**Universidad Oscar Lucero Moya de Holguín**

**Facultad de Informática y Matemática**

# *Sistema informático para la gestión de información de la valoración económica de proyectos*

**Trabajo de diploma para optar por el título de Ingeniero Informático**

Autor: Edder Yoel Peña Rodríguez

Tutores: Ing. Yisel Clavel Quintero

Dr.C. Rodolfo García Bermúdez

**Holguín, 2014**

*"Estoy convencido de que la mitad de lo que separa a los empresarios exitosos de los que no triunfan es la perseverancia."*

*Steve Jobs*

# RESUMEN

La Empresa de Proyectos e Ingeniería (ENPA), perteneciente al Ministerio de la Agricultura, es una entidad de consultoría, ingeniería y diseño, que tiene el objetivo de asumir la amplia variedad de proyectos que demanda el sector agropecuario del país. Los proyectos desarrollados en esta entidad llevan asociados consigo un costo de elaboración, el cual es calculado y procesado por el jefe del departamento en el cual se ejecute el proyecto.

El Sistema para la Gestión de Información de la Valoración Económica de Proyectos desarrollados en los departamentos productivos de la Empresa Nacional de Proyectos e Ingeniería surge por la necesidad de contar con una herramienta que apoye el desempeño y calidad de trabajo de los jefes de departamento, pues hasta el momento realizan el trabajo de forma manual, esto dificulta el registro, actualización y recuperación de la información, lo cual se traduce en ineficiencias.

En este documento se expone la fundamentación del tema, la propuesta de herramientas para el desarrollo del producto, así como su descripción y construcción, con el propósito fundamental de gestionar la información de la valoración económica de proyectos de una manera eficiente.

Para el desarrollo de esta aplicación se empleó la metodología de desarrollo de *software* Iconix, como gestor de bases de datos PostgreSQL, y el lenguaje de programación Java. Con el objetivo de lograr que las interfaces de la aplicación *Web* estén enriquecidas en componentes visuales se utilizó el *Framework* ZK y como Servidor *Web* para el soporte de esta aplicación el Apache Tomcat.

# ABSTRACT

The Information Management System of the Economic Evaluation of Projects developed in the production department of the “Empresa Nacional de Proyectos de Ingeniería”, arises from the need of a tool that supports the performance and quality of work of the heads of department. So far they have been doing the work manually and this makes the registration, updating and retrieval of information a very inefficient process.

Given the need to expedite this process, and its characteristics, a Web Application was developed to facilitate information management and data processing with new potentialities. In this document is shown the theme fundamentation, the proposed tools for the product development, and the description and construction of it, with the primary purpose of managing information of economic valuation of projects. It was used as development methodology Iconix, PostgreSQL as database manager and Java as programming language. The Web Application interfaces and visual components where enriched in the ZK Framework and was used as Web Server to support this application Apache Tomcat.

# ÍNDICE

# ÍNDICE DE FIGURAS



# ÍNDICE DE TABLAS

# Introducción

La informática se ha convertido en la actualidad en una de las ciencias que ha proporcionado una gran cantidad de avances en todos los procesos de cambio tecnológico. En el mundo, prácticamente nada es concebido sin la utilización de herramientas informáticas, por lo que cada vez más existen sitios, aplicaciones de escritorio y aplicaciones *Web* para facilitar la organización, el manejo y el acceso a los contenidos y datos, disponibles en las diferentes entidades y empresas de la sociedad.

En este sentido, las empresas de proyecto de nuestro país no se han quedado atrás, cada vez son más las empresas e instituciones de este tipo, que utilizan las tecnologías digitales para mejorar los procesos que ayudan de forma general a la eficiencia de su funcionamiento.

En Cuba, con la implantación y desarrollo de nuevas tecnologías, son varias las instituciones que aprovechan estas posibilidades, para sumarse al grupo de beneficiados por la informática y la digitalización de los procesos. La Empresa Nacional de Proyectos e Ingeniería (ENPA) también tiende a la informatización de sus principales procesos y actividades. Esta investigación se realiza en la ENPA, la cual está compuesta por áreas productivas y departamentos de apoyo.

El departamento de diseño es la célula fundamental que agrupa procesos de producción vinculados a las actividades que se desarrollan en esta empresa. Además generan una serie de tareas cuyo responsable administrativo es el Jefe de Departamento. Una de las tareas administrativas esenciales y por la que un Jefe de Departamento debe velar que se cumpla es la relacionada el cumplimiento del plan de producción.

El Jefe de Departamento ejecuta y evalúa el proceso de gestión del cierre y es el encargado de conformar controlar la estabilidad y cumplimiento de los planes de la empresa en conjunto con la dirección de la empresa.

Después de que los jefes de departamento generan sus cierres individuales, entonces el Jefe de Producción obtiene esos planes y realiza la conformación

del cierre general de la empresa. Actualmente el Jefe de Producción no conoce la situación real del estado de la producción que se está desarrollando en cada departamento, en los diferentes lugares tales como: el Departamento de Desarrollo, Departamento de Arquitectura, entre otros. Dado el análisis realizado al proceso gestión de información de la valoración económica de proyectos se encontraron las siguientes deficiencias:

- Se necesita tener el control de las valoraciones económicas de los proyectos en red, para así poder obtener reportes sobre el estado de la producción, cumplimiento de la misma, tendencias y necesidades.
- El método de guardar la documentación utilizado hasta la fecha dificulta la recuperación y actualización de la información y no se garantiza la perdurabilidad de la misma en el tiempo.
- Es importante destacar que el Jefe de Producción, recibe orientaciones y solicitudes de informes por parte de la dirección de la entidad, lo que lo obliga a procesar un gran volumen de datos que realiza de forma manual y necesita consultar varios documentos con el riesgo de que no estén actualizados, lo que provoca gran consumo de tiempo y posibilidad de cometer errores.
- No se cuenta con una forma eficiente de obtención de informes, que muestren los datos actuales sobre la existencia y estado de la producción de la empresa, para su correcto control y seguimiento del cumplimiento de la misma.

Las consideraciones anteriores revelan el siguiente **problema científico**: ¿Cómo favorecer la gestión de información de la valoración económica de proyectos en la ENPA?

El **objeto de estudio** en el cual se enmarca el problema planteado lo constituye el proceso de gestión de información de la valoración económica de proyectos.

Se propone como **objetivo** para dar solución al problema científico: desarrollar un sistema informático para favorecer la gestión de información de la valoración económica de los proyectos en la ENPA.

Precisándose como **campo de acción**: la informatización del proceso de gestión de información de la valoración económica de proyectos en la ENPA.

Para guiar la investigación se plantean las siguientes **preguntas científicas**:

1. ¿Cuáles son los fundamentos teóricos que sustentan la gestión de información de la valoración económica de proyectos?

2. ¿Cómo desarrollar un sistema informático que favorezca la gestión de información de la valoración económica de proyectos en la ENPA?

3. ¿Será sostenible el sistema informático propuesto?

4. ¿Cómo valorar el grado de aceptación del sistema informático propuesto?

Para dar respuesta a las preguntas científicas y cumplir el objetivo trazado, se realizaron las siguientes **tareas**:

1. Determinar los fundamentos teóricos que sustentan la gestión de información de la valoración económica de proyectos.

2. Analizar el sistema informático que favorezca la gestión de información de la valoración económica de proyectos en la ENPA.

3. Realizar la valoración de sostenibilidad del sistema informático.

4. Diseñar el sistema informático que favorezca la gestión de información de la valoración económica de proyectos en la ENPA.

5. Implementar el sistema informático que favorezca el proceso de gestión de información de la valoración económica de proyectos en la ENPA.

6. Valorar el grado de aceptación del sistema informático a través de Criterio de Expertos.

Para darle solución a las tareas planteadas se usaron una combinación de métodos teóricos y empíricos.

**Métodos teóricos**

**Análisis y síntesis**: se utilizó en el estudio de la información referente a los proyectos, permitió descomponer el sistema informático en sus diferentes partes para lograr un mejor entendimiento.

**Histórico lógico**: se empleó en el análisis del proceso de gestión de información de la valoración económica de proyectos en la ENPA, para conocer con mayor profundidad los antecedentes y las tendencias actuales.

**Enfoque sistémico**: se utilizó para el análisis y determinación de las relaciones funcionales, dependencias y la modelación del sistema informático.

**Modelación**: permitió representar de manera simplificada el flujo de trabajo en la ENPA y una mejor comprensión del proceso de gestión de información de la valoración económica de proyectos en la ENPA, así como obtener un producto de mayor calidad.

**Métodos empíricos**

**Entrevista**: permitió la interacción directa con el personal implicado en el sistema informático, recoger las informaciones importantes, las necesidades de los Jefes de Departamento para ayudar en su labor y determinar los principales requerimientos del sistema informático.

**Encuesta**: permitió elegir los expertos en materia de informática para la validación de la solución propuesta y obtener valoraciones conclusivas de estos sobre el sistema informático.

La presente investigación consta de introducción, dos capítulos, conclusiones, recomendaciones, glosario de términos, bibliografía y anexos.

**El Capítulo 1 se titula "Fundamentación Teórica"** y contiene los fundamentos de los aspectos relacionados con el objeto de estudio. Además, en este capítulo se hace una descripción de las principales tendencias y tecnologías para la construcción de la solución propuesta y de la metodología de desarrollo de *software* empleada.

**El Capítulo 2, "Descripción y Elaboración del Sistema"**, contiene los aspectos de mayor peso en el desarrollo del sistema informático propuesto. También explica el uso de la metodología de desarrollo de software expuesta en el capítulo inicial para el diseño de la aplicación. Además, se hace un estudio de sostenibilidad según su impacto socio-humanista, administrativo, tecnológico y ambiental y se hace la validación del mismo a través del criterio de usuarios del sistema.

# Capítulo 1. Fundamentación teórica

En este capítulo se presenta un análisis general de los fundamentos teóricos, con el propósito de entender con más claridad los procesos de negocios para obtener una solución eficiente al problema de este trabajo. Además, se hace referencia a las definiciones básicas relacionadas con la solución propuesta, así como las tecnologías para su desarrollo. Por último, se refleja la metodología de desarrollo de *software* que se determinó a utilizar para dar solución a la propuesta, así como las tecnologías y herramientas escogidas para el desarrollo de la aplicación por su vigencia en la actualidad.

## 1.1 Empresa de Proyectos e Ingeniería

La Empresa de Proyectos e Ingeniería (ENPA), perteneciente al Ministerio de la Agricultura, es una entidad de consultoría, ingeniería y diseño, que tiene el objetivo de asumir la amplia variedad de proyectos que demanda el sector agropecuario del país. Entre su objeto social se encuentra:

- Brindar servicios técnicos, proyección e ingeniería de proyectos de inversión y de la construcción, así como en topografía e investigaciones ingenieras aplicadas a la construcción, hidrológicas y otras.
- Ofrecer servicios técnicos en obras de arquitectura e ingenierías, incluyendo las obras industriales del sector agropecuario y otros servicios técnicos de defectación.
- Realizar la administración y dirección integrada de proyectos.

Para cumplir con su objeto social, la entidad cuenta con especialistas altamente calificados en las áreas de Arquitectura, Investigaciones aplicadas, Desarrollo agropecuario y Servicios ingenieros (Calzadilla 2012).

### 1.1.1 Diagnóstico de la situación existente en la empresa

En el objeto social de la Empresa de proyectos e ingeniería ENPA se encuentra el brindar servicios técnicos, proyección e ingeniería de proyectos de inversión y

de la construcción, así como en topografía e investigaciones ingenieras aplicadas a la construcción, hidrológicas y otras, servicios que se convierten en proyectos los cuales conllevan consigo un costo asociado.

Los proyectos son realizados por un proyectista o equipo de estos, los cuales después de un amplio estudio y proceso de desarrollo son colocados en una carpeta compartida en un servidor para ser revisados por los departamentos de Calidad y el departamento de diseño, donde radica el jefe de producción que es el máximo responsable encargado de revisar que los proyectos cumplan con todos los requerimientos establecidos.

Cada proyecto tiene un costo asociado en el cual se reflejan los gastos en los cuales ha incurrido la entidad para su desarrollo, como son por ejemplo: los proyectistas que participan en él con el salario de cada uno, gastos en materias primas, insumos consumidos entre otros gastos. Al terminar un proyecto el especialista principal del departamento en el cual se desarrolle dicho proyecto debe realizar la valoración económica. Para esto emplea una ficha de costo la cual se colocará en el servidor para su posterior revisión por el jefe de producción. Una vez revisadas todas esas valoraciones el jefe de producción procede a crear el cierre de producción de la entidad ver (**Figura 1.1**).

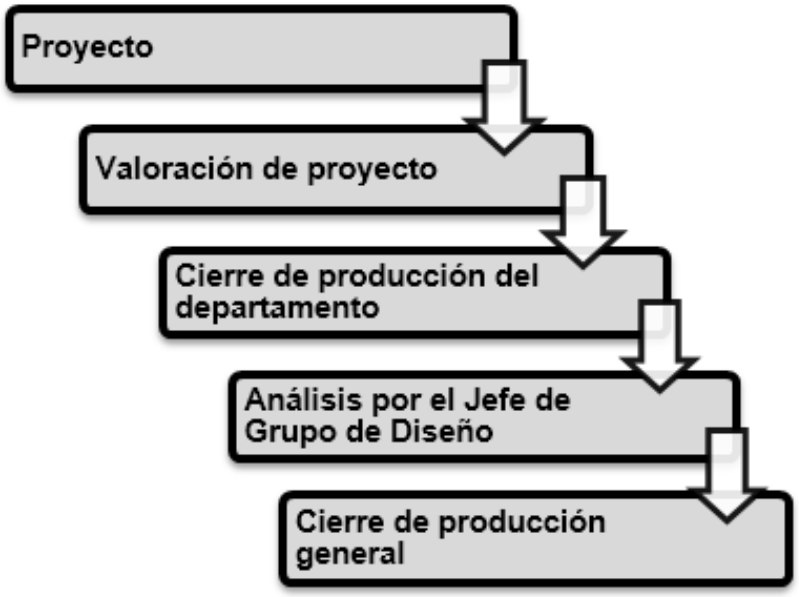



**Figura 1.1: Descripción del proceso**

## 1.2 Sistemas informáticos vinculados al campo de acción

Se realizó un estudio de *benchmarking* y se encontraron diferentes sistemas informáticos existentes vinculados al campo de acción, los cuales fueron valorados y ayudaron en gran medida a proponer ideas al diseño del sistema informático requerido, a analizar y plantear una serie de necesidades existentes, a señalar algunas perspectivas de desarrollo posibles, a proponer soluciones, herramientas, ventajas y concretar lo que se desea como producto final. Estos se describen a continuación.

### SISTEMA DE CONTROL DE COSTOS VISUAL

El sistema de control de costos cuenta con una serie de submódulos integrados (órdenes de compra, inventario de materiales, contratación, registro de facturas) que permiten controlar los costos directos de un proyecto a través de un proceso de integración. El sistema de costos directos tiene implementado los modelos de costos por salidas de almacén y facturación. Este sistema de costos está diseñado bajo la modalidad de multiusuario o monousuario (Sistemas 2012).

### Sistema de evaluación económica

El sistema permite evaluar la viabilidad de un proyecto constructivo, evaluando la rentabilidad a través del TIR y el VPN. El sistema parte de la definición de los costos del terreno, los costos de construcción, los honorarios, los impuestos, el plan de ventas, aportes de socios y préstamos a terceros (Sistemas1 2012).

### EvalAs: *Software* para Evaluación de Proyectos de Inversión Productivos

Permite el ingreso de información de Proyectos de Producción. Datos de: costos fijos, impuestos, costos variables, etc. Los períodos pueden indicarse en años en meses. A partir de los datos ingresados, se calcula la Matriz de Flujos de Caja y luego los principales indicadores financieros.

Permite detallar la información de los créditos solicitados para financiar el Proyecto. El software calcula automáticamente las cuotas para los préstamos con sistemas de amortización francés y alemán, permitiendo adicionalmente el

ingreso manual de los montos de las cuotas, si se utiliza otro sistema. A partir de estos datos, se calculan nuevamente los principales indicadores del Proyecto (EvalAs 2012).

Luego de un estudio acerca del estado del arte sobre sistemas informáticos existentes relacionados con la gestión de información para la valoración económica de proyectos, se concluye que ningún sistema encontrado cumple con los requisitos previstos y funcionalidades del sistema diseñado. En atención a lo anterior se puede apreciar los beneficios de desarrollar herramientas que brinden en mayor medida una adecuada respuesta a los requerimientos y necesidades de los departamentos productivos de la ENPA, y se brindan mayores posibilidades de adaptaciones futuras y acceso a las informaciones, todo sobre bases del uso de un lenguaje de programación sobre plataforma Web, obteniéndose de esta forma la propuesta final de desarrollo del presente trabajo.

## 1.3 Aplicaciones *Web*

En la ingeniería de *software* se denomina aplicación *Web* a aquellas aplicaciones que los usuarios pueden utilizar mediante el acceso a un servidor *Web* a través de *Internet* o de *Intranet* por medio de un navegador. En otras palabras, es una aplicación *software* que se codifica en un lenguaje soportado por los navegadores *Web* (HTML, JavaScript, Java, asp.net, php, etc.) en la que se confía la ejecución al navegador.

También una aplicación *Web* es un conjunto de páginas *Web* enlazadas que visualizan la información que se quiere mostrar a través de ella. Constituye una de las mejores herramientas para divulgar, gestionar y compartir la información, por lo que trae consigo un aumento de la eficiencia en cuanto a la manipulación de gran cantidad de elementos (LOPEZ 2010).

Ventajas de las aplicaciones *Web*:

- Ahorra tiempo: se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa.

- No hay problemas de compatibilidad: Basta tener un navegador mínimamente actualizado para poder utilizarlas.
- No ocupan espacio en el disco duro.
- Actualizaciones inmediatas: Como el *software* lo gestiona el propio desarrollador, cuando se conecta se usa siempre la última versión que haya lanzado.
- Consumo de recursos bajo: Dado que toda (o gran parte) de la aplicación no se encuentra en el ordenador local, muchas de las tareas que realiza el *software* no consumen recursos porque se realizan desde otro ordenador.
- Multiplataforma: Se pueden usar desde cualquier sistema operativo porque sólo es necesario tener un navegador.
- Portables: Es independiente del ordenador donde se utilice (un PC de sobremesa, un portátil, un móvil...), porque se accede a través de una página *Web* (sólo es necesario disponer de acceso a Internet o a la intranet donde se aloje la aplicación).
- La disponibilidad suele ser alta, porque el servicio se ofrece desde múltiples localizaciones para asegurar la continuidad del mismo.
- Los virus no dañan los datos porque estos están guardados en el servidor de la aplicación.
- Colaboración: Gracias a que el acceso al servicio se realiza desde una única ubicación es sencillo el acceso y compartición de datos por parte de varios usuarios.
- Los navegadores ofrecen cada vez más y mejores funcionalidades para crear aplicaciones enriquecidas de Internet (García 2013).

