el servidor de aplicaciones en Rails y el documento SWF de Flash.

Las ventajas a obtener de ActionScript son las mismas descritas en la fase de elaboración de la iteración dos para el módulo diseñador de centros de datos y que se resumen a las capacidades gráficas que posee el lenguaje mencionado y la suite de diseño CS4 de Adobe.

Adicionalmente se integró el módulo de autenticación y una interfaz principal donde el usuario podrá seleccionar el módulo que desea utilizar además de contar con ciertas funcionalidades extra. El desarrollo de esta

última parte se realizo bajo el esquema MVC descrito en la iteración uno y se ilustra en la figura 3.1.

Respecto al diseño del modelo de datos para esta iteración, al módulo visualizador no se le agregó ninguna estructura, relación o atributo adicional al adicionado en la iteración dos. Debido a que la función del visualizador es obtener los datos de los elementos del área de diseño y recrearlos para poder ser seleccionados y visualizados en detalle, este módulo utiliza los datos colocados por el módulo diseñador de centros de datos en las estructuras de la figura 3.18. Es de notar que las operaciones del visualizador son de solo lectura contra la base de datos, es decir, que las estructuras de la figura 3.18 funcionan como una interfaz de comunicación entre el diseñador y el visualizador, permitiendo así que cuando se guarden cambios realizados en el módulo diseñador podrán reflejarse inmediatamente en el módulo visualizador.

Para el módulo de autenticación y funcionalidades adicionales como agregar usuarios y cambiar contraseña, se incluyeron nuevas estructuras al modelo de datos que dieran soporte a estas funcionalidades, así como también a la funcionalidad de registro de las actividades en una bitácora. Para el manejo de roles y usuarios se incluyeron cuatro nuevas estructuras, las cuales son necesarias para dar soporte a esta característica. La figura 3.35 ilustra las nuevas estructuras, comenzando por "usuarios", la cual almacena los usuarios del sistema con su nombre de usuario, contraseña y el rol que tiene asociado. Seguidamente, se puede apreciar la relación directa con la estructura "roles", la cual almacena todos los roles del sistema. Las estructuras "funcionalidades" y "funcionalidad_en_rol" contemplan que las funcionalidades asociadas a un rol sean dinámicas, es decir, que en la aplicación se puedan desplegar dinámicamente las funcionalidades asociadas a un rol directamente desde la base de datos. Adicionalmente, se agregó la estructura "bitacora", la

cual permite llevar un seguimiento de las acciones que se realizan en el sistema, tal como la utilización de cierto módulo, creación de usuarios, modificaciones al área de diseño, etc.

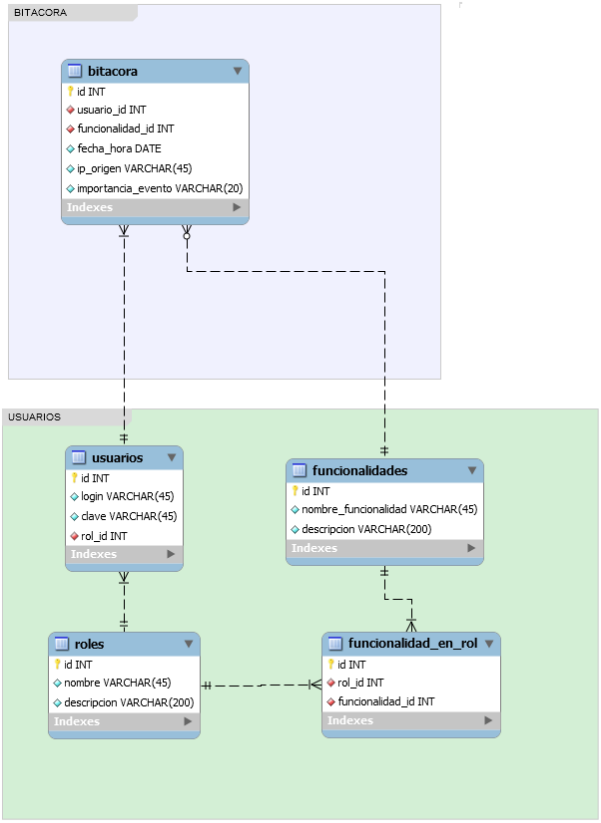

Figura 3.35 Diagrama del modelo de datos iteración 3

Una vez obtenido el modelo de datos del sistema cargador de datos, del sistema diseñador, del sistema visualizador y de los módulos complementarios ya es posible acoplar todos estos en un único modelo de datos que pasará a ser el modelo final de la aplicación, la cual lleva por nombre DatDesigner. El mencionado modelo de datos se ilustra en la figura 3.36, donde podemos apreciar el acoplamiento de los diagramas de las primeras cuatro iteraciones.

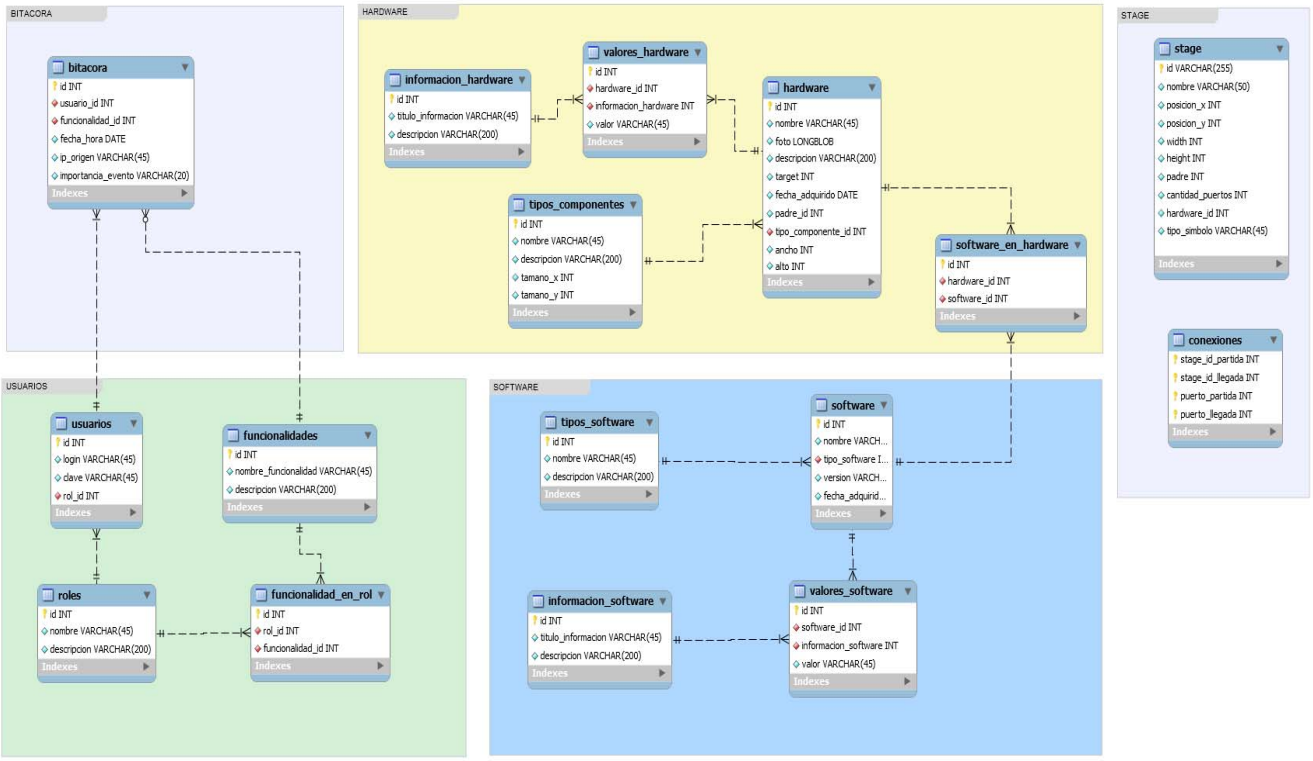

Figura 3.36 Modelo de datos del sistema

Posteriormente, se definieron y diseñaron los diagramas de casos de uso que permitirán reflejar las interacciones entre los usuarios y las funcionalidades del módulo visualizador. A continuación se presentan los casos de uso para el módulo visualizador de centros de datos divididos en dos niveles (esto debido a que los casos de uso de nivel uno ya son lo suficientemente específicos y no están compuestos por otros casos de uso adicionales):

- **Nivel 0:** *Muestra de una manera general la interacción de usuario con el sistema. Los casos de uso de nivel 0 del visualizador se muestran en la figura 3.37.*

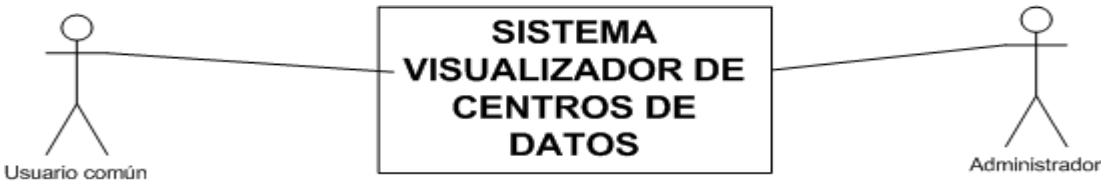

Figura 3.37 Diagrama de casos de uso del visualizador – Nivel 0

- **Nivel 1:** *En este nivel podemos apreciar mayor detalle, ya que se muestra qué funcionalidades existen dentro del módulo visualizador, las cuales son muy precisas. Los casos de uso de nivel 1 se muestran en la figura 3.38.*