En general, la gran ventaja de utilizar estas tecnologías es precisamente que las aplicaciones *Web* se desarrollan en un ambiente Cliente/Servidor. Esto trae como beneficio que la mayor parte del procesamiento se realice en el servidor, se dispone de una buena interfaz gráfica y una mejor interacción con el usuario, que garantiza la integridad de la información (GuíaProgramador 2012).

Para el desarrollo del sistema informático para la gestión de información de la valoración económica de proyectos en la ENPA, se hará un análisis de las tendencias y tecnologías actuales que se utilizan en el mundo para desarrollar aplicaciones *Web*.

## 1.4 Tendencias y Tecnologías actuales para el desarrollo del sistema informático propuesto

Para llevar a cabo la realización de una aplicación es primordial realizar un análisis de las tendencias y tecnologías actuales, a continuación se detallan aquellas que fueron estudiadas y se especifican las escogidas para el desarrollo de la presente investigación.

### 1.4.1 Arquitectura cliente – servidor en la *Web*

La arquitectura cliente - servidor consiste básicamente en que un programa, el Cliente informático, realiza peticiones a otro programa, el servidor, que les da respuesta (**Figura 1.2**). Según *International Business Machines* (IBM) la arquitectura Cliente-Servidor: "Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas" (Tareas 2013).

En esta arquitectura se pueden distinguir 3 capas o niveles: **Figura 1.2: Arquitectura Cliente-Servidor**

- Manejador de Bases de Datos (Nivel de almacenamiento)
- Procesador de aplicaciones o reglas del negocio (Nivel lógico)
- Interfaces del usuario (Nivel de presentación)

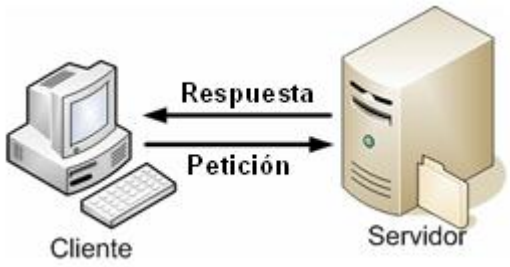



**Figura 1.2: Arquitectura Cliente-Servidor**

Esta arquitectura facilita que los recursos sean centralizados, pues el servidor puede administrar los recursos que son comunes a todos los usuarios. Posee seguridad mejorada gracias a que la cantidad de puntos de entrada que permite el acceso a los datos no es muy alta (Tareas 2013).

Además es escalable, fácil de mantener, presenta centralización del control pues los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado, no pueda dañar el sistema. La existencia de plataformas de *hardware* cada vez más baratas, permite que los costes se hayan reducido considerablemente. Además, se pueden utilizar componentes (tanto de *hardware* como de *software*) de diferentes fabricantes, lo cual favorece la implantación de soluciones. El servidor presenta a todos sus clientes una interfaz única y bien definida. El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa. Muestran información de forma sencilla a través de un hipervínculo (Jeri 1998).

El esquema Cliente/Servidor también contribuye a una disminución de los costos de entrenamiento del personal, pues favorece la construcción de interfaces gráficas interactivas, las cuales pueden hacer que los "datos" se conviertan en "información" y además son más intuitivas y fáciles de usar.

Actualmente la gran mayoría de las aplicaciones de gestión empresarial hacen uso del modelo Cliente/Servidor, precisamente en el desarrollo de la aplicación se tendrá en cuenta por las siguientes razones:

- Petición de sistemas fáciles de usar
- Productividad y calidad

- Bajo costo y alto rendimiento
- Acceso fácil a la información
- Nuevas tecnologías y herramientas de alta productividad

### 1.4.2 Servidores *Web*

Un servidor *Web* es un programa que está en ejecución, y en espera de solicitudes de forma continua. Consta de un intérprete de Protocolo de Transferencia de Hipertextos (HTTP, por sus siglas en inglés) el cual se mantiene a la espera de peticiones y responde con el contenido solicitado. El cliente se encarga de interpretar el código y lo exhibe en pantalla (Quintanilla 2012). Entre los más utilizados internacionalmente se encuentran:

- Servidor HTTP Cherokee es un Servidor *Web* libre, multiplataforma, abierto bajo la licencia GPL. Apunta a ser un servidor *Web* bastante rápido que también soporta las funcionalidades más comunes de servidor. Está escrito completamente en C, es escalable y puede usarse como un Sistema integrado.
- *Internet Information Server*, IIS, es una serie de servicios para los ordenadores que funcionan con *Windows*. Originalmente era parte del *Option Pack* para *Windows* NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como *Windows* 2000 o *Windows Server* 2003. *Windows XP* Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.
- Apache Tomcat es un servidor *Web* de código libre robusto. El proyecto está dirigido y controlado por el Grupo Apache, que son un grupo de personas voluntarias de todo el mundo que planifican y desarrollan el servidor y la documentación relacionada. Además de este grupo, cientos de personas han contribuido con el proyecto (Garcías 2010).

Después de haberse realizado un análisis de las tecnologías existentes para la publicación de aplicaciones *Web* se decidió escoger como mejor alternativa al servidor *Web*, Apache Tomcat.

**Apache Tomcat**

Apache Tomcat es un servidor *Web* de *software* libre desarrollado por la *Apache Software Foundation* cuyo objetivo es servir o suministrar páginas *Web* (en general, hipertextos) a los clientes *Web* o navegadores que las solicitan.

Hoy en día Apache Tomcat es uno de los servidores *Web* más utilizado del mundo. Es un *software* de código abierto que funciona sobre cualquier plataforma. Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad. Surgió en abril de 1996 y ya en julio del 2002 era utilizado por el 57% de los sitios *Web* de *Internet*. La arquitectura utilizada es cliente/servidor. El protocolo para la transferencia de hipertexto es HTTP que está basado en el envío de mensajes y establece el conjunto de normas mediante las cuales se envían las peticiones de acceso a una *Web* y la respuesta de ella (Garcías 2010).

Apache Tomcat muestra una serie de características por las cuales ha sido uno de los principales servidores *Web* utilizados. A continuación se mencionan algunas características de este servidor *Web*:

- Su licencia es de código abierto del tipo BSD que permite el uso comercial y no comercial de Apache.
- Tiene una talentosa comunidad de desarrolladores siguiendo un proceso abierto de desarrollo.
- Arquitectura modular: los usuarios de Apache Tomcat pueden adicionar fácilmente funcionalidad a sus ambientes específicos.
- Portabilidad: Apache Tomcat trabaja sobre todas las versiones recientes de UNIX y Linux, *Windows*, BeOs, *mainframes*.
- Es robusto y seguro (García 2009).

Hoy en día Apache Tomcat se encuentra en la versión 8. Esta versión hace de Apache Tomcat una solución *Web* más flexible, transportable y escalable.

Para el desarrollo del Sistema Informático propuesto se hará uso de la arquitectura cliente-servidor donde se utilizara el Apache Tomcat para dar soporte al sistema por las ventajas antes mencionadas y ya que es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realiza se efectúe con la mayor eficiencia, y permite simplificar las actualizaciones y mantenimiento del sistema.

### 1.4.3 Sistemas Gestores de Bases de Datos

Entre los Sistemas Gestores de Bases de Datos (SGBD) comúnmente utilizados en el mundo se encuentran Oracle, MySQL, Microsoft SQL *Server*, PostgreSQL, InterBase, Microsoft Acces, IBM, DB2, entre otros. Los SGBD son *software* con capacidad para definir, mantener y utilizar una base de datos (BD), es decir un conjunto de datos comunes que se almacenan sin redundancia para ser útiles en diferentes aplicaciones (Mcgraw-hill 2011). Estos sistemas deben permitir definir estructuras de almacenamiento y acceder a los datos de forma eficiente y segura. Además deben proporcionar un mecanismo que asegure que la BD se actualice correctamente cuando varios usuarios lo están haciendo concurrentemente, de ahí que uno de sus principales objetivos sea permitir que varios usuarios tengan acceso concurrente a los datos que comparten (Rodríguez 2006).

Para seleccionar el SGBD más adecuado para el desarrollo de la solución propuesta, se realizó un estudio de aquellos que cumplieran la condición de ser de licencia libre dado el pedido del cliente y finalmente se decidió el empleo del gestor PostgreSQL por las características y ventajas que posee, las cuales se mencionan a continuación.

## PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada al *software* libre. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales, las cuales trabajan en su desarrollo. Dicha comunidad es denominada el Grupo Global de Desarrollo *PostgreSQL* (*PGDG* por sus siglas en ingles).

Soporta casi toda la sintaxis del lenguaje de consulta estructurado (*SQL* por sus siglas en ingles) y posibilita muchas características modernas tales como: consultas complejas, integridad referencial, vistas, integridad transaccional, control de concurrencia multi-versión. También soporta almacenamiento de objetos grandes (imágenes, sonido y video) (PÉREZ 2010).

Provee otras funcionalidades importantes tales como:

- Permite salvar el estado de la base de datos en momentos concretos, para su posterior recuperación.
- Posibilita destinar discos físicos a un índice o a una tabla concreta.
- Optimizaciones en la velocidad de ejecución y en el consumo de memoria de la aplicación.
- Cuenta con herramientas gráficas como PgAdmin, phpPgAdmin, las cuales hacen que la administración de la base de datos sea sencilla.
- Debido a su licencia libre PostgreSQL puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita para cualquier fin, ya sea privado, comercial o académico (Helmle 2010).

Son múltiples las ventajas de este gestor de Base de Datos, entre ellas están:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.

- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.
- Posibilidad de instalar un número ilimitado de veces sin temor de sobrepasar la cantidad de licencias, la principal preocupación de muchos proveedores de bases de datos comerciales.
- Velocidad y rendimiento excepcionales.
- Confiabilidad a toda prueba.
- Seguridad de primera clase.
- Flexibilidad para extenderse según se requiera.
- Bajo Costo Total de Operación (TCO) (Obando 2012).

Cada usuario obtiene una visión consistente de lo último a lo que se le hizo cambios. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases de datos, y se elimina la necesidad del uso de bloqueos explícitos.

### 1.4.4 Lenguajes de Programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (FERNANDEZ 2012).

En la actualidad existen numerosos lenguajes de programación para desarrollar aplicaciones informáticas, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. A medida que pasa el tiempo, las tecnologías han ido evolucionando y cada vez surgen nuevos problemas que requieren de una solución adecuada. Esto propició el desarrollo de lenguajes de

programación para las *Web*, que favorecieran la interacción con los usuarios y utilizaran sistemas Gestores de BD (Informático 2010).

Existen numerosos lenguajes de programación empleados para el desarrollo de aplicaciones *Web*, entre los que destacan: Java, con sus tecnologías Java Servlets y JSP, PHP, Perl, Ruby, Python, HTML, XML.

En la presente investigación se escogió el lenguaje Java para el desarrollo de la aplicación por el conjunto de características que posee el mismo, el cual se detalla a continuación.

**Java**

Java es un lenguaje sencillo de aprender, su sintaxis es la de C++ “simplificada”. Los creadores de Java partieron de la sintaxis de C++ y trataron de eliminar de este todo lo que resultase complicado o fuente de errores en este lenguaje (Bates 2005).

Entre las principales características que favorecieron el crecimiento y difusión del lenguaje Java podemos encontrar:

- Lenguaje simple

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir *applets* interesantes desde el principio. Todos aquellos familiarizados con C++ encontrarán que Java es más sencillo, ya que se han eliminado ciertas características (Álvarez 2010).

- Orientado a objetos

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red (James Gosling 2013).

- Distribuido

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

- Interpretado y compilado a la vez

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los *bytecodes*, semejantes a las instrucciones de ensamblador.

Por otra parte, es interpretado, ya que los *bytecodes* se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (*run-time*).

- Robusto

Java fue diseñado para crear *software* altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

- Seguro

Dada la naturaleza distribuida de Java, donde las *applets* se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

- Indiferente a la arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos

diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas *hardware* y *software*. El resto de problemas los soluciona el intérprete de Java (Álvarez 2010).

- Portable

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM) (James Gosling 2013).

***Java Database Connectivity***

Java *Database Connectivity*, más conocida por sus siglas JDBC, es una API (*Application Programming Interface*) que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, y usa el dialecto SQL del modelo de base de datos que se utilice (JDBC 2010).

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella para establecer una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tarea con la base de datos sobre la cual tenga permiso: consulta, actualización, creación,

modificación y borrado de tablas, ejecución de procedimientos de almacenado en la base de datos, etc. (Macedo 2013)

JDBC requiere que el programador defina e implemente las operaciones de persistencia, y use un patrón para la conversión de objetos a registros. Este proceso se realiza asignándole a cada objeto una tabla y cada campo de esa tabla representa un atributo del objeto, si se realiza un análisis profundo del tema, es evidente que el programador debe gastar mucho tiempo para desarrollar el proceso; además debe tener una vasta experiencia en lo que a lenguaje de consulta estructurado se refiere. JDBC no considera la transparencia de datos ni la persistencia por enlace, cuando se refiere a la persistencia por enlace o persistencia en profundidad, no es más que el proceso de convertir automáticamente en persistentes, todos los objetos referenciados directa o indirectamente por el objeto persistente. Además no contempla la integridad de las referencias, y debe ser el programador el que plasme la integridad referencial. Permite crear múltiples métodos para múltiple funcionalidad ya que JDBC ha preferido incluir gran cantidad de métodos, en lugar de hacer métodos complejos con gran cantidad de parámetros. Además es 100 % portable y tiene buen rendimiento (Páez 2010).

En el desarrollo de la aplicación *Web* se usará JDBC para la conexión y persistencia de los datos debido a que se tiene suficiente conocimiento de la misma, es flexible ya que permite consultas complejas y dará solución a la problemática del trabajo con la base de datos.

### 1.4.5 Frameworks

En el desarrollo de *software*, un *framework* es una estructura de soporte definida en la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, librerías y un lenguaje de *scripting* entre otros *software* para ayudar a desarrollar y unir los diferentes componentes de un proyecto. También ayudan a la hora de realizar el mantenimiento del sitio gracias a la organización durante el desarrollo de la

aplicación. Además, simplifican el desarrollo de las aplicaciones mediante la automatización de muchas de las tareas comunes. Un *framework* proporciona estructura al código fuente, y obliga al programador a crear código más legible y más fácil de mantener (Gutiérrez 2010).

Si se aplica la definición anterior al desarrollo *Web*, se puede llegar a la conclusión que un *framework Web* es una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones *Web*. En cierto sentido se puede afirmar que proveen una capa de abstracción sobre la arquitectura original ocultándola o adaptándola para no tener que utilizar el protocolo HTTP de manera nativa y así acelerar los tiempos de desarrollo y mantenimiento (Gutiérrez 2010).

En general, los *framework* están diseñados para potenciar el desarrollo de aplicaciones Web, ofreciendo funcionalidades generales o de uso específico que contribuyen con un desarrollo más rápido y de una calidad superior.

Algunos de los *frameworks* más utilizados en la actualidad son: Struts, Tapestry, ASP.NET, Cocoon, Grails, Ruby on Rails (ROR), JSF, ZK, Hibernate, Sinfony 2.

Para el desarrollo de esta aplicación, se escogió el uso del *framework* ZK debido a las características de este que a continuación se mencionan.

***Framework* ZK**

Para el desarrollo de esta aplicación, el uso del *framework* ZK es favorable, pues al incluir un lenguaje de marcado como ZUML, permite a los no expertos diseñar eficientemente interfaces de usuario, y aunque está basado en Ajax no es necesario que el desarrollador tenga conocimientos de este o de *JavaScript*. Posee un modelo basado en componentes intuitivo dirigido por eventos, y al permitir centrar toda la lógica de programación en el servidor no hay necesidad de programar manualmente código para ejecutarse del lado del cliente usando JavaScript, por lo tanto no se expone al cliente a la lógica del negocio de la aplicación (Johnson 2009).

ZK es un *framework* centrado en el servidor, con la tecnología java y las del cliente haciendo uso del lenguaje *JavaScript*, basado en componentes y conducido por eventos que permite interfaces de usuario enriquecidas para las aplicaciones *Web* (Pérez 2012).

- Incluye un motor manejado por eventos basado en *AJAX*, un conjunto rico de componentes XUL y XHTML y un lenguaje de marcado llamado Lenguaje de Marcación de Usuario de ZK (ZUML, por sus siglas en inglés). El cual está diseñado para que desarrolladores no expertos diseñen interfaces de usuario de forma eficiente. Permite a un desarrollador mezclar diferentes tipos de lenguaje de marcación, tales como el lenguaje XUL de Mozilla y XHTML, todos ellos en la misma página, además de embeber scripts en lenguaje Java (interpretado por *BeanShell*) y usar expresiones para manipular los componentes y acceder a los datos.
- Se puede representar una aplicación en componentes XUL y XHTML ricos y especiales, y manipularlos sobre eventos lanzados por la actividad del usuario, similar a lo que es hecho en las aplicaciones desktop.
- A diferencia de otros *frameworks* AJAX, en lo que concierne a ZK, AJAX es una tecnología detrás de la escena. La sincronización del contenido de componentes y el oleoducto de eventos son hechos automáticamente por el motor de ZK (Gracia 2010).
- EL Cargador de ZK está basado en la petición del usuario, este carga una página, la interpreta, y renderiza el resultado en páginas HTML en respuesta a las peticiones de URL.
- El Motor de Actualización Asíncrona de ZK (ZK AUE, por sus siglas en inglés) trabaja junto al Motor del Cliente de ZK (ZK CE, por sus siglas en inglés) como lanzador y receptor. Ellos envían los eventos que ocurren en el navegador a la aplicación que corre en el servidor, y actualizan el árbol DOM en el navegador basado en como los componentes son

manipulados por la aplicación. Este es el tan mencionado Modelo de Programación Conducido por Eventos.