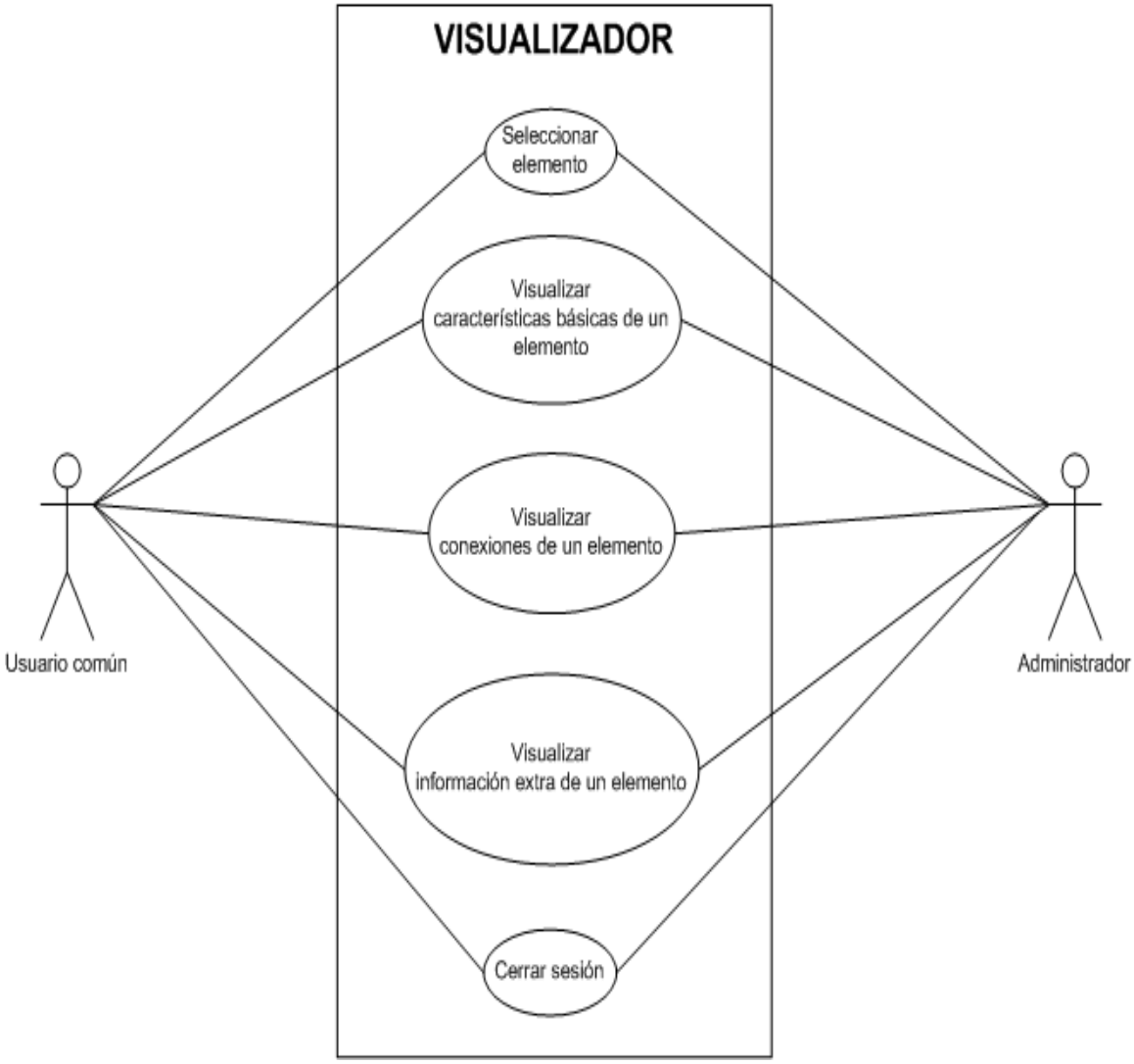

Figura 3.38 Diagrama de casos de uso del visualizador – Nivel 1

Además del diagrama de casos de uso de la figura 3.38, a continuación se describen cada uno de los casos de uso que en él se encuentran y no fueron descritos anteriormente:

- **Seleccionar elemento:** *Permite al usuario seleccionar un elemento del área de diseño haciendo click.*

  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento mediante el diseñador.*

  Post-condición: *Se debe seleccionar el elemento y mostrarse en un segundo plano individual con sus propiedades.*

- **Visualizar características básicas de un elemento:** *Permite al usuario ver las características de un elemento seleccionado.*

  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento mediante el diseñador.*

  Post-condición: *Se deben mostrar las características básicas del elemento seleccionado.*

- **Visualizar conexiones de un elemento:** *Permite al usuario ver las conexiones de un elemento seleccionado.*

  Pre-condición: *El usuario debe haber agregado previamente al menos un elemento mediante el diseñador.*

  Post-condición: *Se deben mostrar las conexiones del elemento seleccionado.*

- **Visualizar información extra de un elemento:** *Permite al usuario ver la información extra de un elemento seleccionado.*

Pre-condición: *El usuario debe haber agregado previamente al menos un elemento mediante el diseñador.*

Post-condición: *Se debe mostrar la información extra del elemento seleccionado.*

Como ya se mencionó, los casos de uso de nivel 1 del módulo visualizador son bien específicos y no están compuestos de otros casos de uso, por lo que no se requiere de otro nivel de refinamiento de casos de uso. Cabe destacar que hasta este punto ya se tienen los casos de uso de los tres módulos más importantes del sistema, pero aún no se han considerado las funcionalidades de autenticación, agregar usuario y cambiar contraseña del sistema y que deben ser contempladas en la presente fase de la presente iteración. Es por esto que a continuación se presentan los casos de uso generales del sistema DatDesigner organizados en dos niveles:

- **Nivel 0:** *Los casos de uso de nivel 0 del sistema DatDesigner se muestran en la figura 3.39.*

Figura 3.39 Diagrama de casos de uso del sistema DatDesigner – Nivel 0

- **Nivel 1:** *Los casos de uso de nivel 1 se muestran en la figura 3.40. Es importante señalar que en dicha figura se contemplan tres casos de uso que en realidad son una representación lógica de los casos de uso respectivos de cada módulo, es decir, el caso de uso "utilizar cargador de datos" refiere directamente a los casos de usos del*

módulo cargador de datos, por lo cual no necesitan una definición en la presente iteración. Asimismo ocurre con los casos de uso "Utilizar diseñador de centros de datos" y "Utilizar visualizador de centros de datos".