Otra característica de ZK es el uso de XML Lenguaje de Interfaz de Interusuario (ZUL, por sus siglas en inglés) como lenguaje de descripción de formas gráficas. (ZK se refiere al lenguaje de descripción como “ZUL”.) ZUL nos permite definir formas como documentos XML en donde las etiquetas individuales corresponden a controles sobre un formulario, simplificando el proceso de diseñar interfaces *Web*. Podemos también crear formularios en Java utilizando una API dedicada, algo así como la biblioteca *Swing* (Pérez 2012).

ZK posee un mecanismo que le permite al desarrollador construir aplicaciones usando el patrón Modelo Vista Controlador (MVC). Este mecanismo separa por capas el modelo, la vista y el controlador de una manera realmente genial (Pérez 2012).

- La vista puede crearse haciendo uso de los lenguajes java o ZUML, presentando el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- El modelo usa *beans* que contienen información de diversas fuentes de datos, siendo la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- El controlador responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Se crea al heredar de diferentes clases útiles que implementan de la interfaz, estas clases permiten un acceso directo a los componentes, objetos embebidos, y variables externas creadas en la vista (página zul), y la creación de métodos intuitivos (Gracia 2010).

### 1.4.6 Patrón de Diseño Modelo-Vista-Controlador aplicado a la Web

Uno de los patrones más conocidos en el desarrollo *Web* es el patrón Modelo-Vista-Controlador (MVC). Este patrón permite y obliga a separar la lógica de control, la lógica de negocio y la lógica de presentación (Andalucía 2012). Para comprender cómo trabajan los *frameworks Web* existentes es imprescindible conocer el patrón MVC, el mismo será utilizado para la realización del sistema informático.

El patrón Modelo-Vista-Controlador es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios. Este patrón organiza la aplicación en tres modelos separados: el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio; el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información; el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema (González 2012).

## 1.5 Herramientas de desarrollo

Para el desarrollo del sistema propuesto como solución se utilizó un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés, *Integrated Development Environment*). Un IDE no es más que un programa informático compuesto por un conjunto de herramientas de programación, que puede dedicarse en exclusiva a un sólo lenguaje de programación o bien permitir utilizar varios. Para el desarrollo del sistema informático propuesto se propone el uso del IDE Eclipse, para le presentación de reportes se propone el uso del *iReport* y el *JasperReport*.

### 1.5.1 Entorno de Desarrollo Integrado Eclipse

Eclipse es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce

como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios *Web*, programas en C++ o aplicaciones Java. No es más que un entorno de desarrollo integrado (IDE, por sus siglas en inglés) en el que encontrarás todas las herramientas y funciones necesarias para tu trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar. (Eclipse 2005)

Eclipse provee al programador con *frameworks* muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de *Software*, Aplicaciones *Web*. El entorno de desarrollo integrado (IDE, por sus siglas en inglés) de Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la Plataforma de Cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.

Este mecanismo de módulos es una plataforma ligera para componentes de *software*. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos.

### 1.5.2 iReport + Jasper Report

La herramienta *iReport* es un constructor/diseñador de informes visuales, poderoso, intuitivo y fácil de usar para *JasperReports, JasperReports* es una biblioteca de clases escrita en Java la cual permite generar informes. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. *iReport* está además integrado con *JFreeChart*, una de la biblioteca gráficas *OpenSource* más difundida para Java. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC*, TableModels, JavaBeans, XML*, etc. (Herrera 2013).

Esta herramienta es además *opensource*. Maneja el 98% de las etiquetas de *JasperReports*. Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de los *textfields*, cartas, *subreports* (subreportes).

Este editor gráfico se utilizó para el diseño de los diferentes reportes junto a la biblioteca de clases *JasperReport* para interpretar los archivos generados en dicho editor en la aplicación. El funcionamiento se basa en escribir un XML donde se recogen las particularidades del informe que se necesita. Este XML lo tratan las clases del *Jasper* para obtener una salida. Esta salida puede ser en formato PDF, XML, HTML, CSV, XLS, RTF, TXT, etc. Lo que permite que el usuario obtenga un reporte de acuerdo a sus necesidades y lo vea o imprima si desea en cualquier formato (Herrera 2013).

## 1.6 Metodologías de desarrollo de *software*

En el proceso de desarrollo de *software* intervienen innumerables variables de las más diversas naturalezas, algunas con comportamiento sumamente difuso o imprevisible, que minan constantemente el avance hacia el éxito y hacen de sus rutinas y decisiones tareas altamente riesgosas, difíciles para controlar la calidad y eficiencia y cuantificar su eficacia (Quintero, 2010).

Desde hace bastante tiempo existe una alternativa: el uso de una metodología, que no es más que un conjunto de procedimientos para la realización de un nuevo *software* (Burbano 2009), que impone un proceso disciplinado con el fin de hacerlo más predecible y eficiente. Consiste en un lenguaje de modelamiento y un proceso (Burbano 2009).

El lenguaje de modelamiento es la notación gráfica, que incluye diferentes tipos de diagramas. El proceso define quién debe hacer qué, cuándo y cómo alcanzar un objetivo (Sánchez 2004).

Actualmente existen metodologías que permiten desarrollar *software* de superior calidad, debido a la facilidad de control que éstas proporcionan, aparejado a la posibilidad de concebir inicialmente las bases para el desarrollo del *software* y su éxito en el tiempo y costo fijados. Cada metodología, a pesar de suministrar los aspectos antes mencionados, posee características peculiares que facilitan la

selección de una de éstas a partir de las necesidades de la organización y las características específicas del proyecto a desarrollar.

En los últimos tiempos han cobrado auge las metodologías ágiles, debido a que cada vez más los desarrolladores necesitan obtener aplicaciones en menor tiempo, más vistosas y de menor costo; y los usuarios exigen calidad, sistemas fáciles de mantener, extender y modificar (Sánchez 2004).

El objetivo principal de las metodologías ágiles es minimizar la documentación de desarrollo, considerando el código fuente la parte más importante y el *software* que funciona como la principal medida del progreso. Considera a los usuarios finales como parte del equipo de desarrollo, por la importancia que concede al trabajo conjunto de ambos durante todo el proceso (Canos 2008).

Surgen como reacción a las metodologías monumentales o pesadas, ya que éstas últimas son muy burocráticas; hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda (Canos 2008). Son adaptables en lugar de predictivas, por lo que aceptan el cambio como bienvenido, se adaptan y crecen en el cambio, incluso al punto de cambiarse ellas mismas. Son orientadas a la gente y no orientadas al proceso, por lo que afirman que el papel del proceso es apoyar al equipo de desarrollo en su trabajo y no al contrario (Canos 2008).

Dentro de estas metodologías, algunas de las más nombradas en la literatura son:

- XP (*eXtreme Programming*, por sus términos en inglés), con gran énfasis en las pruebas.
- Scrum, que se enfoca principalmente en la planeación iterativa y el seguimiento del proceso, generalmente para proyectos de equipos.
- ICONIX, relativamente ágil y lo suficientemente robusta para un proyecto de mediana envergadura.

Teniendo en cuenta lo anteriormente planteado, la cantidad de bibliografía disponible y que el proyecto es de mediana envergadura y cuenta con un solo desarrollador, se determinó escoger para el desarrollo del proyecto la metodología ICONIX, proceso simplificado (comparado con otros más

tradicionales) de desarrollo de *software*, orientado a objeto que emplea el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Model Language*). Está entre la complejidad de RUP (*Rational Unified Process*, por sus términos en inglés), que es una de las metodologías tradicionales y la simplicidad de XP, sin descartar las etapas de análisis y diseño; siendo el primero muy útil para *software* industriales y el segundo muy útil para *software* pequeños; por tanto, ICONIX es una mezcla entre la agilidad de XP y la robustez de RUP.

### 1.6.1 ICONIX

El ciclo de vida de un proyecto, según ICONIX, se divide en cuatro fases principales: Definición de Requerimientos; Análisis, Diseño Conceptual y Arquitectura Técnica; Diseño detallado e Implementación; y Prueba, (**Figura 1.3**) la cual se puede añadir a conveniencia del desarrollador y teniendo en cuenta las características propias del sistema (González 2009).

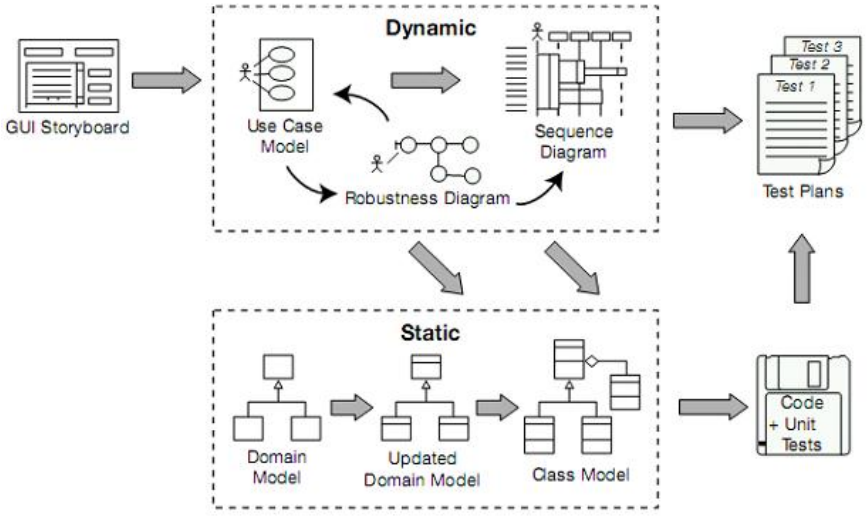



**Figura 1.3: Fases de ICONIX**

El proceso de ICONIX es un modelado de objetos conducido por casos de uso, como RUP; también es relativamente pequeño y firme, como XP, pero no desecha el análisis y diseño de éste. Este proceso también hace uso dinámico

del UML, ya que se pueden usar solo algunos diagramas, sin exigir la utilización de todos como en el caso de RUP; aun así es posible seleccionar otros aspectos del UML para complementar los materiales básicos. Esto brinda un enfoque flexible y abierto mientras que guarda un enfoque afilado en el seguimiento de requisitos (Peña 2010).

Otras de las características más importantes son: el enfoque iterativo e incremental, ya que se pueden refinar las distintas fases a medida que se vayan identificando nuevos objetos; y la trazabilidad, permitiendo seguir cada caso de uso durante todo el proceso. Además, es centrado en datos, es decir, se descompone en fronteras de datos y es basado en escenarios que descomponen los casos de uso.

Las características antes mencionadas garantizan que la metodología ICONIX sea la apropiada para llevar a cabo el proyecto que se emprende.

## 1.7 Conclusiones del capítulo

En este capítulo se realizó un estudio de los fundamentos teóricos que sustentan la gestión de información de la valoración económica de proyectos en la ENPA. A partir de esto se definieron los conceptos principales asociados al problema y al objeto de estudio, y se escogieron las herramientas y tecnologías para el desarrollo de la aplicación, teniendo en cuentas sus ventajas y desventajas.

Por otra parte, se resaltaron las características, ventajas y desventajas de las tecnologías de desarrollo para una adecuada elección y combinación de éstas, eligiéndose el *framework* ZK para el desarrollo de las páginas Web enriquecidas, se optó por la opción de utilizar PostgreSQL como servidor de bases de datos, y Apache Tomcat como servidor *Web*, como lenguaje de programación Java y para la construcción y presentación de los reportes se eligieron el *JasperReport* en conjunto con el *iReport*. Además, se decidió el empleo de la metodología ICONIX para el desarrollo del sistema por las facilidades que ésta brinda.

# Capítulo 2. Descripción y Elaboración del sistema

En este capítulo se realiza una descripción de la solución propuesta para cumplir el objetivo de la investigación. Se tratan temas relacionados con el análisis, diseño y desarrollo de la aplicación *Web* propuesta, basado en los requisitos funcionales y no funcionales del mismo. Para ello se utiliza la metodología ICONIX y los artefactos que ella proporciona. Además, se detalla el procedimiento de valoración de sostenibilidad realizado a la aplicación *Web* resultante del proceso de desarrollo de *software* aquí descrito, así como se hace una exposición de los resultados que se alcanzaron con la implementación de la aplicación, a través de encuestas y entrevistas que midieron la satisfacción de los usuarios del sistema con respecto al mismo.

## 2.1 Análisis de Requisitos

El proceso de captura de requerimientos se realiza para conocer las principales funcionalidades que debe cumplir un sistema. Este proceso permite proporcionar un entendimiento común entre los actores y los desarrolladores del sistema, acerca de los elementos que serán tenidos en cuenta en la construcción del sistema informático. Además, es la base para la obtención de los casos de uso del sistema.

### 2.1.1 Requerimientos del Sistema

Los requisitos o requerimientos se pueden clasificar en: funcionales y no funcionales. Los requerimientos funcionales son condiciones que el sistema debe ser capaz de realizar. Describen el comportamiento de las entradas para producir salidas y surgen de la razón fundamental de la existencia del producto. Los requerimientos no funcionales se refieren a las cualidades del sistema. Se precisan con la intención de alcanzar un *software* de calidad, y la aceptación total de los usuarios finales (Peña 2010).

## Requerimientos Funcionales

El sistema deberá ser capaz de:

1. Permitirle a los usuarios autenticarse para entrar al sistema
2. Permitir la validación de los usuarios
3. Permitirle al Administrador gestionar los usuarios del sistema
4. Permitirle  a los usuarios cerrar sesión
5. Permitirle al Administrador gestionar **cargos**
6. Permitirle al Administrador gestionar **departamentos**
7. Permitirle al Administrador gestionar **insumos** y sus **unidades de medida (UM)**
8. Permitirle al Administrador gestionar **proyectistas**
9. Permitirle al Administrador gestionar **proyectos**
10. Permitirle al Jefe de departamento gestionar valoraciones de proyectos
11. Permitirle al Jefe de Departamento buscar proyectos
12. Permitir generar reportes

## Requerimientos no Funcionales

***Apariencia o interfaz externa***

La interfaz de la herramienta en su conjunto debe ser muy asequible, simple de usar, intuitiva y sugerente, a través de elementos visibles que identifique cada acción facilitándole al usuario la navegación por la misma.

El color predominante en el diseño será el verde, cumpliendo con los estándares del manual de identidad de la empresa.

Los botones e Iconos expresan acciones posibles a realizar (insertar, editar, eliminar etc.), se hace uso de imagen que los identifiquen, de forma tal que signifique la acción que deben ejecutar.

En caso de no poder ejecutar una acción por un usuario, visualizar un mensaje de error que especifique por qué no se pudo ejecutar.

***Hardware***

Para ejecutar el *software* los requerimientos mínimos de *hardware* son: microprocesador Intel Pentium III a 1 GHz de velocidad de procesamiento u otro similar, con 512 MB de memoria RAM y una tarjeta de red. El sistema está basado en la arquitectura cliente-servidor, y la máquina computadora donde esté alojada la aplicación debe estar conectada a la red.

***Software***

Para la confección del producto se utilizará el IDE Eclipse y el *framework* ZK para el desarrollo de las páginas *Web*; como gestor de Base de Datos Postgres SQL; servidor Windows con Apache Tomcat para la publicación del Sitio, como navegador puede utilizarse *Internet Explorer*, Mozilla FireFox, entre otros.

**Portabilidad**

Las herramientas utilizadas para el desarrollo del sistema son tecnología de *software* libre y a su vez multiplataforma, lo cual le confiere al sistema esta última característica.

**Ayudas del Sistema**

Debe contar con un Manual de Usuario y una Ayuda de forma tal que en todo momento le permita al usuario orientarse respecto a las opciones que le brinde el sistema.

**Usabilidad**

El sistema debe permitir ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente *Web* en sentido general.

**Seguridad**

La seguridad constituye un factor de vital importancia por lo que el sistema debe permitir un acceso diferenciado a la información que brinda, debe garantizar que cada usuario pueda introducir, editar o eliminar la información que le corresponde gestionar, según los roles de seguridad definidos para el sistema.

Debe ser posible identificar en cada momento el usuario que intenta realizar una acción y comprobar que tiene suficientes privilegios para llevarla a cabo.

**Rendimiento**

Al ser una aplicación cliente/servidor los tiempos de respuestas deben ser rápidos, así como la velocidad de procesamiento de la información. Además el tiempo de respuesta debe ser corto, por lo que el acceso a la base de datos se debe efectuar de forma rápida.

## 2.2 Modelo del Dominio

La Metodología Iconix propone realizar un modelo del dominio que no es más que un artefacto que refleja de forma visual los términos principales del problema de la investigación, es decir la construcción de un glosario de términos (diccionario de palabras) de las clases conceptuales u objetos del mundo real, que permite identificar algunas clases que se utilizarán en la construcción del sistema. Un modelo del dominio también proporciona un vocabulario común para favorecer la comunicación clara entre los miembros de un equipo (DOUG ROSENBERG 2005).

En la siguiente **Tabla 2.1** se muestran los conceptos más significativos para el desarrollo de la aplicación *Web* propuesta.

**Tabla 2.1 Definición de objetos del dominio**

| Concepto | Definición |
|---|---|
| **Cargo** | Es la categoría ocupacional que tendrán los proyectistas |
| **Departamento** | Unidad organizativa a la cual pertenecen los proyectistas |
| **Proyecto** | Producto que comercializa la entidad |
| **Proyectista** | Especialistas que tendrán participación en un momento dado en los proyectos |

| | |
|---|---|
| **Insumo de Materiales** | Materias primas consumidas en la realización de los proyectos |
| **UM** | Unidad de medida de los insumos de materiales |

En la **Figura2.1** se muestra mediante un diagrama de clases las relaciones entre los conceptos definidos anteriormente.