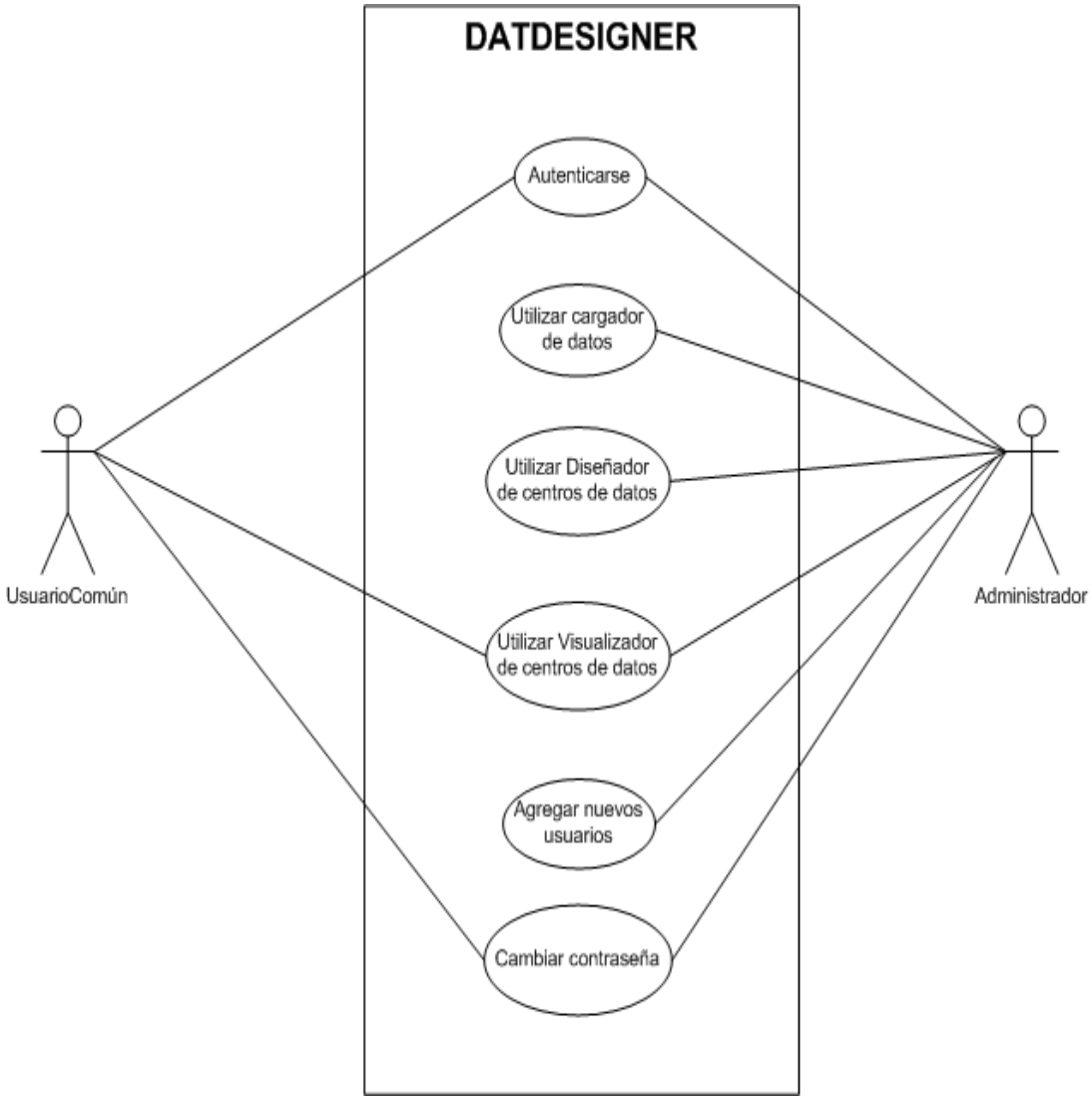

Figura 3.40 Diagrama de casos de uso del sistema DatDesigner – Nivel 1

Además del diagrama de casos de uso de la figura 3.40, a continuación se describen cada uno de los casos de uso que en él se encuentran y no fueron descritos anteriormente:

- **Autenticarse:** *Permite al usuario ingresar sus datos de usuario y contraseña para poder ingresar al sistema de manera segura.*
  Pre-condición: *El usuario debe haber sido registrado por un administrador.*
  Post-condición: *Se debe permitir al usuario el ingreso al sistema.*

- **Agregar nuevos usuarios:** *Permite al usuario agregar nuevos usuarios al sistema.*
  Pre-condición: *Ninguna.*
  Post-condición: *Se debe agregar el nuevo usuario al sistema con su respectivo nombre de usuario, rol y contraseña.*

- **Cambiar contraseña:** *Permite a un usuario cambiar su contraseña.*
  Pre-condición: *El usuario debe estar registrado y haber ingresado al sistema.*
  Post-condición: *Se debe cambiar la contraseña del usuario.*

Una vez definidos los casos de uso del visualizador y para las funcionalidades generales de la aplicación y una vez culminado también el modelo de datos de dichas funcionalidades, se presentan los diagramas de clases del visualizador y de las funcionalidades generales en esta iteración. Es importante destacar que no existe un diagrama de clases para todo el modulo visualizador, ya que su funcionamiento consta de funciones adicionadas a clases previamente definidas. La parte más importante a destacar es el funcionamiento de la clase "ItemVisualizar", de la cual

heredan todos los elementos lógicos del centro de datos. A continuación se muestra el diagrama de dicha clase y sus subelementos.

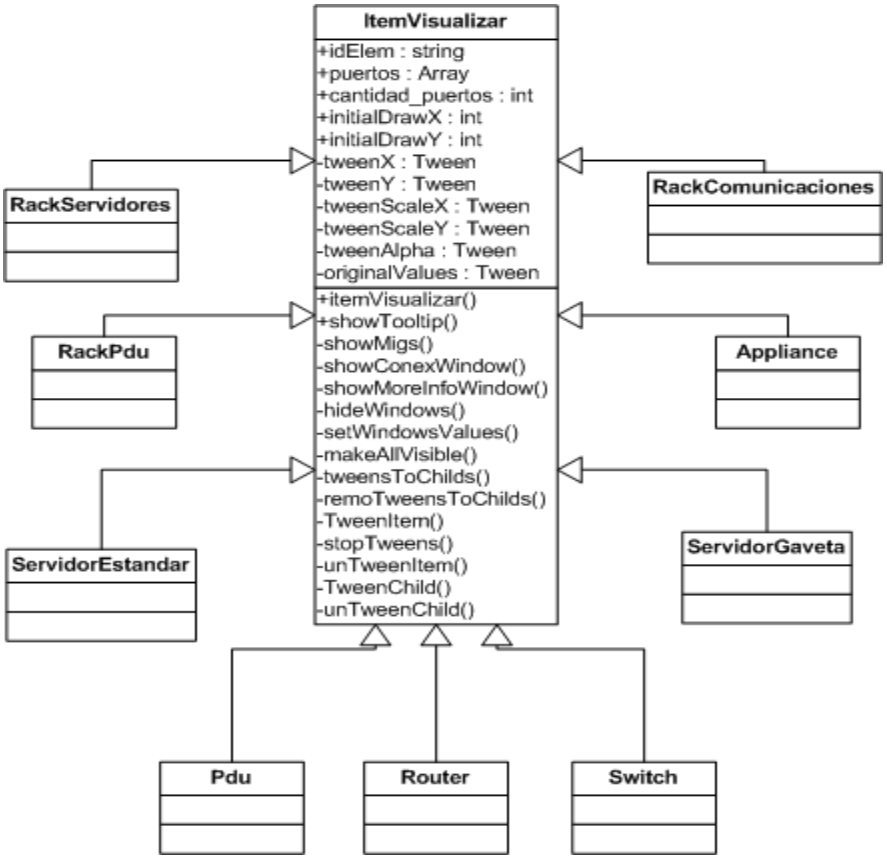

Figura 3.41 Diagrama de clases del visualizador de centros de datos

Es importante destacar que la clase "ItemVisualizar" y sus métodos serán mejor explicados en la fase de construcción. Asimismo, el diagrama de clases de las funcionalidades generales se muestra a continuación. Dichas clases serán explicadas en detalle posteriormente en la fase de elaboración de la iteración actual.

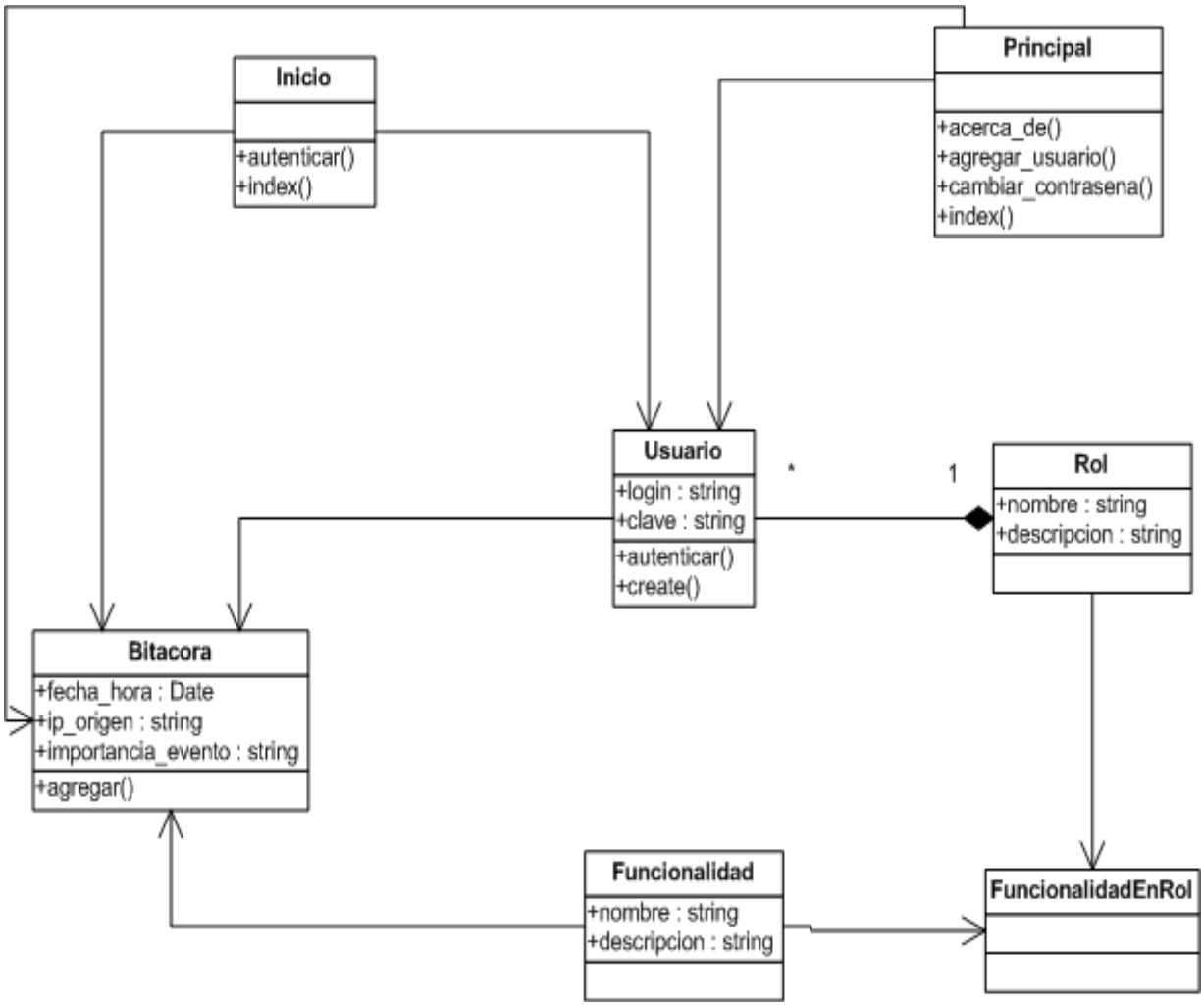

Figura 3.42 Diagrama de clases de las funcionalidades generales del sistema

Luego de esto, se procede al diseño de las interfaces requeridas para el visualizador de centros de datos. Manteniendo una sincronía con las interfaces del los módulos cargador y diseñador, la interfaz del visualizador utiliza la misma línea de colores y consta solamente del área de diseño donde están dispuestos los elementos colocados mediante el módulo diseñador. La figura 3.43 muestra la interfaz inicial del módulo visualizador de centros de datos, la cual refleja la diagramación y algunos elementos para ilustrar la disposición de los mismos en el área de diseño mostrada por el visualizador.

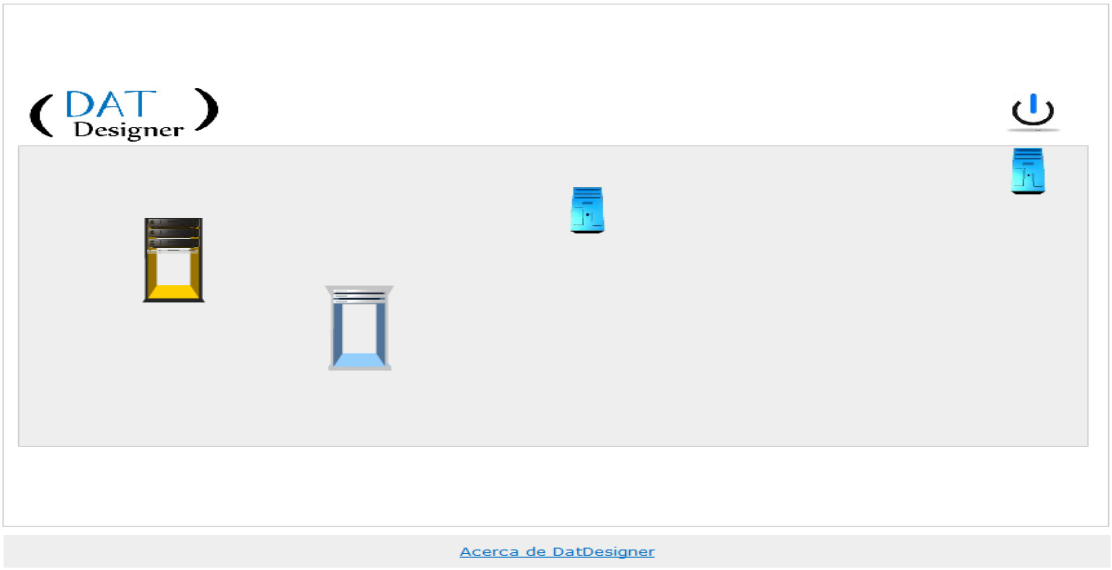

Figura 3.43 Interfaz de usuario inicial del visualizador

Adicionalmente, se define un diagrama de secuencia que ilustra la visualización de características y conexiones de un elemento que pueden realizarse mediante la interfaz definida previamente.

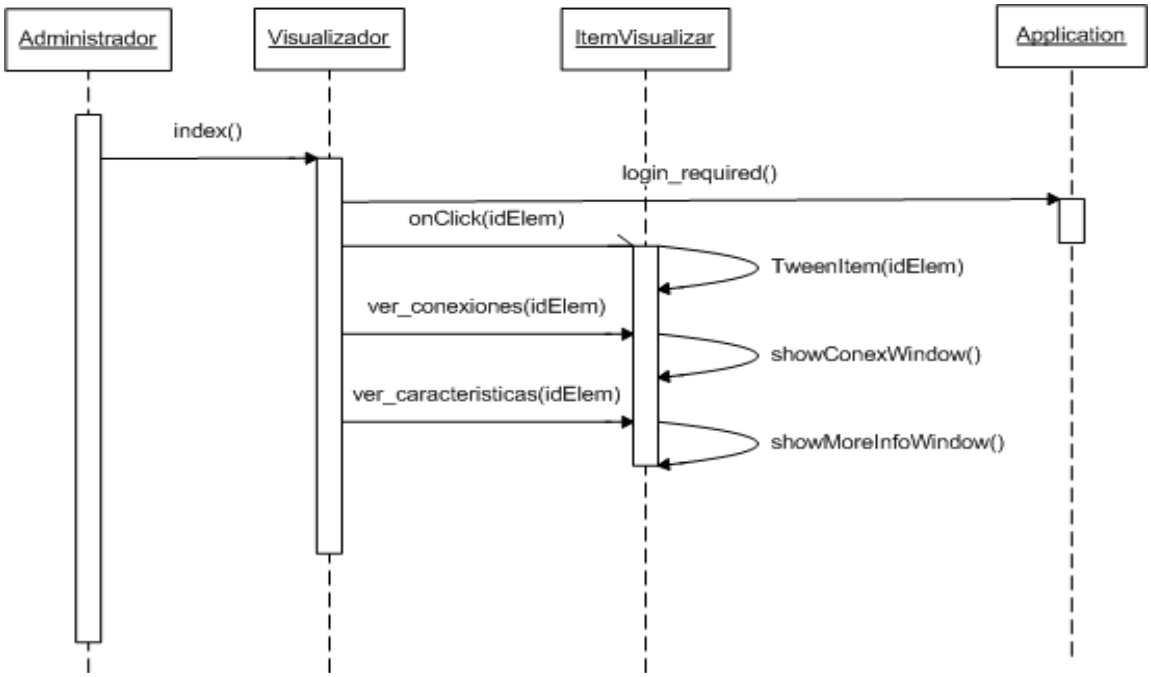

Figura 3.44 Diagrama de secuencia para visualizar las características y conexiones de un elemento mediante el módulo visualizador

Igualmente se realizó el diseño de las interfaces para el módulo de autenticación, la cual será la pantalla de entrada al sistema, contentiva de una solicitud de datos para autenticación y la respectiva identificación del sistema, y para la vista inicial, la cual una vez el autenticado el usuario, puede elegir el módulo o funcionalidad que desea utilizar. Las figuras 3.45 y 3.46 muestran respectivamente la interfaz de autenticación y la vista principal de un administrador.

Figura 3.45 Interfaz de usuario para autenticación

Figura 3.46 Interfaz de usuario de la vista principal

3.4.3. Fase de Construcción

En la presente fase se procede al desarrollo del diseño propuesto en las fases de inicio y elaboración del módulo visualizador de centros de datos, autenticación y demás funcionalidades del sistema general. Debido a que el módulo visualizador trabaja sobre las estructuras "stage" y "conexiones" propuestas en la figura 3.18, éste utilizará los mismos modelos que el módulo diseñador para extraer información de dichas tablas (Stage y Conexion). Sin embargo, las necesidades de este módulo obligan a la inclusión de nuevos métodos en el controlador que realiza la interfaz de comunicaciones (ServidorStageController). En principio, el módulo visualizador utiliza de interfaz la aplicación sobre Ruby on Rails planteada en la figura 3.16 para interactuar con la base de datos y se agregaron las siguientes dos funcionalidades al controlador de comunicaciones para dar soporte a esta comunicación:

```
def obtener_hardware_from_stage
    #Se busca en la base de datos los elementos del area de diseño
    #Se procesan y se les da formato
    #Se envían via AMF
```

```
end

def obtener_more_info_from_stage
    #Se obtiene toda la información relacionada con los elementos del area
    #de diseño mediante las distintas estructuras en donde se encuentra
    #dicha información

    #Se procesa y se envían via AMF

 end
```

La función "obtener_hardware_from_stage" se encarga de obtener la información como hardware de los elementos que se encuentran en el área de diseño. Es importante destacar que la estructura "stage" no almacena información sobre las características de los elementos de hardware que son necesarias para el visualizador. Asimismo, el objetivo de la función "obtener_more_info_from_stage" se encarga de obtener la información adicional asociada al hardware que no se encuentra almacenada en la propia estructura hardware.

Seguidamente se encuentra la implementación en ActionScript del visualizador, la cual se divide en dos componentes principales: a) El código asociado a las comunicaciones y creación de elementos en el área de diseño, el cual es sigue la misma línea de la implementación de estas funcionalidades en el módulo diseñador y b) el código de la clase "ItemVisualizar" de la cual heredan todas las instancias de los elementos del área de diseño y que contiene la implementación de los efectos gráficos sobre dichos elementos.

Una vez que se tiene la creación de los elementos en el área de diseño se puede apreciar mejor mediante el diagrama de clases de la figura 3.41 de la fase de elaboración, la implementación interna de cada elemento en la clase "ItemVisualizar" de cual heredan cada uno de estos elementos.