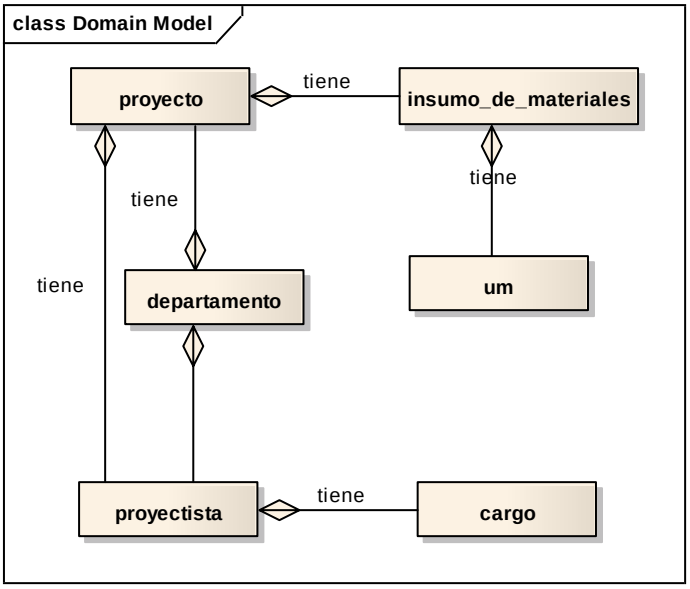



**Figura 2.1: Modelo del Dominio**

### 2.2.1 Modelo de Casos de Uso del Sistema

En esta etapa se precisan cuáles son las funcionalidades del sistema propuesto. Para ello se utiliza el artefacto UML modelo de casos de uso.

Los casos de uso del sistema son una técnica para especificar el comportamiento del *software* y se parte de la identificación de los requerimientos del sistema. El modelo de casos de uso describe lo que hace el sistema para el usuario; la forma en que éste usa el sistema se representa con casos de uso, derivados de los requisitos funcionales que los relacionan. La representación de cada caso de uso facilita especificar la secuencia de acciones que el sistema

puede llevar a cabo interactuando con el o los actores, incluyendo alternativas dentro de la secuencia, ver (**Figura 2.2**) (Stephens 2007).

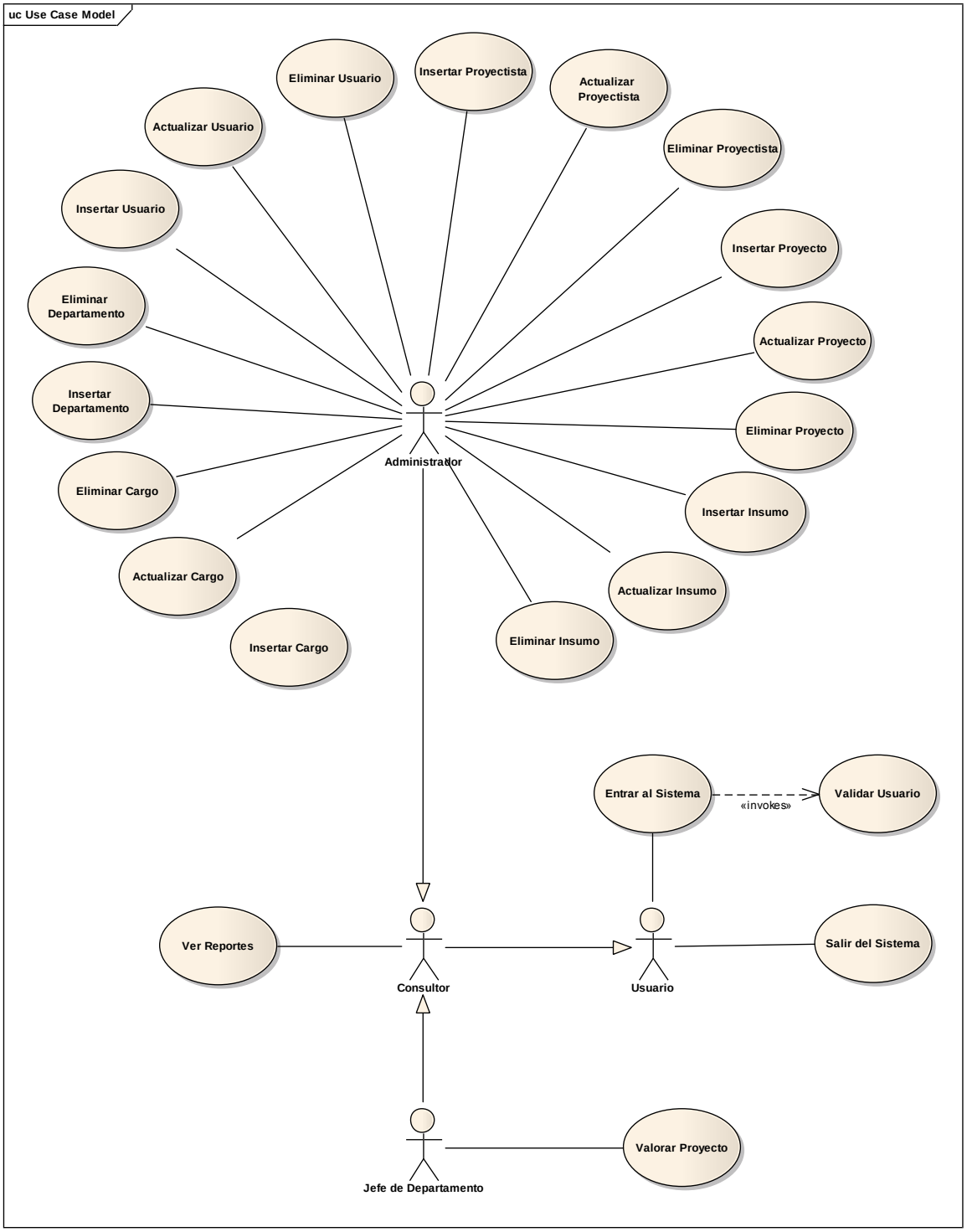



**Figura 2.2: Diagrama de Casos de Uso del Sistema**

### Actores del Sistema

Un actor es un rol que alguien o algo juega cuando interactúa con el sistema para beneficiarse de sus resultados. Los actores pueden ser Clientes o potenciales clientes, Socios, Proveedores, Autoridades, Propietarios, Sistemas de información externos al negocio u otras partes de la organización, si esta es grande. Los actores del sistema no son parte de él. Pueden intercambiar información con él, pueden ser reciente pasivo de información y pueden representar el rol que juega una o varias personas, un equipo o un sistema informatizado. Cada diagrama de caso de uso es iniciado por un actor, este no es más que una figura y el análogo a un rol que los usuarios pueden jugar (ver **tabla 2.2**), muchas veces el actor es llamado como ”Usuario”, aunque frecuentemente le es dado el nombre del rol específico que desempeña en el sistema (Rosenberg 2001).

**Tabla 2.2: Actores del Sistema**

| Actores del Sistema | Descripción |
|---|---|
| **Administrador** | Será el usuario encargado de la administración del sistema |
| **Jefe de Departamento** | Usuario con permisos para gestionar las valoraciones de su departamento |
| **Consultor** | Usuario con privilegios de solo lectura |
| **Usuario** | Persona que usará el sistema |

### 2.2.2 Descripción Textual de los Casos de Uso del Sistema

Las descripciones de los casos de uso (CUS) se escriben en párrafos simples, en perspectiva del usuario usando la voz pasiva, siguiendo el flujo acción del

actor/ respuesta del sistema, donde se describe lo que hace el usuario y la respuesta que da el sistema. En las cuales se reflejan dos cursos, un curso básico que muestra cómo se debe realizar el proceso del caso de uso y un curso alterno para describir los posibles errores que se pueden presentar durante su ejecución, es decir lo contrario de la funcionalidad.

A continuación se muestra las descripción textual del CUS Insertar Cargo (ver **tabla 2.3**), el resto se puede encontrar en los anexos (ver Anexo 1).

**Tabla 2.3 Descripción Textual del CUS Insertar Cargo**

| Caso de Uso: Insertar Cargo | |
|---|---|
| **Curso básico** | El **Administrador** da clic en el menú Inicio y dentro de él va al submenú Cargos donde escoge la opción Insertar cargos y el sistema muestra la ventana de Crear nuevo cargo con datos a llenar (nombre, salario básico), al ser llenados, el usuario da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos y que el nombre del cargo no se encuentre ya creado, almacena los datos y muestra un mensaje de que el Cargo ha sido insertado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Ya existe el cargo: El sistema muestra un mensaje de que el cargo ya existe. |

## 2.3 Análisis y Diseño Preliminar

En el análisis y diseño preliminar se realizan los diagramas de robustez a cada caso de uso.

### 2.3.1 Análisis de Robustez

El análisis de robustez está seriamente relacionado a la descripción textual de los casos de uso, ayuda a refinar el texto y a identificar los objetos que participan en cada uno de ellos. Además, es una herramienta para saber si las

especificaciones del sistema son razonables. Este proceso se realiza mediante la construcción de un diagrama de robustez, para cada caso de uso que permite reflejar gráficamente las interacciones entre los objetos participantes en cada uno.

El análisis de robustez ayuda a asegurar que se han identificado la mayoría de las clases del dominio antes de empezar los diagramas de secuencia. Los tres estereotipos usados durante el análisis de robustez son:

**Los Objetos Interfaz** son los objetos con que los actores interactúan recíprocamente en el nuevo sistema, generalmente como ventanas, pantallas, diálogos y menús.

**Los Objetos Entidad** trazan a menudo las tablas de la base de datos y archivos que contienen la información que necesita sobrevivir a " la ejecución de caso de uso".

**Los Objetos Control** (controles) incluyen la lógica de la aplicación y sirven como el camino que une entre los usuarios y los datos guardados. Es la unión entre los objetos interfaz y entidad. Los controladores son " los objetos reales" en un modelo, pero los controladores normalmente sirven como el guía para asegurar que no se olvide de cualquier funcionalidad y la conducta del sistema requerido por sus casos de uso (Stephens 2007).

A continuación se presenta el diagrama de robustez para el CUS Insertar Cargo (ver **Figura 2.3**, ver el resto de los diagramas en el Anexo 2).

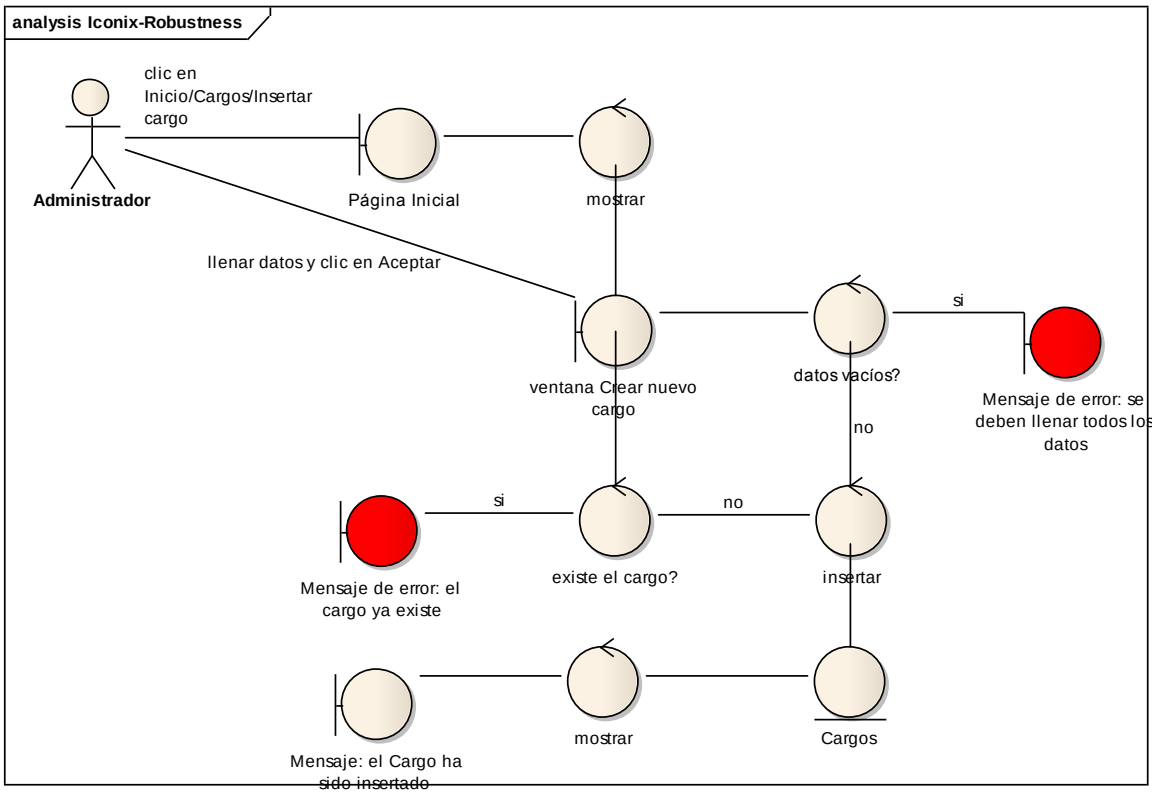



**Figura 2.3: Diagrama de Robustez del CUS Insertar Cargo**

## 2.5 Diseño e Implementación del Sistema

En la fase de diseño se muestran los diagramas de secuencia asociados a los principales casos de uso del sistema, los cuales reflejan el comportamiento de las clases.

### 2.5.1 Diagrama de Secuencia

El diagrama de secuencia es el núcleo del modelo dinámico y muestra todos los cursos alternos que pueden tomar los casos de uso. Se compone de 4 elementos: curso de acción, objetos, mensajes y métodos. Tiene tres objetivos elementales: asignar comportamiento a las clases, mostrar en detalle cómo las clases interactúan entre sí durante el tiempo de vida del caso de uso y terminar la distribución de las operaciones entre clases (Stephens 2007). La **Figura 2.4**

muestra el diagrama de secuencia del CUS Insertar Cargo, el resto de los diagramas se muestran en el Anexo 3.

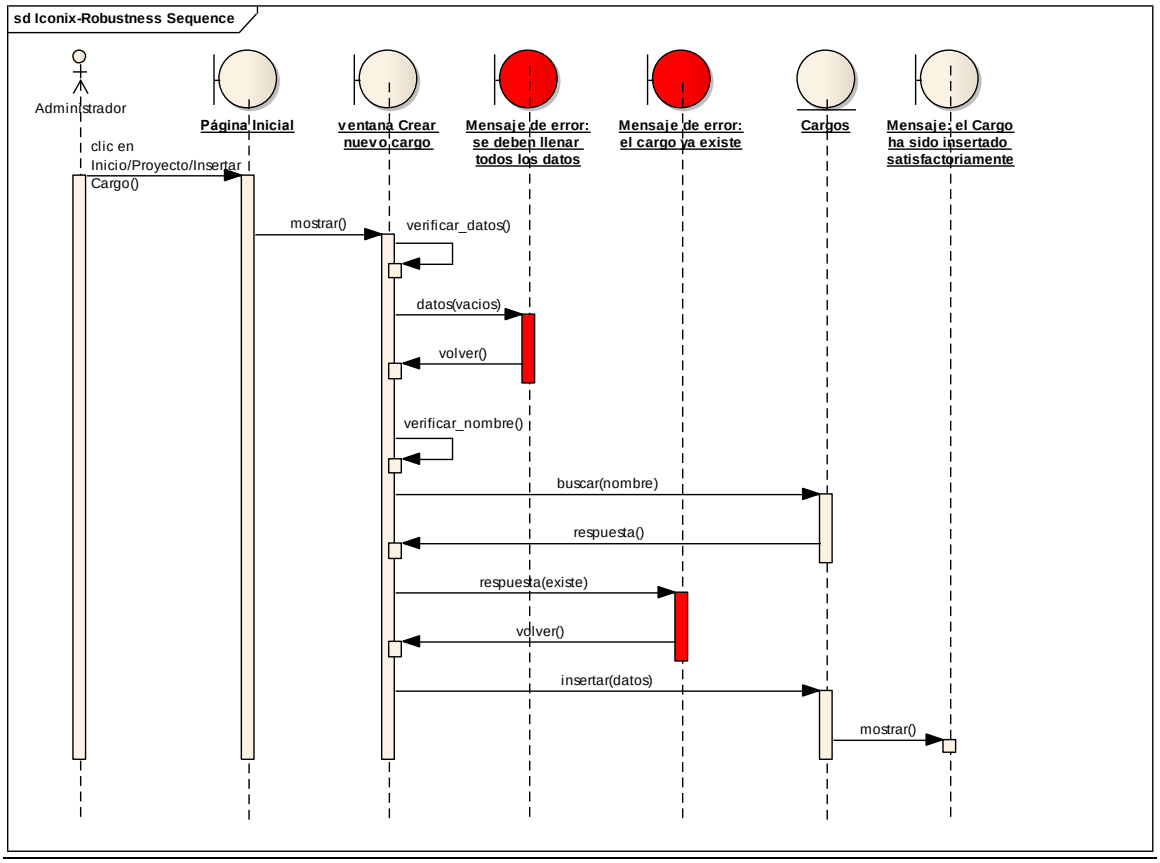



**Figura 2.4: Diagrama de Secuencia del CUS Insertar Cargo**

### 2.5.2 Diagrama de Clases Persistentes

El diagrama de clases persistentes representa las tablas y campos de las mismas, así como sus relaciones. Las clases persistentes son las clases que son capaces de guardar su estado en un medio permanente y representan además un modelo lógico de la base de datos, formado por las tablas que permanecen en la misma. El diagrama de clases persistentes del sistema modela la información que trasciende en el tiempo, incluso después de cerrada la aplicación (Ver Anexo 4).

Al culminar esta fase se muestra una idea de cómo se puede implantar el sistema, cómo funciona, a través de la distribución física.

### 2.5.3 Diagrama de Despliegue

El diagrama de despliegue, el cual representa las principales capas que conforman la aplicación, y es un modelo de objetos que describe la distribución física del sistema, permite distribuir las funcionalidades entre nodos. Conjuntamente es una colección de nodos y arcos; donde cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. La **Figura 2.5** muestra el diagrama de despliegue del sistema informático propuesto.

.