De manera resumida, en el diagrama de la figura 3.41 se manejan una serie de eventos donde se aplican efectos llamados "Tween" a los

ítems del área de diseño. Estos efectos son activados o desactivados al hacer click sobre un elemento del área. Además, dicha serie de eventos están divididos en dos grupos: a) Los efectos que aplican a los elementos de nivel cero, que sería aquellos que no son contenidos por ningún otro elemento y b) los efectos que afectan a los elementos de nivel uno, que serían aquellos elementos contenidos en otros elementos. El efecto "Tween" consiste en una animación que opaca y desaparece todos los elementos del área de diseño salvo aquel que el usuario ha seleccionado, con la finalidad de mostrarle solo la información de dicho elemento. Adicional a los efectos "Tween" se pueden apreciar eventos sobre las ventanas, que muestran la información de los elementos y eventos que despliegan dicha información. Un flujo de visualización se muestra en la figura 3.47 donde se puede apreciar cada interfaz que recibe el usuario al interactuar con un elemento en el visualizador. Por último para integrar los archivos SWF en la aplicación Rails se utilizó el plugin "swf_fu" que habilita un helper que permite integrar fácilmente en una vista un documento SWF mediante el comando "<%= swf_tag 'visualizador'%>" donde "visualizador" es el nombre del archivo. Igualmente se realizó para el diseñador ("<%= swf_tag 'disenador'%>").

Figura 3.47 Flujo de ejecución del módulo visualizador

Una vez culminado el visualizador ya se tienen los tres grandes módulos de la aplicación DatDesigner y hasta ahora integrados bajo la misma aplicación Ruby on Rails. Lo siguiente a desarrollar del sistema es el módulo de autenticación con manejo de roles y usuarios y las funcionalidades adicionales como agregación de nuevos usuarios, cambio de contraseñas y bitácora. Para dar soporte a la autenticación y al manejo de roles y usuarios se definieron nuevos modelos en la aplicación. Son cinco en total que se encargan de proveer la interfaz necesaria para la interacción de esta parte de la aplicación con la base de datos y se listan a continuación.

- **Usuario**
- **Rol**
- **Funcionalidad**
- **FuncionalidadEnRol**
- **Bitacora:**

Dichos modelos están definidos en el diagrama de clases de la fase de elaboración de la presente iteración. Posteriormente, se agregaron los controladores necesarios para contener la lógica entre la vista de inicio (figura 3.45) y la vista principal (figura 3.46), así como para contener las funciones de agregar usuario y cambiar contraseña. Dichos controladores se presentan a continuación.

- **InicioController:** *Este controlador se encarga de realizar la autenticación del sistema. Lo más relevante de él es la encriptación de claves para la autenticación. Por medidas de seguridad, las contraseñas dentro de la base de datos se encuentran encriptados mediante una fórmula que no permite el desencriptamiento, por ende, la validación contra la base de datos debe aplicar dicha fórmula y el resultado será comparado contra la clave almacenada en la base de datos. El fragmento de código que realiza esto es el siguiente:*

```
clave = Digest::SHA1.hexdigest("1#{clave}0").strip
usuario = Usuario.autenticar(login, clave)
```

- **PrincipalController:** *Este controlador maneja la información del usuario. Provee funciones que soportan las funcionalidades de agregar nuevos usuarios y cambiar contraseña. En ambas funciones se realiza la encriptación de la contraseña por medidas de seguridad. La sentencia que permite la encriptación y creación de usuarios con la misma se encuentra a continuación:*

```
clave = Digest::SHA1.hexdigest("1#{form[:clave]}0").strip

u = Usuario.create({ :login=>form[:usuario],
        :clave=>clave, :rol_id=>rol})
```

Por último, para culminar la cuarta iteración del sistema DatDesigner, a continuación se presenta la sentencia necesaria para realizar el registro de un evento en la bitácora. Cabe destacar que dicha sentencia es colocada en el controlador de cada evento que se quiera registrar.

```
Bitacora.insertar("EVENTO",session[:login_usuario] ,
 Funcionalidad::NOMBRE_FUNCIONALIDAD,request.remote_ip,
tipo_importancia)
```

3.5. Iteración V

Durante esta quinta y última iteración se realizó el análisis, diseño y desarrollo de la funcionalidad de monitoreo de temperatura y recursos del sistema la cual permite el monitoreo del estado real de los componentes de hardware del centro de datos en cuanto a valores de temperatura, velocidad de ventiladores, memoria, etc. Además se realizó el análisis del modelo de datos del sistema, necesario para poder incorporar esta nueva funcionalidad.

3.5.1. Fase de Iniciación

En esta fase se realizaron las tareas de levantamiento de información y definición de requerimientos necesarios para el desarrollo de la funcionalidad de monitoreo.

La recopilación de requerimientos de software para esta funcionalidad fueron los siguientes:

- *Un servidor de bajo consumo de recursos que puede ejecutarse en los servidores de hardware y permita comunicación con el servidor*

de la aplicación para proveer información del estado de los recursos de dicho hardware.

- *Interfaz que permita elegir un hardware en particular que pueda ser monitoreado para consultar el estado de los recursos.*
- *Una opción que permita activar el monitoreo continuo de los recursos, donde se pueda indicar cada cuanto tiempo se debe realizar dicho monitoreo automáticamente.*
- *Mensajes de alerta cuando se produzca el fallo de un recurso, o los valores monitoreados no estén dentro del rango establecido como buen funcionamiento.*
- *Permitir el monitoreo continuo de varios recursos a la vez.*

3.5.2. Fase de Elaboración

En esta última iteración se realiza el análisis, diseño y desarrollo de la funcionalidad de monitoreo de la aplicación DatDesigner. En esta fase se realiza el diseño de la solución para dicha funcionalidad, así como el análisis del modelo de datos y la realización de los diagramas de casos de uso y diagramas de clase.

Para esta funcionalidad, por no requerir de un registro histórico de información sobre el estado de los recursos, sino que la información mostrada es obtenida en tiempo real, el modelo de datos no sufre muchas modificaciones. Realizando un análisis a las funcionalidades requeridas y al diseño de la solución se obtiene que para dar soporte al sistema monitor se debe modificar solo una estructura del modelo de datos, la cual es la estructura "Stage". Dicha modificación es para agregar dos nuevas características que son necesarias para guardar información de si un elemento del área de diseño puede ser monitoreado, es decir, que cuente con la configuración necesaria para conectarse con la aplicación de monitoreo y la segunda característica es la dirección donde se encuentra

este hardware dentro del centro de datos. La estructura "Stage", una vez modificada se encuentra en la figura 3.48.

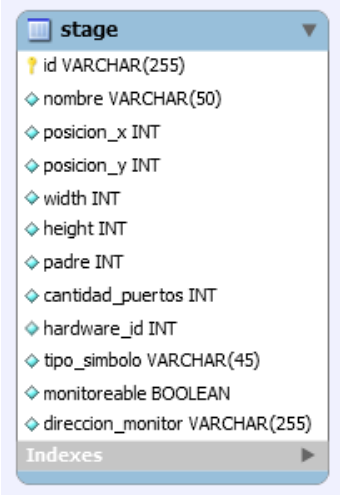

Figura 3.48 Modificaciones a la estructura Stage para el monitor

Seguidamente, para dar cabida dentro del sistema a la funcionalidad de monitoreo, y en base a los requerimientos obtenidos en la fase de iniciación de esta iteración, se definen los casos de uso. Para esta iteración solo se realizaran casos de uso de nivel uno. Es importante señalar que a pesar de ser una funcionalidad más del sistema, esta se divide en dos grandes componentes que trabajan dentro del mismo esquema lógico, pero físicamente separados. El primer componente es el monitor, que se encuentra en ejecutándose en el hardware monitoreado y se comunica con la aplicación DatDesigner mediante DRB (Distributed Ruby, Previamente definido en el marco teórico) y la otra parte de la funcionalidad es la que se encuentra dentro de la aplicación Rails como tal y se encarga de consultar remotamente mediante DRB los valores de temperatura, así como también recibir las solicitudes mediante REST en caso de estar activado el monitoreo asíncrono continuo.

Los casos de uso de nivel uno del monitor de recursos se muestran en la figura 3.49.