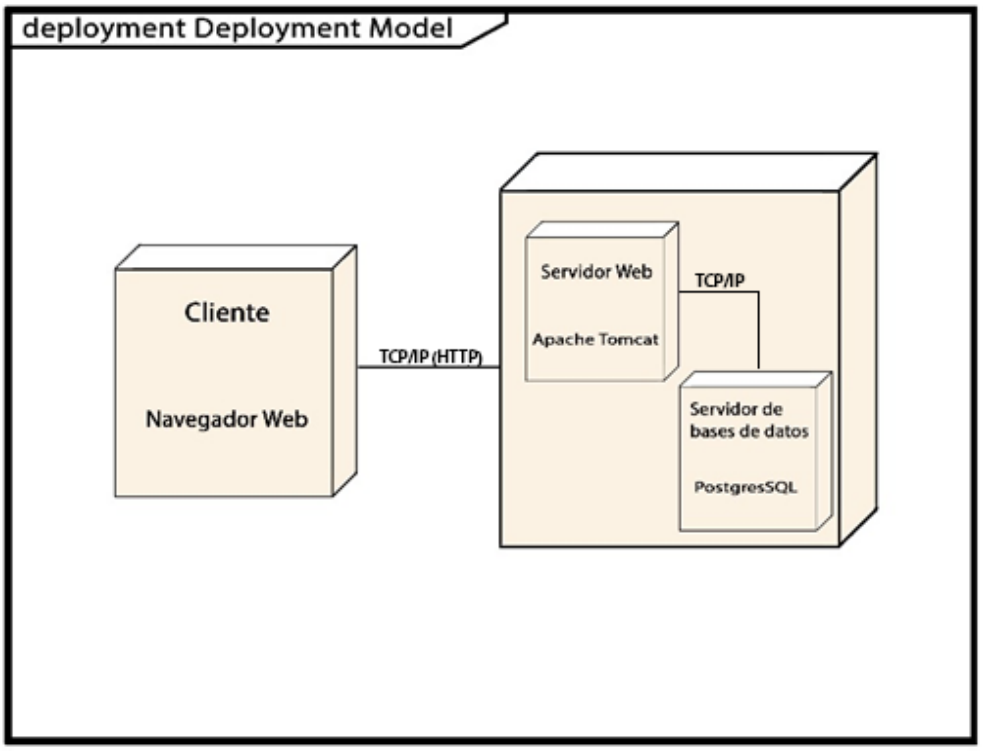



**Figura 2.5: Diagrama de Despliegue**

### 2.5.4 Diagrama de clases

El Diagrama de  Clases define un diagrama que muestra todas las clases del sistema. Iconix propone realizarlo después de los diagramas de secuencia,

como una última actualización del Modelo del Dominio con los nuevos objetos o clases identificadas. El diagrama se muestra en el Anexo 5.

### 2.5.5 Estándar de Código

Uno de los factores que definen la calidad de un *software* es la facilidad de mantenimiento, esta fase del *software* resulta ser la más compleja y costosa en el proceso de desarrollo del mismo. Una de las tareas que implica el mantenimiento del *software* en su fase inicial, es la correcta comprensión del código que fue escrito por otras personas. Si el código está poco documentado y no se utilizó un estándar para desarrollarlo, implica que aumente la complejidad para su mantenimiento, por lo que resulta sumamente engorroso. Para el desarrollo de la aplicación se hizo uso de un estándar de código (Ver Anexo 6).

### 2.5.6 Tratamiento de Errores

En cada página de la aplicación se garantiza que cuando se produzca alguna excepción el usuario del sistema sea advertido mediante mensajes que se muestran en la página. Estos mensajes también son mostrados en las interfaces de entrada de datos, donde se valida que el usuario no entre datos incorrectos y que no deje campos en blanco entre otros.

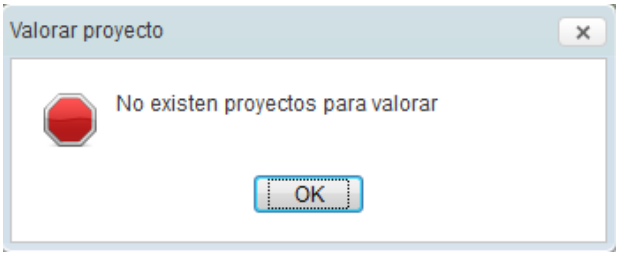



**Figura 2.6: Mensaje de error al intentar crear una valoración y no existen proyectos sin valorar**

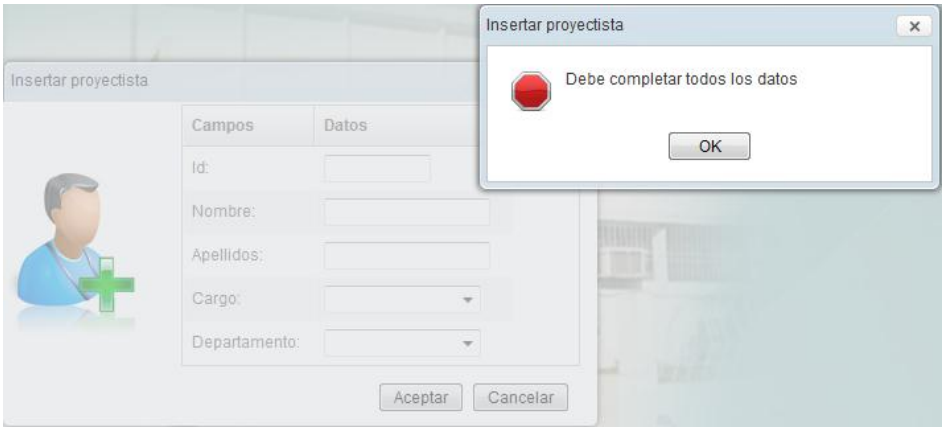



**Figura 2.7: Mensaje de error al intentar insertar dejando campos vacíos**

## 2.6 Pruebas

La fase de prueba según ICONIX debe comenzarse antes de la implementación. Los preparativos para la prueba comienzan desde la etapa de análisis, identificando los casos de prueba a partir de los diagramas de robustez, los cuales son codificados durante la implementación. La realización de la prueba en fases tempranas permite eliminar gran cantidad de errores, incluso antes de que existan.

Las pruebas están estrechamente relacionadas con los requerimientos, ya que tienen como objetivo satisfacer a los mismos. Debe haber al menos una prueba que asegure que cada requerimiento ha sido implementado correctamente, para lograr esto ICONIX propone algunos tipos de prueba y cuándo usarlos.

Para cada caso de uso se ejecutaron varias pruebas, comprobando que se diera cumplimiento a cada requerimiento del sistema, para lo que se utilizaron datos reales. Los problemas detectados como resultado de las pruebas realizadas se tomaron en cuenta y a la vez fueron corregidos en iteraciones posteriores.

## 2.7 Valoración de Sostenibilidad

La valoración de sostenibilidad de un producto informático no es más que el proceso de evaluación de impactos ambientales, socio humanistas,

administrativos y tecnológicos del mismo, previsibles desde el diseño del proyecto, que favorece su autorregulación, para la satisfacción de la necesidad que resuelve, con un uso racional de recursos y la toma de decisiones adecuadas a las condiciones del contexto y el cliente (Concepción 2006).

Con el propósito de favorecer el proceso productivo de la Empresa Nacional de Proyectos e Ingeniería ENPA, se tuvo en cuenta que el sistema informático propuesto como solución fuera sostenible desde las dimensiones administrativa, socio-humanista, ambiental y tecnológico.

Este procedimiento se realiza antes de la etapa de análisis, para conocer tempranamente si es sostenible y factible la solución que se propone antes desarrollarla. Se explica en este epígrafe para no entorpecer el flujo de la metodología de Ingeniería de *Software* empleada.

**Dimensión Administrativa**

En la dimensión administrativa se valora si la solución planteada ahorra recursos. Se tienen presente los gastos implicados para desarrollarla e implantarla, la calidad de la producción y los servicios, la administración de recursos y la toma de decisiones administrativas, de manera que se garantice la sostenibilidad administrativa del sistema.

Para el desarrollo del sistema propuesto se estimaron los valores de costo, tiempo y recursos requeridos, para lo que se recurrió al Modelo Constructivo de Costos (*COCOMO* II, por sus siglas en inglés, *Constructive Cost Model*), que posibilitó realizar el análisis de factibilidad. La factibilidad de un proyecto esta dada por la determinación de la posibilidad de hacer según restricciones (tiempo, presupuesto, etc.) identificadas y aprobadas teniendo en cuenta los criterios organizativos, económicos, técnicos y de tiempo (Cocomo 2000).

*COCOMO* II es un modelo que permite estimar el coste, esfuerzo y tiempo asociados a la construcción de un *software*. Este modelo incluye los siguientes pasos: obtener los puntos de función (*UFP*, por sus siglas en inglés), estimar la cantidad de instrucciones fu ente (*SLOC*, por sus siglas en inglés) y aplicar las fórmulas de Boehm (Boehm 1981).

Para obtener los puntos de función es necesario identificar las siguientes características del sistema: Entradas Externas (EI, por sus siglas en inglés), Salidas Externas (*EO*, por sus siglas en inglés), Consultas Externas (*EQ*, por sus siglas en inglés), Ficheros Lógicos Internos (*ILF*, por sus siglas en inglés) y Ficheros de Interfaz Externa (E I F, por sus siglas en inglés) (Ver Anexo 7).
Luego se cuenta la cantidad de funciones de característica por cada nivel de complejidad y se multiplica por el peso asociado en la **Tabla 2.4**. Todos estos productos se suman y se obtiene la cantidad de puntos de función desajustados. El sistema propuesto no tiene *EQ* (salidas asociadas al sistema que no tienen elementos de filtraje de información) ni *ILF*.

**Tabla 2.4: Puntos de función desajustados (UFP)**

| Elementos | Simples | Peso | Medios | Peso | Complejos | Peso | Subtotal |
|---|---|---|---|---|---|---|---|
| **EI** | 21 | 3 | 0 | 4 | 0 | 6 | 63 |
| **EO** | 3 | 4 | 1 | 5 | 0 | 7 | 17 |
| **EQ** | 0 | 3 | 0 | 4 | 0 | 6 | 0 |
| **ILF** | 10 | 7 | 0 | 10 | 0 | 15 | 70 |
| **EIF** | 0 | 5 | 0 | 7 | 0 | 10 | 0 |
| **Total** | 34 | | 1 | | 0 | | 150 |

La estimación de la cantidad de líneas de código fuente (SLOC, por sus siglas en inglés) del proyecto se basa en la cantidad de líneas de código por punto de función del lenguaje a usar en la implementación del sistema. Los lenguajes de programación a utilizar son: Java y SQL. Se estima que del empleo total de código, el 90 % es de Java y el 10% de SQL. (Ver **Tabla 2.5**)

**Tabla 2.5: Cantidad de Líneas de Código Fuente**

| Características | Valor |
|---|---|
| **Lenguaje Java (90%)** | SLOC = UFP * Ratio * Porciento<br>SOLC = 150 * 55 * 90% = 7425,00<br>KSLOC = 7,425 |
| **Lenguaje SQL (10%)** | SLOC = UFP * Ratio * Porciento<br>SLOC = 150 * 31 * 10% = 465,0<br>KSLOC = 0,465 |
| **KSLOC Total (Líneas de Código Fuente en miles)** | KSLOC (Java) + KSLOC (SQL) =7,89 |

Para determinar el esfuerzo asociado al desarrollo del sistema (PM), el tiempo de desarrollo (TDEV) y el costo (CHM), se utilizan los multiplicadores de esfuerzo (EM, por sus siglas en inglés), los factores de escala (SF, por sus siglas en inglés) y los valores constantes A, B, C y D (ver **Tabla 2.8**). Para obtener el valor correspondiente a los multiplicadores de esfuerzo y los factores de escala se le asigna una escala de muy bajo (mayor valor), bajo, nominal, alto, muy alto y extra alto (menor valor), la cual está asociada a valores de acuerdo con las características que se ajustan al producto a desarrollar y al equipo de trabajo (Ver **Tabla 2.6** y **2.7**).

**Tabla 2.6: Factores de Escala**

| Factor | Descripción | Escala | Valor |
|---|---|---|---|
| **PREC** | Precedencia | Bajo | 4.96 |
| **FLEX** | Flexibilidad | Muy Alto | 1.01 |
| **RESL** | Riesgos | Extra Alto | 0 |
| **TEAM** | Cohesión del Equipo | - | - |
| **PMAT** | Madurez de las Capacidades | Extra Alto | 0 |

| | | | |
|---|---|---|---|
| **Suma** | | | 5.97 |

**Tabla 2.7: Multiplicadores de esfuerzo**

| Multiplicadores | Descripción | Escala | Valor |
|---|---|---|---|
| **RCPX** | Confiabilidad y complejidad del producto | Muy alto | 1.33 |
| **RUSE** | Nivel de reutilizabilidad del desarrollo | Extra alto | 1 |
| **PDIF** | Dificultad de uso de la plataforma | Bajo | 1.29 |
| **PERS** | Capacidad del personal de desarrollo | Alto | 0.83 |
| **PREX** | Experiencia del personal de desarrollo | Muy alto | 0.87 |
| **FCIL** | Facilidades de desarrollo | Alto | 0.87 |
| **SCED** | Exigencias sobre el calendario | Bajo | 1 |
| **Producto** | | | 1.07784906 |

**Tabla 2.8: Constantes**

| Constante | Valor |
|---|---|
| **A** | 2.94 |
| **B** | 0.91 |
| **C** | 3.67 |
| **D** | 0.28 |

El esfuerzo, el tiempo de desarrollo y el costo se calculan a partir de las fórmulas de Boehm, como se muestra en la **Tabla 2.9**.

**Tabla 2.9: Esfuerzo, Tiempo de desarrollo y Costo**

| Cálculo de | Justificación | Valor |
|---|---|---|

| | | |
|---|---|---|
| PM**: Esfuerzo (hombres/mes)** | $PM = A * KSLOC^{E} * \prod_{i=1}^{7} EMi$<br>$E = \mathrm{B} + 0.01 * \sum_{j=1}^{5} \mathrm{SFj} = 1.0376$ | 26,99972 ≈ 27 |
| TDEV**: Tiempo de Desarrollo (meses)** | $F = D + 0.2 * (E - B) = 0.30552$<br>$TDEV = C * PM^{F}$<br>$CH\ Estimado(Cantidad\ de\ Hombres\ Estimados)$<br>$CH = \frac{PM}{TDEV} = 2{,}6997 \approx 3\ \text{hombres}$ | 10.028 ≈ 10 meses |
| **Costo (pesos)** | $CHM\ (Costo\ por\ Hombres/Mes)$<br>$CHM = CH * Salario\ Promedio\ (225)$<br>$CHM * 225 = \$675.00$<br>$COSTO = CHM * TDV$ | $6750,00 |

A pesar del costo estimado para desarrollar el sistema informático propuesto, no se incurre en gastos debido a que el desarrollador es un estudiante. Además, los beneficios que éste brindará al proceso de producción de la Empresa Nacional de Proyectos e Ingeniería ENPA.

Con la implantación del sistema informático disminuirá el tiempo de gestión de los cierres de producción por la facilidad que proporcionará el mismo al realizar algunas tareas automáticamente. Por otra parte, permitirá realizar una búsqueda de información con la calidad requerida por parte del usuario.

Uno de los resultados que genera el corto tiempo empleado anteriormente aludido, es la disminución del gasto de energía eléctrica, en sintonía con el ahorro de energía que lleva a cabo el país.

Las herramientas utilizadas para el desarrollo del sistema en su totalidad son libres, por lo que no se incurre en gastos para desarrollar y aplicar el proyecto.

A partir de lo antes analizado y los beneficios que proporciona el sistema, se arribó a la conclusión que éste es sostenible desde la dimensión administrativa.

**Dimensión Socio-Humanista**

En la dimensión socio-humanista se evalúa cómo el sistema propuesto contribuye con el desarrollo del modo de vida de un grupo social y con la formación ético-humanista de los gestores del proyecto informático, y si satisface una necesidad social. El fortalecimiento del factor humano es necesario para mejorar el rendimiento en los servicios. La automatización de los procesos proporciona el bienestar de los trabajadores en el desempeño laboral a partir de las comodidades brindadas.

Con el uso del sistema, la carga de trabajo se reducirá, al facilitar el intercambio y el flujo de información. Estas mejoras en las condiciones de trabajo darán lugar a una gran satisfacción del personal implicado, lo que favorecerá una mejor calidad en el proceso productivo. Se garantizará que la entrega de los resultados sea correcta, segura y en un tiempo breve.

El sistema podrá ser generalizado, pues el problema que resuelve no es sólo de la Empresa de Proyectos e Ingeniería ENPA de la provincia de Holguín, por lo que podrá ser adaptado para cualquier institución análoga del país. Por tales razones, dentro de las concepciones del sistema se tuvieron en cuenta su flexibilidad y versatilidad, para capturar las generalidades, pero también las posibles particularidades que pueden tener las organizaciones de este ámbito o dominio. La naturaleza modular y extensible de la tecnología con que se desarrolla el sistema hace que no constituya un problema la extensión u evolución del mismo.

Se tuvo en cuenta el rechazo al cambio que podía surgir una vez que se implantara el sistema, lo cual era normal, debido a la tendencia del ser humano a hacer costumbre de lo cotidiano y rutinario y a la resistencia inconsciente ante los cambios de su entorno. Para favorecer la aceptación del sistema se llevaron a cabo entrevistas con los usuarios finales, para explicarles en profundidad las

ventajas que el sistema proporcionaría y cómo el sistema podía serles fácil y cómodo de usar.

El sistema se desarrolló en una interfaz con ambiente *Web*. El flujo de trabajo inmerso en el sistema será similar a como se lleva a cabo los departamentos productivos; es decir, lo más intuitivo posible y fácil de manipular, de tal forma que los usuarios no se pierdan mientras trabajaban sobre él y no lo rechacen ante el cambio.

A partir de lo analizado anteriormente, se arribó a la conclusión de que el sistema es sostenible desde la dimensión socio-humanista.

**Dimensión Ambiental**

En la dimensión ambiental se valora si el sistema resulta favorable o no para las personas o cosas y cómo el mismo minimiza daños e impactos. Cada vez más, el derroche excesivo se incrementa a partir de soluciones no razonables ambientalmente. Por lo que resulta muy importante que toda solución de un problema no repercuta en daños y alteraciones al medio ambiente.

En la ENPA las condiciones de las estaciones de trabajo no son las mejores, debido a la ausencia de protectores de pantallas, asientos cómodos y regulables respecto a la posición que se encuentra el monitor y la ubicación correcta de éste, lo cual puede repercutir en daños en la cervical y la columna, estrés y cansancio en la vista de los usuarios.

Teniendo en cuenta lo antes expuesto y las largas horas que los usuarios se hallan frente a las computadoras durante el flujo de trabajo, no se hará uso de colores agresivos a la vista en la interfaz del sistema, sino de los de tonalidades claras que se encuentran en la gama del verde, gris y blanco, en concordancia con los colores del manual de identidad de la empresa y en busca de un efecto atractivo y agradable en la comunicación entre el sistema y los usuarios.