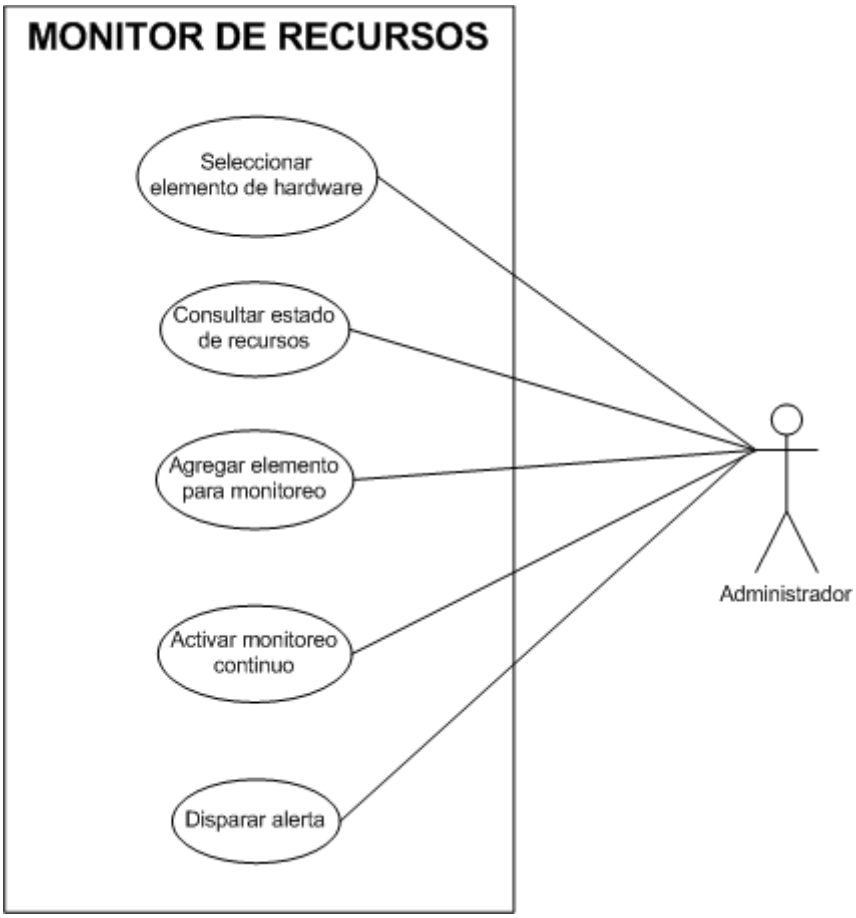

Figura 3.49 Casos de uso del monitor de recursos – Nivel 1

Asimismo, a continuación se encuentra las descripciones de los casos de uso de nivel uno.

- **Seleccionar elemento de hardware:** *Permite al usuario seleccionar un elemento disponible para el monitoreo.*
  Pre-condición: *El elemento debe poder ser monitoreado.*
  Post-condición: *Se debe mostrar la información de estado del recurso seleccionado en caso de estar el servidor accesible, y en caso contrario mostrar un mensaje de error.*

- **Consultar estado de recursos:** *Permite al usuario consultar el estado de un recurso del sistema.*

Pre-condición: *Debe haber seleccionado previamente un recurso disponible, y el servidor de monitoreo debe estar activo y accesible por la aplicación.*
Post-condición: *Se debe mostrar el estado del recurso seleccionado.*

- **Agregar elemento para monitoreo:** *Permite a un usuario agregar un nuevo recurso para ser monitoreado.*
  Pre-condición: *El recurso debe existir y debe poder ser monitoreado.*
  Post-condición: *Se agrega el recurso a la lista de recursos disponibles para monitoreo y se establece comunicación con el servidor de monitoreo del mismo.*

- **Activar monitoreo continuo:** *Permite a un usuario activar el monitoreo asíncrono y continuo de los recursos del sistema.*
  Pre-condición: *El servidor debe estar activo.*
  Post-condición: *Se debe monitorear el estado del recurso constantemente cada cierto tiempo. El intervalo de tiempo es indicado por el usuario.*

- **Disparar alerta:** *Avisa al usuario que un recurso está en riesgo, o funcionando fuera del rango de parámetros establecidos como normales.*
  Pre-condición: *El sistema de monitoreo continuo debe estar activo.*
  Post-condición: *Se debe notificar al usuario, mediante la aplicación principal, sobre un error en el recurso.*

Una vez culminado el análisis de requerimientos y los casos de uso del sistema, se puede plantear el diagrama de clases que representa la arquitectura de la solución. Físicamente, las clases están separadas en dos partes: a) "Monitor", la cual se encuentra ejecutándose en el hardware monitoreado y se comunica remotamente mediante REST con la clase "ServidorMonitor", y b) las otras cuatro clases que se encuentran dentro del sistema DatDesigner, donde la clase "ConexionRemota" se comunica remotamente con "Monitor" mediante DRB. Dicho diagrama de clases se muestra a continuación y es explicado en detalle posteriormente en la fase de elaboración de la presente iteración.

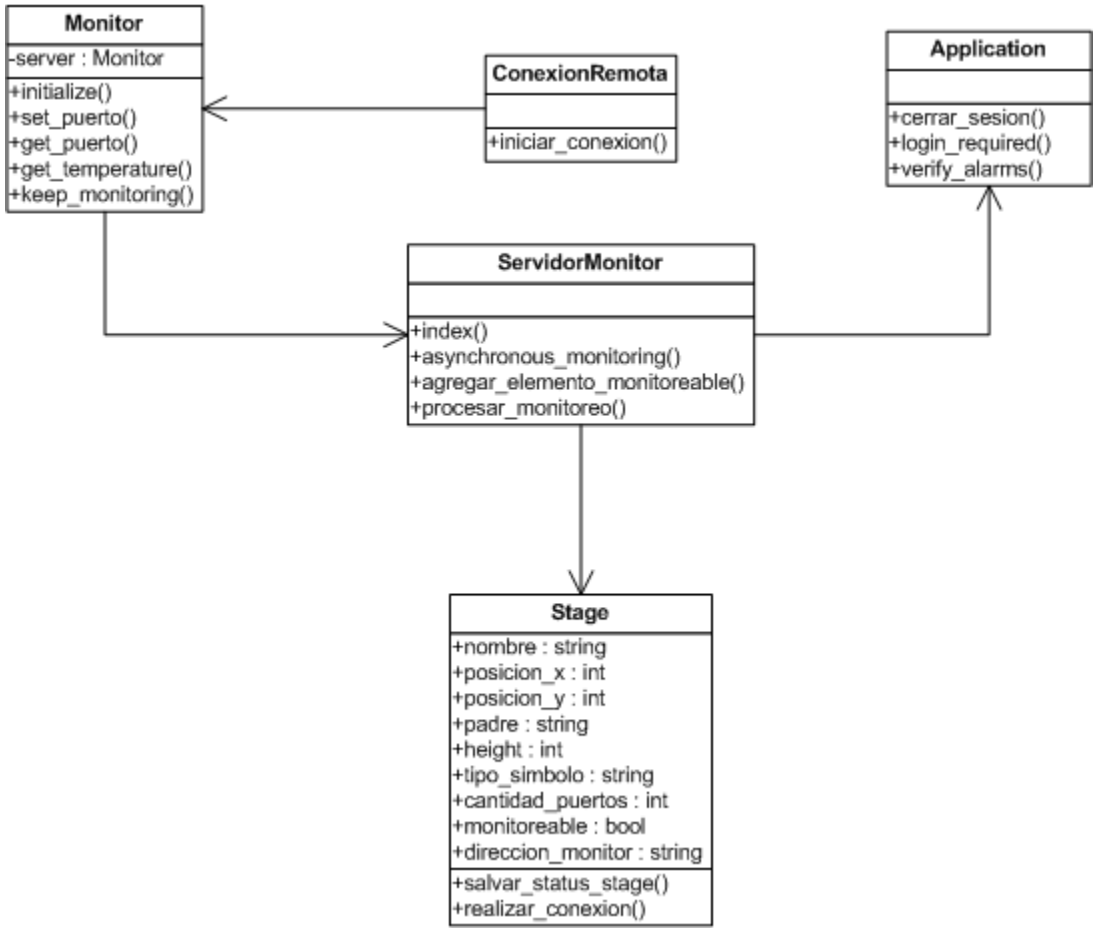

Figura 3.50 Diagrama de clases del monitor de recursos

Asimismo, para explicar en mayor detalle el funcionamiento de la funcionalidad de monitoreo de recursos, a continuación se presenta el diagrama de secuencia que muestra la interacción entre los diferentes componentes que forman el monitor de recursos y la secuencia en la cual realizan dicha interacción.

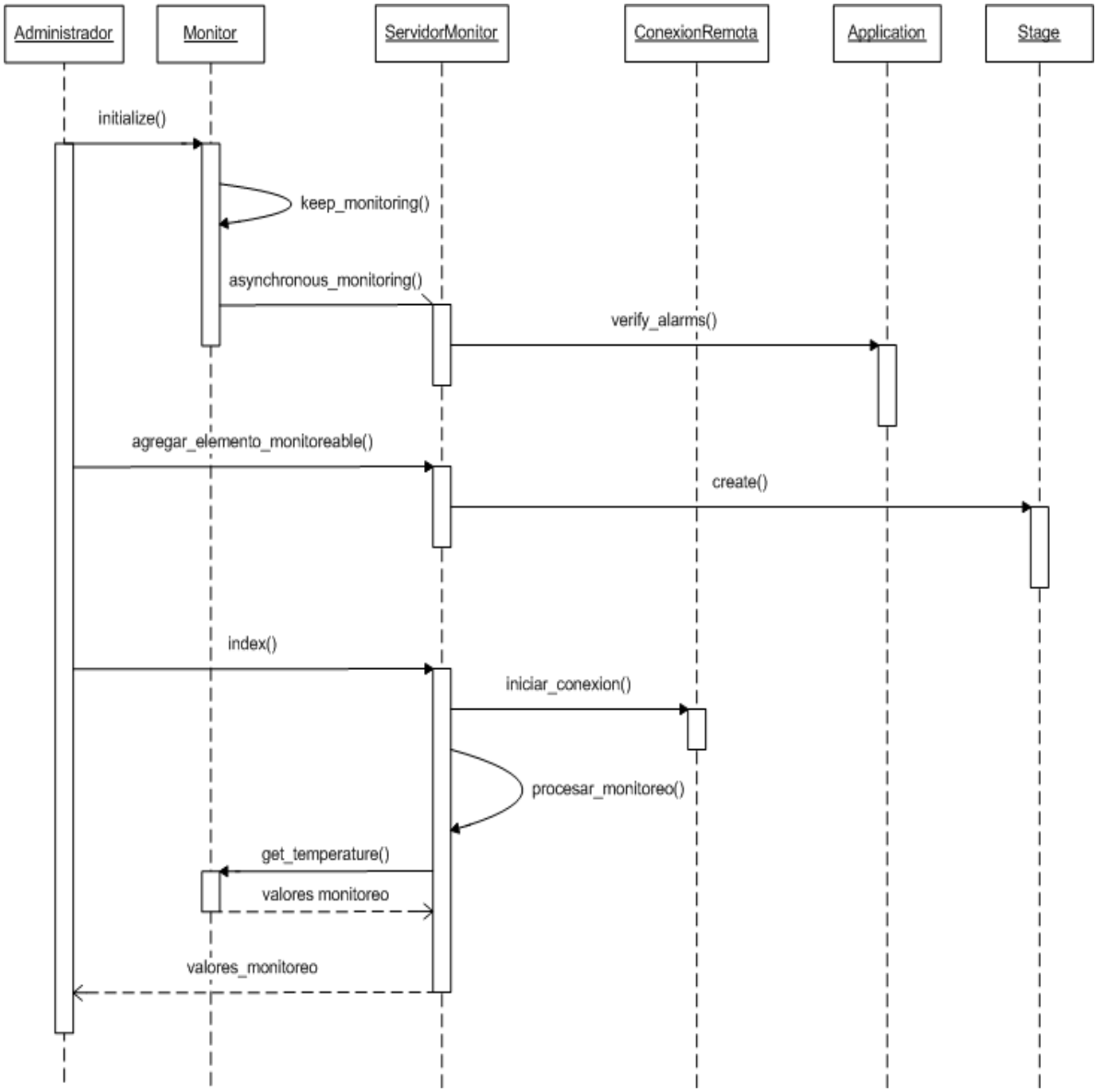

Figura 3.51 Diagrama de secuencia del monitor de recursos

3.5.3. Fase de Construcción

En la presente fase se procede al desarrollo del diseño propuesto en las fases de inicio y elaboración de la funcionalidad de monitoreo de recursos. En principio se realizaron las modificaciones necesarias a la base de datos para dar soporte a las nuevas características y se realizo el desarrollo del modelo "ConexionRemota", el cual será el encargado de establecer la comunicación mediante DRB para obtener los valores de temperatura. Es importante destacar, que mediante DRB se optimizan las comunicaciones y se provee una rápida respuesta al usuario, a pesar de que la información se encuentra remota. A continuación se presenta la implementación que hace posible establecer el punto inicial de dicha comunicación.

```
class ConexionRemota < ActiveRecord::Base
require 'drb'
      def self.iniciar_conexion (direccion, puerto)
            DRb.start_service
            return DRbObject.new(nil, "druby://#{direccion}:#{puerto}")
      end
end
```

Una vez creado el modelo, para invocar las funciones del servidor remoto se debe crear una instancia del modelo anterior mediante el método "new" y posterior a esto se pueden invocar todos los métodos públicos que en el servidor remoto se encuentren.

Luego de tener el modelo que realiza la conexión remota, ya se puede definir el servidor "Monitor" que se encargue de obtener los valores de temperatura. Para ellos se utilizó un controlador llamado sensors (Definido previamente en el marco teórico), el cual arroja la salida de monitoreo a una consola. Es importante destacar que esta función está disponible solo para equipos UNIX que soporten la ejecución de sensors. La figura 3.52 muestra una salida por consola de sensors, el cual tiene configurado dos controladores para monitoreo.

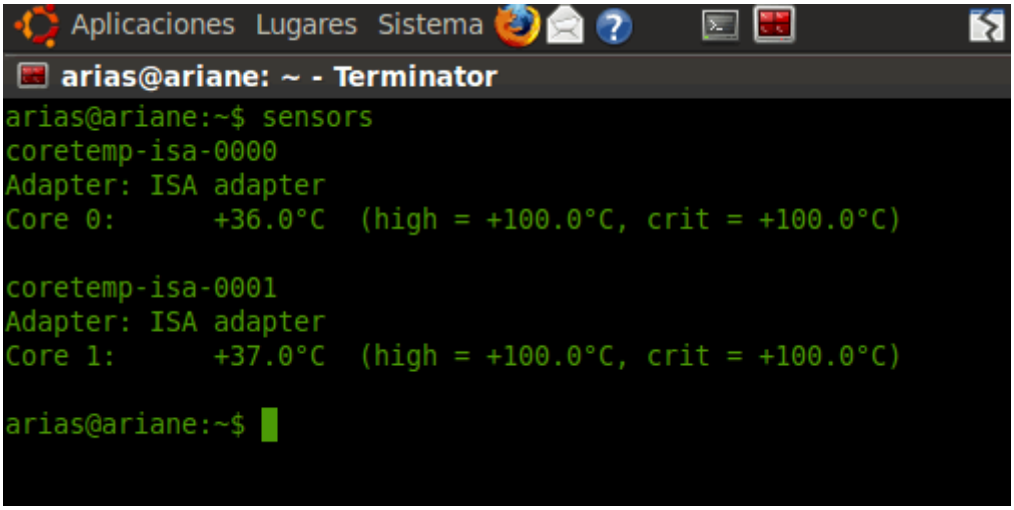

Figura 3.52 Salida por consola de sensors

El objetivo del monitor es capturar esta salida, procesarla y enviarla de regreso al servidor que realizó la invocación remota. Para ello, lo primero que se realizó fue redirigir esta salida a un archivo de texto con la finalidad de utilizar las funcionalidades de manejo de archivos que provee el lenguaje ruby. De acuerdo con esto, la salida producida en el archivo txt se muestra en la figura 3.53. Luego de esta redirección se realiza el procesamiento siguiendo el formato establecido por la salida de sensors, donde se pueden obtener los valores del controlador y adaptador utilizados, así como los valores que provee cada uno.

```
1  coretemp-isa-0000
2  Adapter: ISA adapter
3  Core 0:      +37.0°C  (high = +100.0°C, crit = +100.0°C)
4
5  coretemp-isa-0001
6  Adapter: ISA adapter
7  Core 1:      +39.0°C  (high = +100.0°C, crit = +100.0°C)
8
```

Figura 3.53 Redirección de la salida de sensors a formato txt

La primera sección a describir dentro de la clase Monitor es su método "initialize", el cual captura los parámetros que vienen en la línea de comandos y los procesa como atributos de la clase. Los parámetros que recibe el servidor son los siguientes: a) Identificador del hardware a ser monitoreado, b) Puerto por el cual se produce la salida y levanta el servidor, c) Dirección en la cual consume el servicio de la aplicación DatDesigner, d) Opción que indica si se activa el monitoreo continuo y e) Intervalo de tiempo, en segundos, entre el cual se monitorea continuamente.

La segunda sección importante es el método que realiza el monitoreo. Este es el que redirige la salida de sensors a un documento txt y la procesa. A continuación se muestra la implementación de dicha función.

```
def get_temperature
      #Se realiza la llamada a sensors en el sistema operativo
      system("sensors > salida.txt")
      f = File.new("salida.txt", "r")

      record, records, read_first = {}, [], true

      #para cada entrada en el archivo se lee según el formato y se
      #procesa
      f.each{|line|
            if read_first
                  record[:driver] = line
                  read_first = false
            else
                  if line == "\n"
                        records << record #salvar_buffer
                        read_first = true
                        record = {}
                  else
                        key_value = line.split(":")
                        record["#{key_value[0]}"] = key_value[1]
                  end
            end
      }
      #records tendrá todos los pares nombre valor resultado
      #del monitoreo
      return records
end
```

Asimismo, el monitoreo continuo es la parte más compleja de esta clase. Para ello, el método "keep_monitoring" crea un nuevo hilo de ejecución que se encarga de ejecutarse cada cierto intervalo de tiempo y realizar la llamada a la función de obtener temperaturas para verificar que no existan alertas de seguridad en dicha salida. Sensors provee información sobre los parámetros de funcionamiento normales para cada recurso que monitorea, y cuando algún valor se sale de dicho rango coloca la palabra "ALARM" al final de la línea que contiene el valor del recurso, para indicar que está fuera de sus parámetros normales. Asimismo, si se produce una alerta, contacta vía REST a la aplicación para informar al respecto. A continuación se muestra la implementación de dicha función.

```
def keep_monitoring
      #Se crea el nuevo hilo o Thread
      t1 = Thread.new{
            while true
                  records = get_temperature
                  buffer_error = {}
                  records.each do |r|
                        r.each do |key, val|
                              buffer_error[key] =
                              val if val.include? "ALARM"
                        end
                  end
                  #Se realiza el POST via REST de buffer_error
                  response =
                        Net::HTTP.post_form(URI.parse('http://#{server.
                  dominio}:#{server.puerto_app}/servidor_monitor/asynch
                  ronous_monitoring'),buffer_error)
                  puts "Sleeping #{@periodicidad} secs =>  #{Time.now}"

                  #Se duerme el hilo para no consumir recursos mientras
                  #espera el próximo intervalo de monitoreo
                  sleep(@periodicidad)
            end
      }
end
```

Por último en la clase "Monitor", se debe realizar la instanciación del Monitor y la ejecución del servidor DRB como se muestra a continuación.

```
#Se crea la instancia de la clase monitor
server = Monitor.new
#Se iniciar el servidor DRB
DRb.start_service("druby://#{server.dominio}:#{server.puerto}", server)
#Se realiza el join del hilo de ejecución actual para mantenerse activo
DRb.thread.join
```

Una vez esté activo el servidor, se comunicará vía REST con la aplicación DatDesigner, la cual en el caso de el monitoreo continuo, capturará los parámetros de los recursos en alerta y los mostrará al usuario, esto en caso de haber algún recurso es estado de alerta.