Por otra parte, se tuvieron en cuenta las exigencias fisiológicas del ser humano al seleccionar la tipografía, el tamaño de letra y el espaciamiento entre caracteres, para la cómoda visualización de los contenidos, alineación y tamaño de las imágenes. Todo esto ayuda a evitar el cansancio visual, los daños en la columna y el estrés de los usuarios al minimizar el tiempo frente al monitor, por lo que se favorece la calidad en las labores productivas.

Se podrán reutilizar los componentes y recursos del sistema, debido a las características de modularidad de las tecnologías empleadas y a que se tuvo en cuenta la generalidad en el diseño del mismo, aunque respetando las particularidades específicas de la ENPA de Holguín.

A partir de lo analizado anteriormente se arribó a la conclusión de que el sistema es sostenible desde la dimensión ambiental.

**Dimensión Tecnológica**

En la dimensión tecnológica de evalúa si la tecnología usada es adecuada y asimilable con el usuario. Para el empleo del sistema los usuarios se encuentran capacitados y no necesitan de preparación informática, debido a que hacen uso de computadoras para su labor diaria; además el sistema será intuitivo, fácil de usar y poseerá una Ayuda que facilitará su manipulación.

En cuanto a la infraestructura electrónica, la ENPA cuenta con los recursos precisos para la implantación y aplicación del sistema. Las características que poseen las computadoras utilizadas por los trabajadores cumplen con los requerimientos necesarios para hacer uso del sistema y además se encuentran conectadas a la red.

Con el fin de facilitar el mantenimiento del sistema, se usó un estándar de código, se describieron con comentarios las funciones fundamentales y de forma general lo que hacen las clases, además, se le pusieron nombres intuitivos para proporcionar una mejor comprensión y entendimiento.

Por otra parte, el sistema permitirá su evolución en el tiempo, debido a la flexibilidad que proporcionará (explicado anteriormente en la dimensión socio-

humanista). Además, permitirá cambios, ya sea de mejoras de hardware, red e incluso de plataforma.

A partir de lo analizado anteriormente se arribó a la conclusión de que el sistema es sostenible desde la dimensión tecnológica.

## 2.8 Valoración de la Propuesta de Solución

Para cumplir con las características que plantean las preguntas científicas es necesario validar cada una de las funcionalidades del sistema, para su procesamiento estadístico fue empleado el Criterio de Expertos que utiliza el método Delphi para el procesamiento de las encuestas, el cual posibilitó la emisión de un pronóstico que ratificó la validez de la solución.

El método Delphi es considerado como uno de los métodos subjetivos de pronósticos más confiables, constituye un procedimiento para confeccionar un cuadro de la evolución estadística de las opiniones de expertos o usuarios en un tema tratado. Este método emplea una serie de variables que, a través de cálculos, permiten obtener un resultado final en cuanto a la relevancia de la solución (García 2008).

Como se mencionó el método está basado en la realización de encuestas para conocer el grado de competencia de los posibles expertos (Ver Anexo 8). Para ello se consultaron profesionales con experiencia, creatividad, capacidad de análisis y de pensamiento en el tema tratado. Luego de seleccionar 20 expertos de los 25 usuarios encuestados se les aplicó una encuesta para saber el nivel de satisfacción con respecto a la aplicación, para ello se tuvo presente varios aspectos importantes (Ver Anexo 9).

Después de realizar el procesamiento estadístico de los aspectos antes mencionados y analizados en la encuesta, se comprobó que los resultados finales obtenidos son satisfactorios, contribuyendo positivamente en la calidad del sistema, lo cual posibilitó percibir la satisfacción de las necesidades iniciales.

El Anexo 10 muestra las diferentes tablas obtenidas con el procesamiento de las encuestas y las conclusiones generales del mismo.

Al valorar de forma general los resultados obtenidos para los diferentes criterios analizados, se puede concluir que es factible la implantación del sistema por obtener la categoría de muy adecuado con un 100% en los aspectos encuestados.

## 2.9 Conclusiones del capítulo

En este capítulo se determinaron, a través de la metodología ICONIX, los objetos o entidades fundamentales implicadas en el dominio del problema y la relación existente entre ellas. Se identificaron las exigencias que debe cumplir la solución propuesta, así como los usuarios finales y las acciones que estos deben ejecutar.

Además, se especificaron los cursos básicos y alternos de las acciones del sistema, así como los procesos de diseño, implementación y prueba de la metodología utilizada. Se definió, además, un estándar de código para la implementación.

Se realizó un estudio de la sostenibilidad administrativa, socio-humanista, ambiental y tecnológica del sistema, por lo que se arribó a la conclusión de que su desarrollo es factible, sostenible y perdurable en el tiempo.

Luego de analizar los resultados obtenidos del procesamiento de las encuestas aplicadas a expertos, el sistema alcanzó un nivel muy adecuado de aceptación por parte de estos, lo cual garantiza un alto grado de impacto en su implantación en la ENPA.

# Conclusiones generales

Con el desarrollo del presente trabajo de diploma se obtuvo un sistema informático que favorece la gestión de información de la valoración económica de proyectos en la ENPA, que permite contar con un historial de cada valoración económica de proyectos, generar reportes y se caracteriza por la confiabilidad de los datos, seguridad, usabilidad y rendimiento; dando cumplimiento al objetivo de la investigación.

Las tecnologías, herramientas y metodología empleadas para el desarrollo del sistema fueron acertadas y adecuadas, favoreciendo en gran medida el progreso del mismo, cubriendo completamente las necesidades de los clientes y desarrollador.

Una vez realizada la valoración de sostenibilidad del sistema informático según las dimensiones administrativa, socio-humanista, ambiental y tecnológica, se obtuvo que este es sostenible y perdurable en el tiempo.

Luego de analizar los resultados obtenidos del procesamiento de las encuestas aplicadas a expertos, el sistema alcanzó un nivel muy adecuado de aceptación por parte de estos, lo cual garantiza un alto grado de impacto en su implantación en la ENPA.

# Recomendaciones

Para continuar el desarrollo de esta investigación se recomienda:

- Generar otros reportes con el objetivo de brindar más información que pueda resultar de interés y favorezca la toma de decisiones.
- Extender el uso de la aplicación a otras ENPA en el país.

# Referencias bibliográficas

Acuña, K. B. (2009). "Selección de metodologías de desarrollo para aplicaciones web en la facultad de informática de la universidad de Cienfuegos."

Álvarez, G. (2010). "Características del lenguaje Java." from http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html#top.

Andalucía, J. d. (2012). "Patrón Modelo Vista Controlador ", from http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122.

Ayala, A. P. (2009). Ingeniería de software: Guía para crear sistemas de información.

Bates, K. S. a. B. (2005). Head First Java.

Boehm, B. (1981). Software Engineering Economics.

Brett D. McLaughlin, G. P. a. D. W. (2006). Head First Object Oriented Analysis and Design.

Burbano, D. J. (2009) "Metodologias de desarrollo de Software. ."

Bustos, G. (2010). Guía de Uso de la Herramienta CASE Visual Paradigm Standard.

Calzadilla, A. H. (2012). Manual Integrado de Gestión.

Canos, J. H. (2008). Métodologías Ágiles en el Desarrollo de Software, Universidad Politécnica de Valencia.

Charvat, J. (2003). Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects. J. W. Sons.

Chen, H. y. C., R (2010). ZK Component Development Essentials. P.

Corporation. s.1. .

Clientes, L. (2009). "Lenguajes del lado servidor o cliente." from http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

Cocomo (2000). COCOMO II Model Definition Manual

Concepción, R. (2006) "Procedimiento para la valoración de sostenibilidad de un Producto Informático."

## BIBLIOGRAFIA


Deltaasesores. (2010). "Impacto de Tecnologías Informáticas." from http://www.deltaasesores.com.

Díaz, Z. M. (2011). Sistema para la gestión del plan bibliográfico en los departamentos docentes de la Universidad “Oscar Lucero Moya” de Holguín, Universidad de Holguín.

Doug rosenberg, M. S. A. M. C.-C. (2005). Agile Development with ICONIX Process—People, Process, and Pragmatism J. Sumser.

Eckel, B. (2006). Thinking in Java.

EvalAs. (2012). "EvalAs Software para Evaluación de Proyectos de Inversión Productivos." from http://www.elsitioagricola.com/Eventos/Software_Evaluación_de_Proyectos.htm.

Fernandez, F. G. B. R. M. U. a. V. F. (2012). Lenguajes de programacion y procesadores.

García, F. P. (2013). "Aplicaciones Web." from http://www.amazon.es/Aplicaciones-Web-Sistemas-Microinformaticos-Redes/dp/1492217271.

García, I. P. (2009). " Instalación y Administración de Apache Tomcat 6."

Garcías, L. (2010). "Taller instalación de servidores web Apache Tomcat .".

Gavin King, C. B., Max Rydahl Andersen, Emmanuel Bernard and Steve Ebersole. (2010). "HIBERNATE - Persistencia relacional para Java idiomático." from https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html_single/.

GILFILLAN, I. (2003). La biblia de mysql. A. Multimedia.

González, P. C. ( 2009). Metodologia de desarrollo ICONIX.

González, Y. F. R. a. Y. D. (2012). Patrón Modelo-Vista-Controlador. Revista Telem@tica.

Gracia, L. M. (2010). "Framework MVC ZK." from http://unpocodejava.wordpress.com/2010/01/27/framework-mvc-zk/.

GuíaProgramador. (2012). "Aplicaciones Web." from www.guiaprogramdor.com.

Gutiérrez., J. J. (2009). Que es un Framework.

Helmle, P. E. a. B. (2010). PostgreSQL-Administration.

Herrera, C. (2013) "Informes en Java con iReports."

## BIBLIOGRAFIA

Informático, M. (2010). "Lenguajes de programacion." from http://jorgesaavedra.wordpress.com/2010/05/05/lenguajes-de-programacion/.

Jacobson, I. (2000). El Proceso Unificado de Desarrollo de Software.

James Gosling, B. J., Guy L. Steele Jr. and Gilad Bracha (2013). The Java Language Specification.

JDBC. (2010). "Bases de datos: Puente JDBC (Java Database Connectivity) ODBC (Open Database Connectivity)." from http://html.rincondelvago.com/bases-de-datos_puente-jdbc-odbc.html.

Jeri, R. O. D. H. a. E. (1998). Cliente/Servidor: Guía de supervivencia. México, D.F. , McGraw-Hill Interamericana

Johnson, R. (2009). "Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks."

Kabir, M. J. (2003). La biblia del servidor apache.

Larman, C. (2003). Agile and Iterative Development: A Manager's Guide, Addison Wesley.

Lopez, d. R. M. P. V. A. A. O. P. (2010). Aplicaciones web. Un enfoque practico. R.-m. Editorial.

Macedo, R. D. P. (2013) "JDBC: Java Database Connectivity."

Mcgraw-hill (2011) "Sistemas gestores de base de Datos."

Monografías. (2010). "La Informática y su Impacto Social." Retrieved 04/01/13, from http://monografias.com.

Moseley, R. (2007). Desarrollo de aplicaciones Web.

Obando, P. A. (2012). Manual Curso Básico de PostgreSQL. Autoedición.

Oliva, C. R. (2009). Uso de ICONIX.

Páez, F. J. M. (2010). "Introducción a JDBC." from http://www.adictosaltrabajo.com/tutoriales/

Peña, L. L. B. a. J. R. d. l. (2010). Aplicación de la metodología Iconix en el proceso de desarrollo de software, Universidad de Holguín Oscar Lucero Moya.

Yisel, Clavel Quintero. (2010). Sistema de clasificación automática de noticias a publicar en el periódico ¡AHORA! DIGITAL

## BIBLIOGRAFIA


Pérez, F. F. (2012). "Framework ZK." from http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=zk_serie_1-5.

Pérez, t. G. (2010). Sistemas gestores de bases de datos. Innovación y Experiencias Educativas. 30.

Quintanilla, G. A. G. (2012) "Servidores Web."

Rey, E. G. (2013) "Lenguajes de programacion."

Rodríguez, J. R. M. A. R. M. a. F. M. (2006). Sistemas gestores debases de datos. S. A. U. McGraw-Hill/Interamericana de España.

Rugarcía, B. M. M. A. a. J. M. Z. (2013). Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español Universidad de las Américas Puebla.

RUP. (2009). "El Proceso Unificado de Desarrollo de Software (RUP)." from http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.

Sánchez, M. A. M. (2004). "Metodologia de desarrollo de Software."

Sharon Biocca Zakhour, S. K. a. R. G. (2013). The Java Tutorial: A Short Course on the Basics.

Sistemas1, C. (2012). "Sistema_Evaluacion_Economica."

Sistemas, C. (2012). "Sistema_Control_Costos_Visual." from http://saotechnology.com.

Stephens, D. R. a. M. (2007). Use Case Driven Object Modeling with UML Theory and Practice.

Tareas, B. (2013). "Arquitectura cliente-servidor, servicios de internet y protocolos." from http://www.buenastareas.com/ensayos/Arquitectura-Cliente-Servidor-Servicios-De-Internet-y/31983559.html.

Tutoriales, J. (2009). "HIbernate - Parte 1: Persistiendo Objetos Simples usando Mapeos en XML ", from http://www.javatutoriales.com/2009/05/hibernate-parte-1-persistiendo-objetos.html.

Vegas, J. (2009) "Introducción a las Aplicaciones Web."

Yarlequé, W. C. (2009). Lenguajes de Programación para Aplicaciones Web.

# Anexos

## Anexo 1: Descripciones Textuales de los CUS

| Caso de Uso: Insertar Cargo | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Cargos donde escoge la opción Insertar cargo y el sistema muestra la ventana de Crear nuevo cargo con datos a llenar (nombre, salario básico), al ser llenados da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos y que el nombre del cargo no se encuentre ya creado, muestra un mensaje de confirmación para insertar el cargo, el Administrador hace clic en el botón aceptar del mensaje de confirmación, almacena los datos y muestra un mensaje de que el Cargo ha sido insertado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Ya existe el cargo: El sistema muestra un mensaje de que el cargo ya existe.<br>Selecciona cancelar en el mensaje de confirmación: el sistema cierra el mensaje y vuelve a la interfaz anterior. |

| Caso de Uso: Actualizar Cargo | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Cargos donde escoge la opción Actualizar cargo y el sistema muestra la ventana de Actualizar cargo. El **Administrador** escoge el cargo que desea actualizar, realiza el cambio en los datos que desee y da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos, muestra un mensaje de confirmación, el usuario selecciona aceptar, almacena los datos y muestra un mensaje de que el Cargo ha sido modificado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Selecciona cancelar en el mensaje de confirmación: el sistema cierra |

| | |
|---|---|
| | el mensaje y vuelve a la interfaz anterior. |

| **Caso de Uso: Eliminar Cargo** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Cargos donde escoge la opción Eliminar cargo y el sistema muestra la ventana de Eliminar cargo. El **Administrador** escoge el cargo que desea eliminar y da clic en el botón Aceptar, el sistema muestra un mensaje de confirmación, el **Administrador** da clic en el botón aceptar y el sistema muestra un mensaje de que el Cargo ha sido eliminado satisfactoriamente. |
| **Curso alterno** | No se seleccionó ningún cargo: el sistema muestra un mensaje donde dice que debe seleccionar el cargo a eliminar.<br>El **Administrador** da clic en Cancelar y no se realiza la acción. |

| **Caso de Uso: Insertar Insumo** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Insumos donde escoge la opción Insertar insumo y el sistema muestra la interfaz de Crear nuevo insumo con datos a llenar, al ser llenados da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos y que el nombre del insumo no se encuentre ya creado, muestra un mensaje de confirmación, el administrador selecciona aceptar, almacena los datos y muestra un mensaje de que el insumo ha sido insertado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Ya existe el insumo: El sistema muestra un mensaje de que el insumo ya existe. |

| **Caso de Uso: Actualizar insumo** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Insumos donde escoge la opción Actualizar insumo y el sistema muestra la ventana de Actualizar insumo. El **Administrador** escoge el insumo que desea actualizar, realiza el cambio en los datos que desee y da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos, muestra un mensaje de confirmación, el Administrador selecciona aceptar, almacena los datos y muestra un mensaje de que el Insumo ha sido modificado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Selecciona cancelar en el mensaje de confirmación: el sistema cierra el mensaje y vuelve a la interfaz anterior. |

| **Caso de Uso: Eliminar Insumo** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Insumos donde escoge la opción Eliminar insumo y el sistema muestra la ventana de Eliminar insumo. El **Administrador** escoge el insumo que desea eliminar y da clic en el botón Aceptar, el sistema muestra un mensaje de confirmación, el Administrador da clic en el botón aceptar y el sistema muestra un mensaje de que el Insumo ha sido eliminado satisfactoriamente. |
| **Curso alterno** | No se seleccionó ningún insumo: el sistema muestra un mensaje donde dice que debe seleccionar el Insumo a eliminar.<br>El **Administrador** da clic en Cancelar y no se realiza la acción. |

| **Caso de Uso: Insertar Proyectista** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Proyectistas donde escoge la opción Insertar proyectista y el sistema muestra la interfaz de Crear nuevo proyectista con datos a llenar, al ser llenados da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos y que el nombre del proyectista no se encuentre |

| | |
|---|---|
| | ya creado, muestra un mensaje de confirmación, el **Administrador** selecciona aceptar, almacena los datos y muestra un mensaje de que el proyectista ha sido insertado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Ya existe el proyectista: El sistema muestra un mensaje de que el proyectista ya existe. |

| **Caso de Uso: Actualizar Proyectista** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Proyectistas donde escoge la opción Actualizar proyectista y el sistema muestra la ventana de Actualizar proyectista. El **Administrador** escoge el proyectista que desea actualizar, realiza el cambio en los datos que desee y da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos, muestra un mensaje de confirmación, el Administrador selecciona aceptar, almacena los datos y muestra un mensaje de que el proyectista ha sido modificado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Selecciona cancelar en el mensaje de confirmación: el sistema cierra el mensaje y vuelve a la interfaz anterior. |

| **Caso de Uso: Eliminar Proyectista** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Proyectistas donde escoge la opción Eliminar proyectista y el sistema muestra la ventana de Eliminar proyectista. El **Administrador** escoge el proyectista que desea eliminar y da clic en el botón Aceptar, el sistema muestra un mensaje de confirmación, el Administrador da clic en el botón aceptar y el sistema muestra un mensaje de que el proyectista ha sido eliminado satisfactoriamente. |
| **Curso alterno** | No se seleccionó ningún proyectista: el sistema muestra un mensaje |

| | |
|---|---|
| | donde dice que debe seleccionar el Insumo a eliminar.<br>El **Administrador** da clic en Cancelar y no se realiza la acción. |

| **Caso de Uso: Insertar Proyecto** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Proyecto donde escoge la opción Insertar proyecto y el sistema muestra la interfaz de Crear nuevo proyecto con datos a llenar, al ser llenados da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos y que el código del proyecto no se encuentre ya creado, muestra un mensaje de confirmación, el **Administrador** selecciona aceptar, almacena los datos y muestra un mensaje de que el proyecto ha sido insertado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Ya existe el proyecto: El sistema muestra un mensaje de que el proyecto ya existe. |

| **Caso de Uso: Actualizar Proyecto** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú Proyectos donde escoge la opción Actualizar proyecto y el sistema muestra la ventana de Actualizar proyecto. El **Administrador** escoge el proyecto que desea actualizar, realiza el cambio en los datos que desee y da clic en el botón Aceptar. El sistema verifica que no se dejen campos vacíos, muestra un mensaje de confirmación, el Administrador selecciona aceptar, almacena los datos y muestra un mensaje de que el proyecto ha sido modificado satisfactoriamente. |
| **Curso alterno** | Datos vacíos: El sistema muestra un mensaje de que se deben llenar todos los datos.<br>Selecciona cancelar en el mensaje de confirmación: el sistema cierra el mensaje y vuelve a la interfaz anterior. |

| **Caso de Uso: Eliminar Proyecto** | |
|---|---|
| **Curso básico** | El **Administrador** va al menú Inicio y dentro de él va al submenú |

| | |
|---|---|
| | Proyectos donde escoge la opción Eliminar proyecto y el sistema muestra la ventana de Eliminar proyecto. El **Administrador** escoge el proyecto que desea eliminar y da clic en el botón Aceptar, el sistema muestra un mensaje de confirmación, el Administrador da clic en el botón aceptar y el sistema muestra un mensaje de que el proyecto ha sido eliminado satisfactoriamente. |
| **Curso alterno** | No se seleccionó ningún proyecto: el sistema muestra un mensaje donde dice que debe seleccionar el Insumo a eliminar.<br>El **Administrador** da clic en Cancelar y no se realiza la acción. |

## Anexo 2: Diagramas de Robustez de los CUS

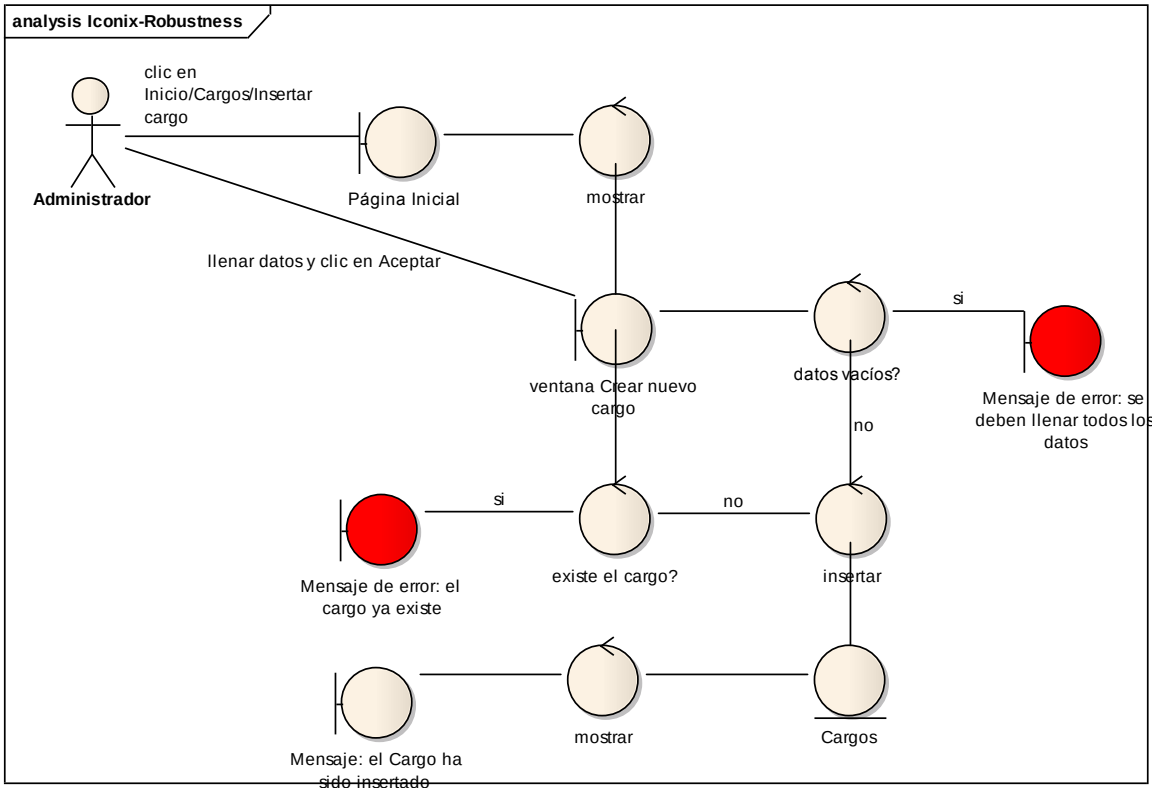



**Diagrama de Robustez del CUS: Insertar cargo**

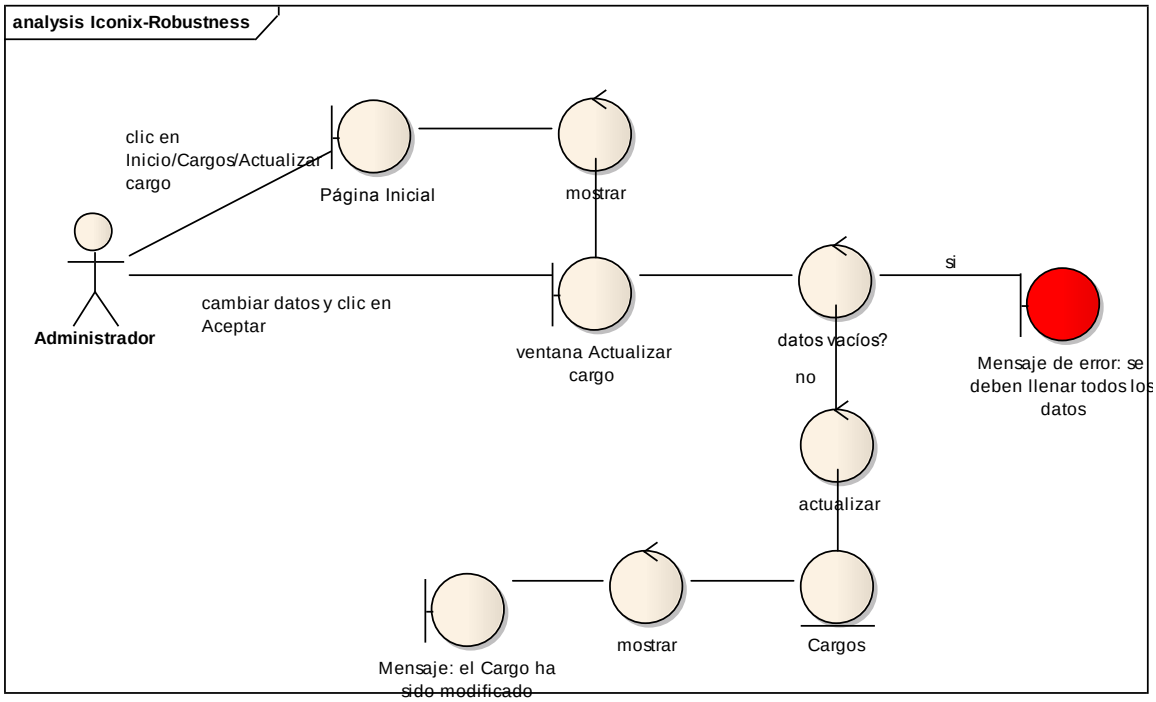



**Diagrama de Robustez del CUS: Actualizar cargo**

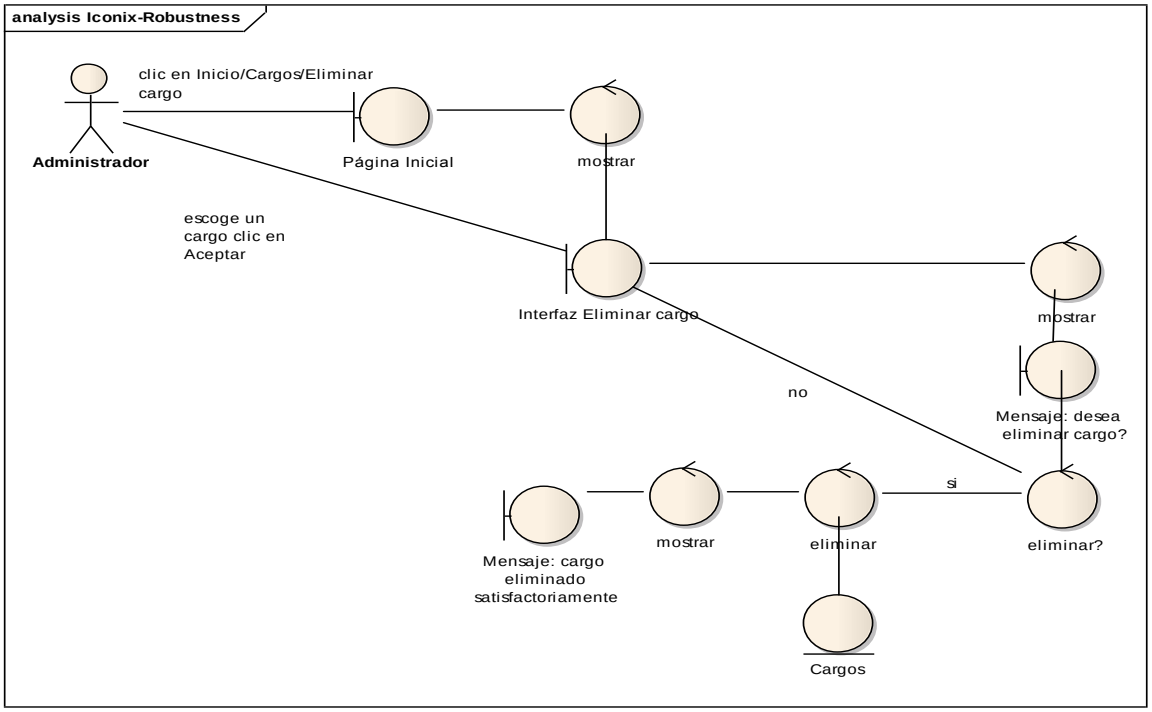



**Diagrama de Robustez del CUS: Eliminar cargo**

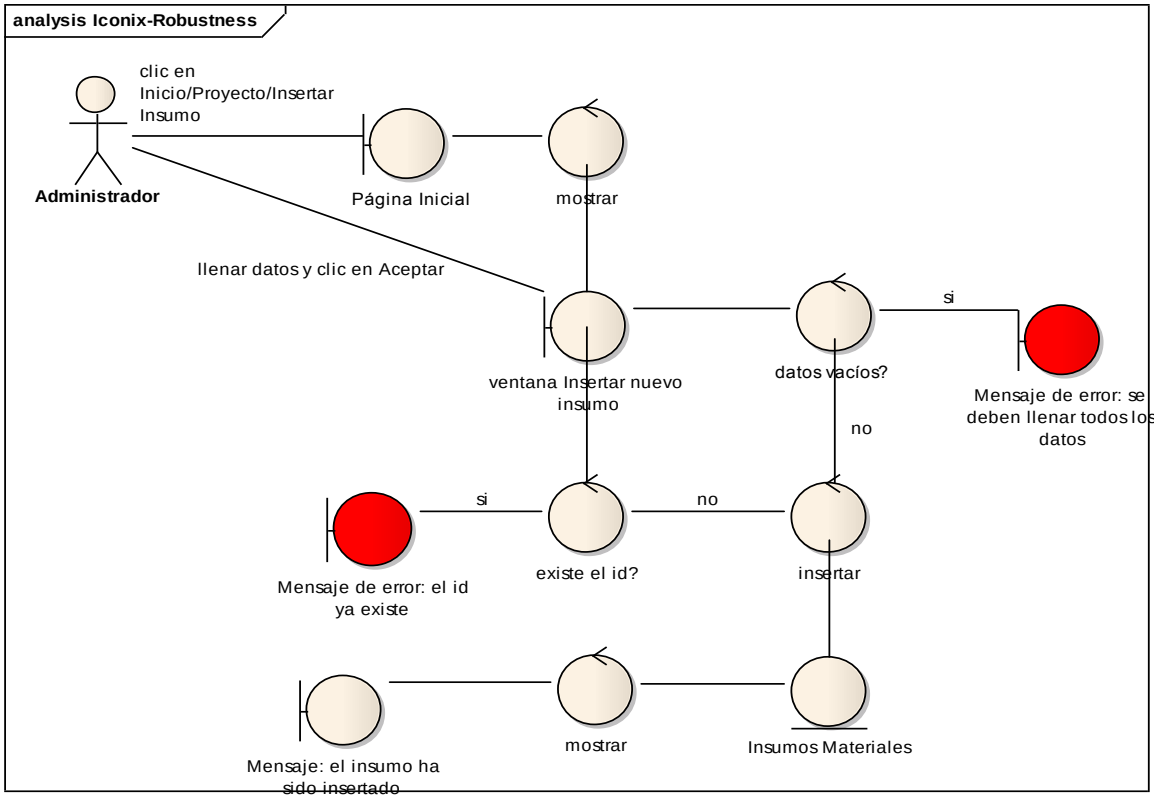



**Diagrama de Robustez del CUS: Insertar insumo**

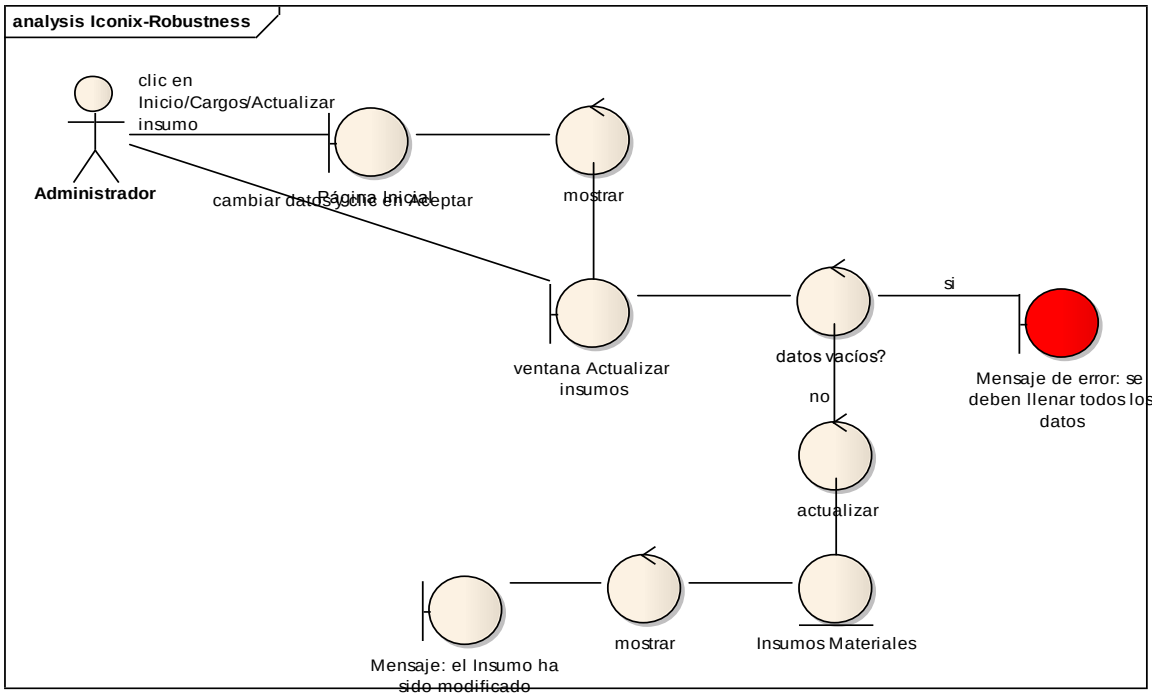



**Diagrama de Robustez del CUS: Actualizar insumo**

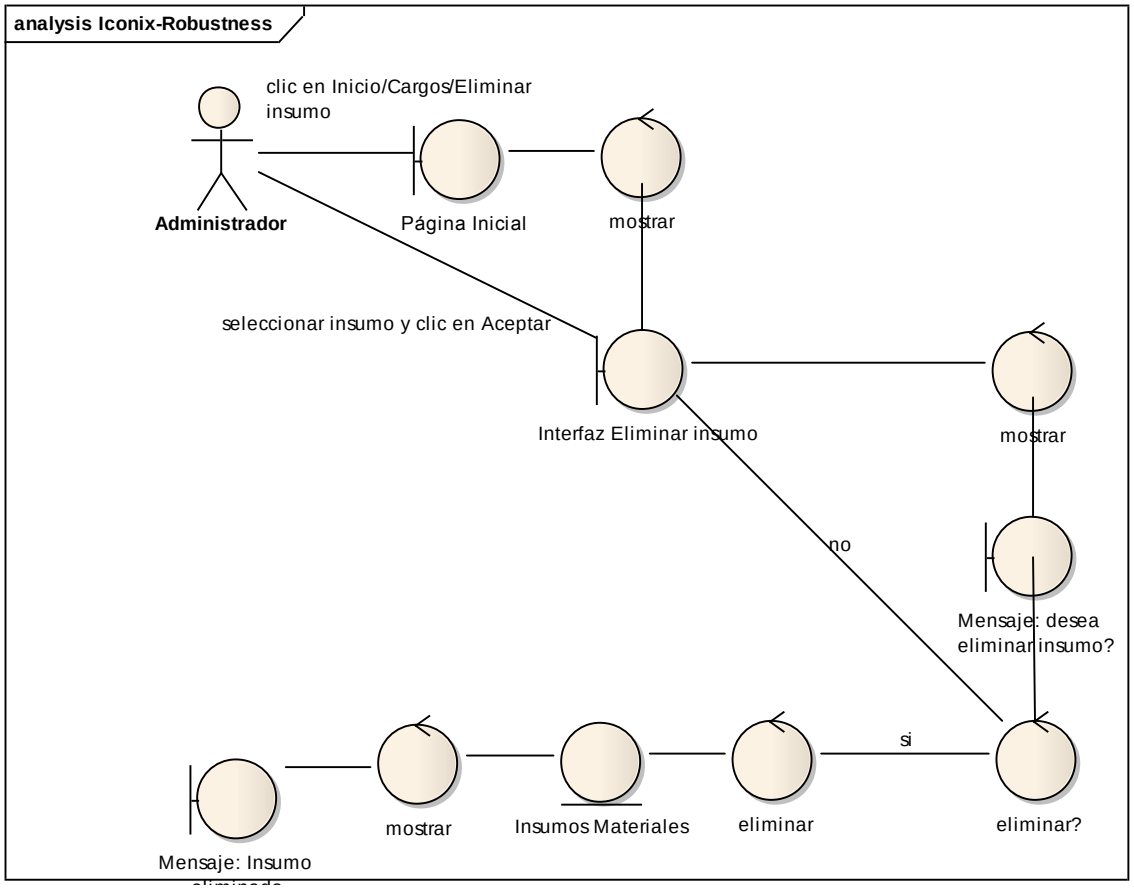