Conclusiones

Concluido el presente Trabajo Especial de Grado, se cumple con el objetivo general de la investigación, el cual permitió diseñar *y desarrollar una herramienta de software libre que permita el diseño de centros de datos, para así poder llevar un inventario de los equipos, reflejar las interconexiones de los mismos y visualizar las aplicaciones que en estos se ejecutan. Dando solución al problema planteado inicialmente, que consistía en llevar un control específico de los componentes de hardware y software que se encuentran en las instalaciones, así como tener un perfil de uso de los equipos y saber el estado de uso y obsolencia en que se encuentran.*

La metodología de desarrollo empleada fue una adaptación de UP ágil, debido a que constituye un método estándar para el análisis, diseño, desarrollo y documentación de sistemas computacionales y puede ser adaptado fácilmente a sistemas Web, teniendo como primordial característica la adaptabilidad para dar respuesta eficaz en cada una de las fases de desarrollo, así como la rapidez y eficiencia a la hora de implementar un sistema, gracias a que está orientado a utilizar solo la documentación necesaria para describir el sistema y hace énfasis en culminarlo rápidamente.

El proceso contó con cinco grandes iteraciones, las cuales a su vez contaron con tres fases: fase de iniciación, fase de elaboración y fase de construcción. De esta manera, la primera iteración se utilizó para el desarrollo del sistema cargador de datos, la segunda iteración cubrió el sistema diseñador y la resolución de problemas, la tercera se utilizó para el refinamiento y corrección de problemas del sistema cargador y el sistema diseñador, la cuarta iteración abarcó el sistema visualizador y refinamiento de las 2 primeras iteraciones, además de las pruebas necesarias para

garantizar su correcto funcionamiento, y la quita y última abarco el sistema de monitoreo de recursos para los elementos de hardware del centro de datos. Asimismo, dentro de cada iteración, las fases se definieron de la siguiente manera: Fase de iniciación, para realizar el levantamiento de información de la iteración; Fase de elaboración para analizar el dominio del problema y realizar el diseño de la solución; Fase de construcción, para realizar la implementación de los componentes de la aplicación en la iteración actual.

Para el desarrollo exitoso del presente Trabajo Especial de Grado, fue necesario realizar un estudio minucioso que permitió puntualizar tópicos relacionados con los centros de datos, abordando los aspectos más importantes de: Componentes de centros de datos, comportamiento de dichos componentes, estructuras, conexiones, capacidades, características y arquitecturas, ya que el propósito de la aplicación es disponer de estos elementos de tal manera que se pueda realizar una representación lógica, lo más cerca posible a la realidad, del centro de datos a diseñar. Todo lo antes mencionado con la finalidad de facilitar el desarrollo y suministrar una solución eficaz al problema planteado.

Ahora bien, la fase de desarrollo fue posible gracias a la selección adecuada de tecnologías, que permitió codificar y optimizar las funcionalidades y requerimientos del sistema, para así cumplir con los objetivos específicos planteados en la propuesta del problema de investigación a resolver, y dentro de los parámetros de tiempo estipulados.

Con el desarrollo del sistema diseñador de centros de datos DatDesigner, se cuenta con una herramienta de software libre que proporciona las capacidades de llevar inventario actualizado de los recursos en un centro de datos, así como el monitoreo y optimización de las actividades cotidianas de administración que en este se llevan a cabo.

Consideraciones y Recomendaciones

Una vez culminado el Trabajo Especial de Grado, y establecidas las conclusiones se hace necesario sugerir las siguientes consideraciones y recomendaciones:

- *Para la utilización de la aplicación es recomendable utilizar monitores con resolución igual o mayor a 1280x1024 ya que permite un mejor desempeño a la hora de utilizar el espacio del área de diseño.*
- *Al momento de diseñar sería recomendable, para trabajos futuros en esta aplicación, considerar los elementos de contexto de los centros de datos, como por ejemplo mesas, sillas, paredes, etc. los cuales no se incluyeron por motivos de practicidad.*
- *Al momento de almacenar información de centros de datos de gran tamaño sería recomendable utilizar soluciones NO SQL para el almacenamiento de los elementos en el área de diseño y sus interconexiones, ya que se obtendría mejor rendimiento en la aplicación.*
- *Las capacidades de edición de características de los elementos en el área de diseño debería ser mayor.*
- *Para mejorar el rendimiento de la aplicación se recomienda migrar a las nuevas versiones del Framework Ruby on Rails.*

Referencias Bibliográficas y Digitales

[U-DC-EF] *Experience festival. Data Center Components. Extraído en 19 Oct 2009 desde http://www.experiencefestival.com/data_center_-_components.*

[U-DC-WK] *Wikipedia. Data Center. Extraído en 19 Oct 2009 desde http://en.wikipedia.org/wiki/Data_center.*

[U-CPD-WK] *Wikipedia. Centro de procesamiento de datos. Extraído en 19 Oct 2009 desde http://en.wikipedia.org/wiki/Centros_de_procesamiento_de_datos*

[U-RED-ML] *MercadoLibre. Componentes de una red. Extraído en 24 Oct 2009 desde http://guia.mercadolibre.com.ve/tutorial-redes-tipos-y-componentes-que-forman-5986-VGP*

[U-RIA-SID] *SIDAR. Interfaces enriquecidas de internet. Extraído en 06 Nov 2009 desde http://www.sidar.org/ponencias/2008/egyrs/rioja/#(1)*

[Duhl, 2003] *DUHL, Joshua. 2003. Rich Internet Applications. Extraído en 10 Nov 2009 desde http://www.adobe.com/platform/whitepapers/idc_impact_of_rias.pdf*

[P-RIA-DB] *BRADBURY, Danny. 2006. Rich prospects as HTML comes of age. Extraído 11 Nov 2009. Computer Weekly: pp 50-52, 18 abril 2006.*

[Google, 2008a] *Google, Inc. Google Maps. Citado 15 agosto, 2008. Disponible en: http://maps.google.com/*

[Google, 2008a] *Google, Inc. Google Maps. Citado 15 agosto, 2008. Disponible en: http://maps.google.com/*

[Google, 2008b] *Google, Inc. Google Suggest. Citado 15 agosto, 2008. Disponible en: www.google.com/webhp?complete=1&hl=en*

[Yahoo, 2008] *Yahoo, Inc. Flickr. Citado 16 agosto, 2008. Disponible en: http://www.flickr.com/*

[González-Moreno; 2006] *Estudio de la API para el manejo de persistencia de EJB 3.0 y del enfoque de desarrollo AJAX aplicados en la construcción de aplicaciones Web enriquecidas. Seminario Facultad de Ciencias, UCV 2006. Extraído en 10 Nov 2009.*

[U-INFIT] *Informit. Ruby Language. Extraído en 18 Nov 2009 desde http://www.informit.com/articles/article.asp?p=18225*

[Steve Burbeck; 1992] *Steve Burbeck, 1992. Model-View-Controller Architecture. Extraído en 18 Nov 2009 desde http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html*

[U-ror] *Ruby on Rails org. Web development that doesn't hurt. Extraído en 18 Nov 2009 desde http://rubyonrails.org/*

[A-Adobe-as3] *Adobe. FLASH AS3 PROGRAMMING. Copyright © 2008 Adobe Systems Incorporated. Todos los derechos reservados.*

[U-AS3-TC] *Adobe. Action Script. Extraído en 18 Nov 2009 desde http://www.adobe.com/devnet/actionscript/*

[P-AMF-AD] *Adobe. Action Message Format 3 Specification. Copyright © 2008 Adobe Systems Incorporated. Todos los derechos reservados.*

[U-PT-ORG] *Prototype. Prototype Framework. Extraído en 18 May 2010 desde http://www.prototypejs.org/*

[U-SW-ER] *Electric Rain. Swift3D Software. Extraído en 18 May 2010 desde http://www.erain.com/Products/Swift3*