**Diagrama de Robustez del CUS: Eliminar insumo**

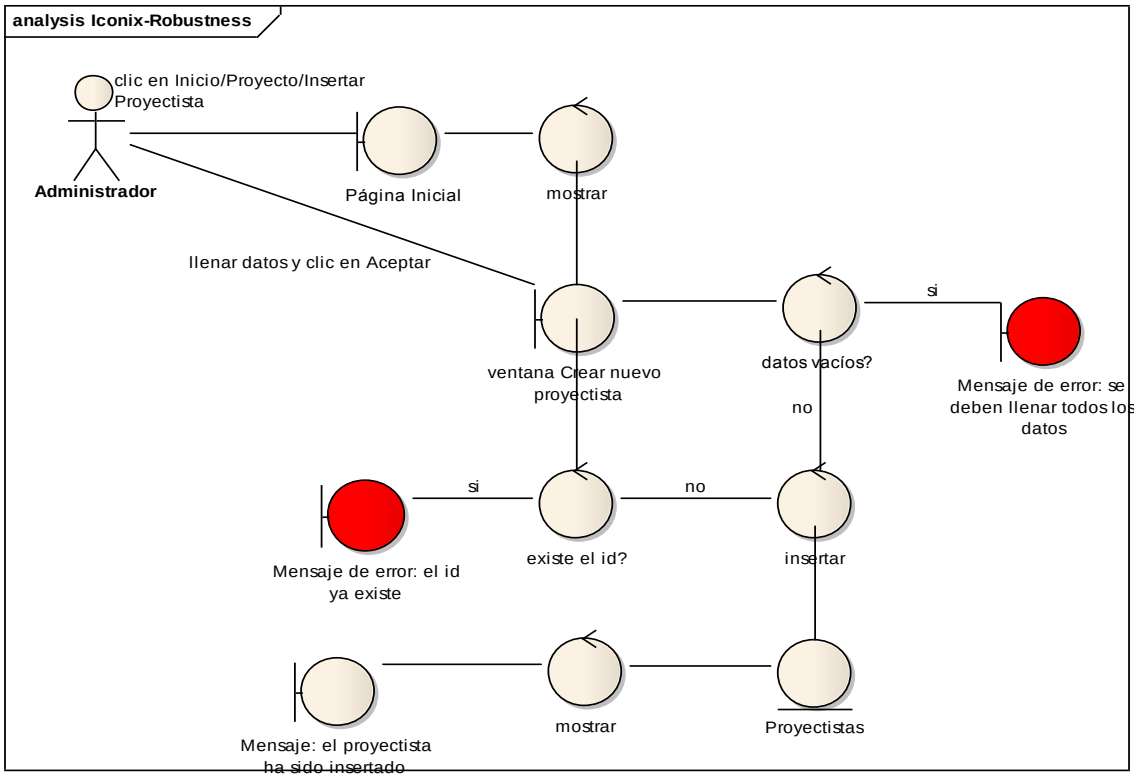



**Diagrama de Robustez del CUS: Insertar Proyectista**

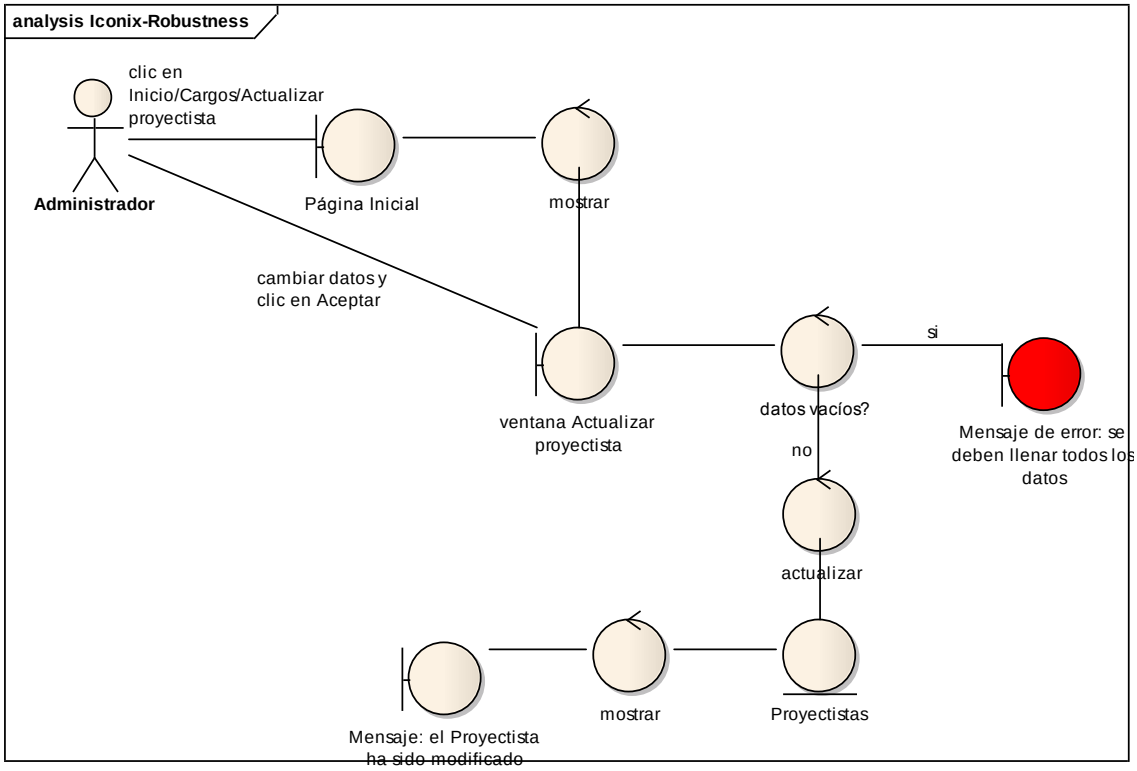



**Diagrama de Robustez del CUS: Actualizar proyectista**

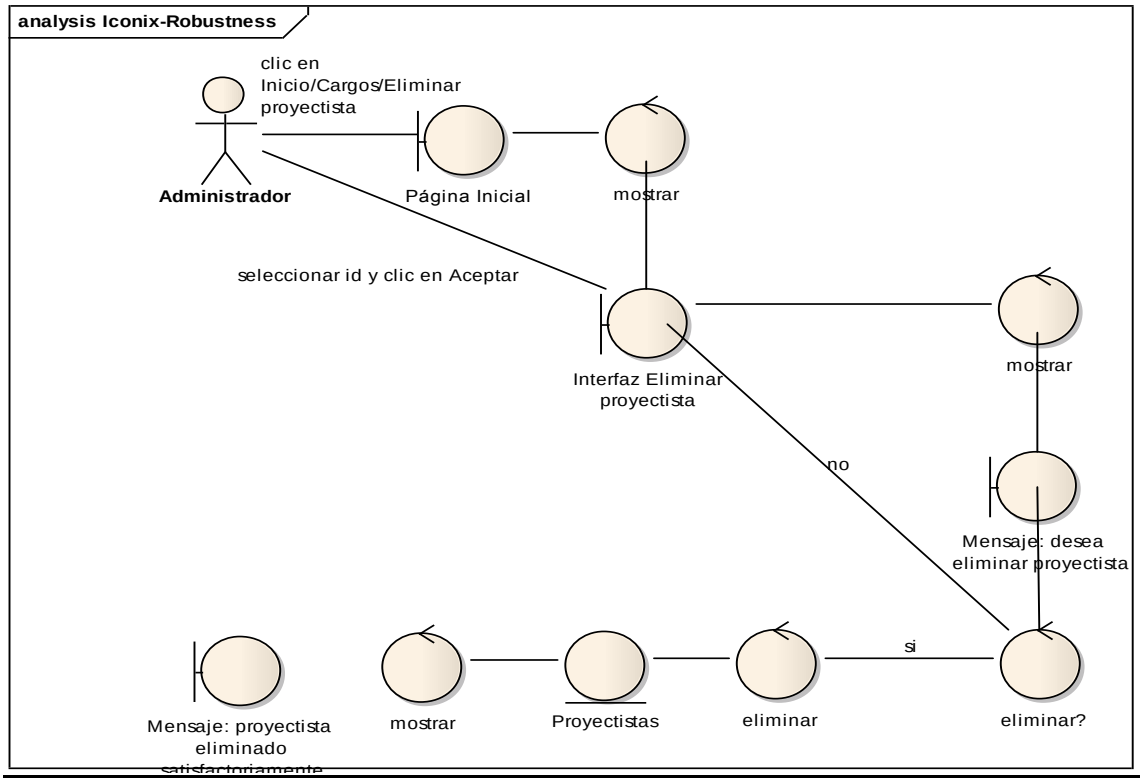



**Diagrama de Robustez del CUS: Eliminar proyectista**

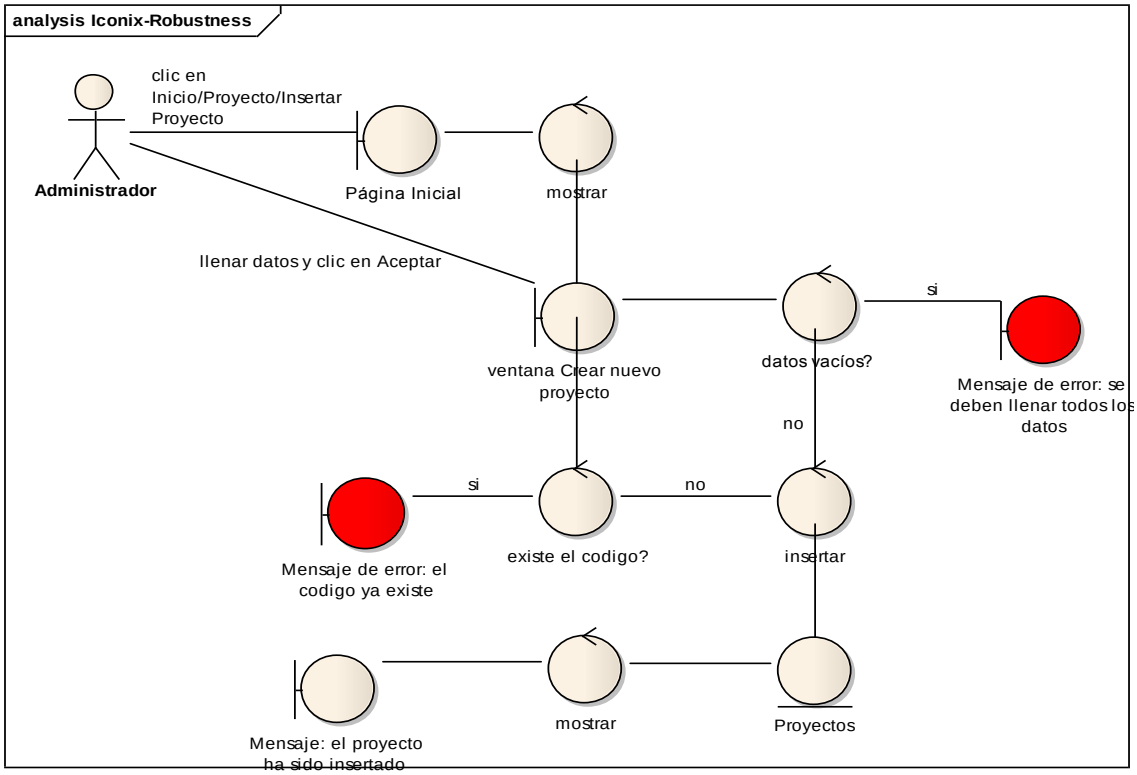



**Diagrama de Robustez del CUS: Insertar proyecto**

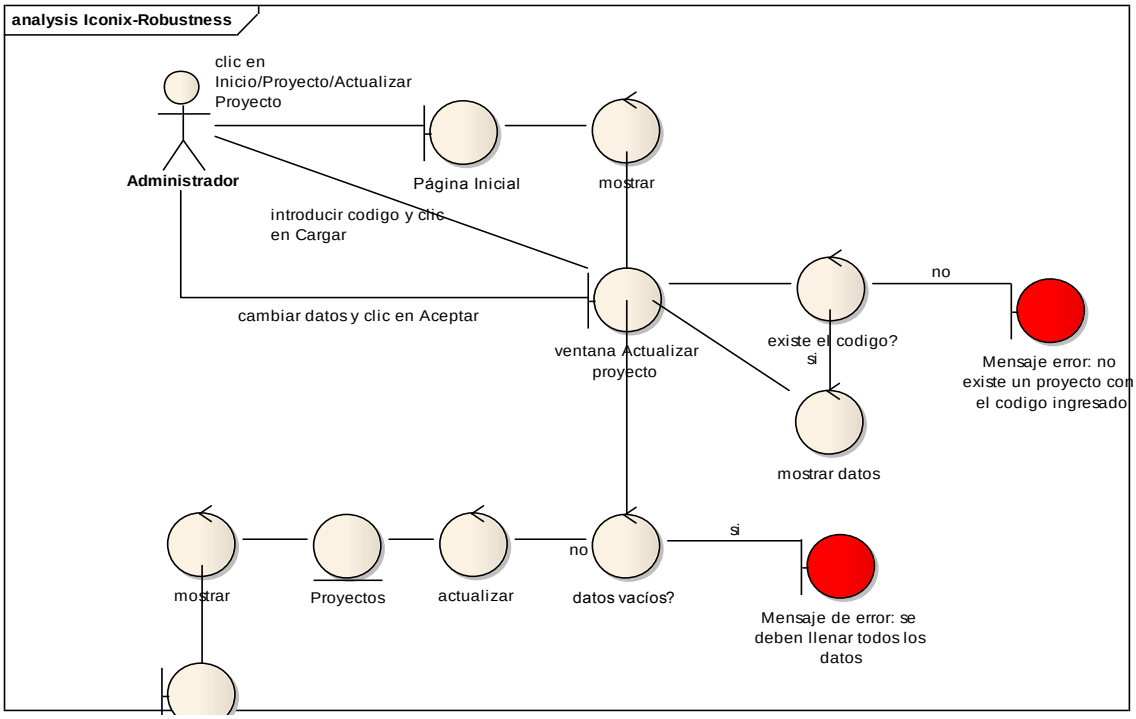



**Diagrama de Robustez del CUS: Actualizar proyecto**

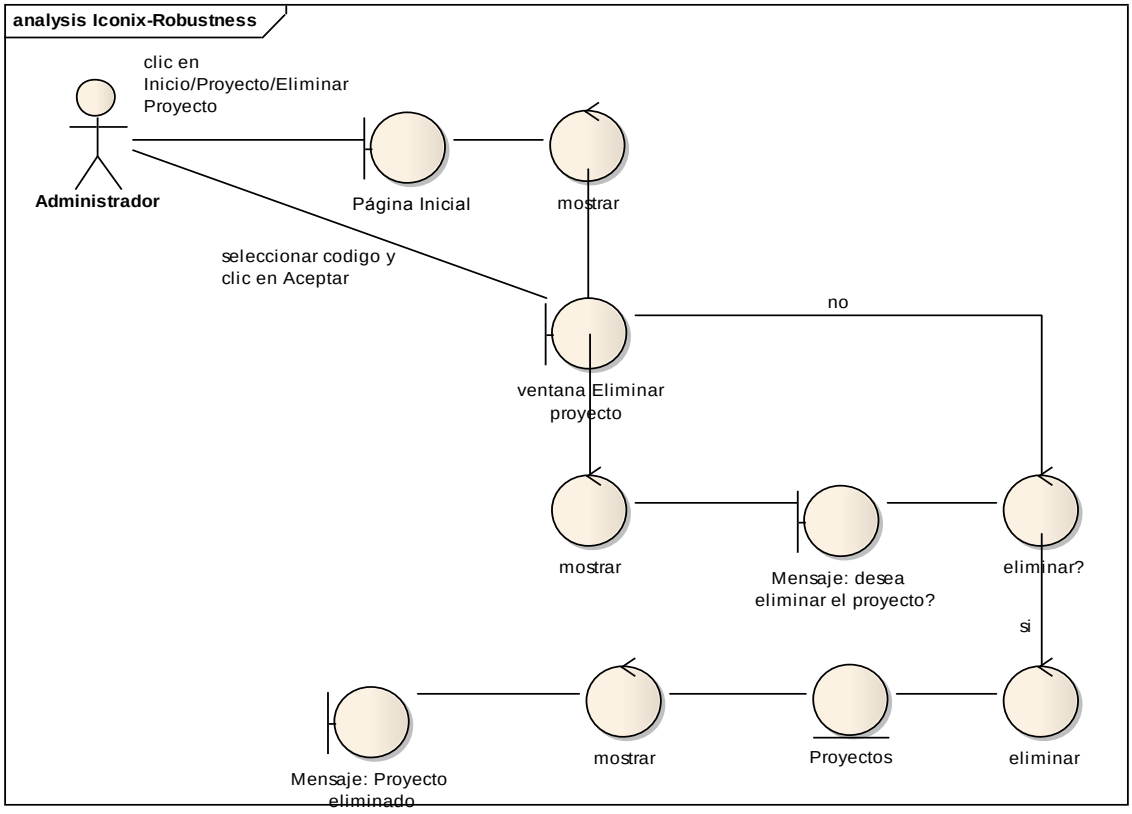



**Diagrama de Robustez del CUS: Eliminar proyecto**

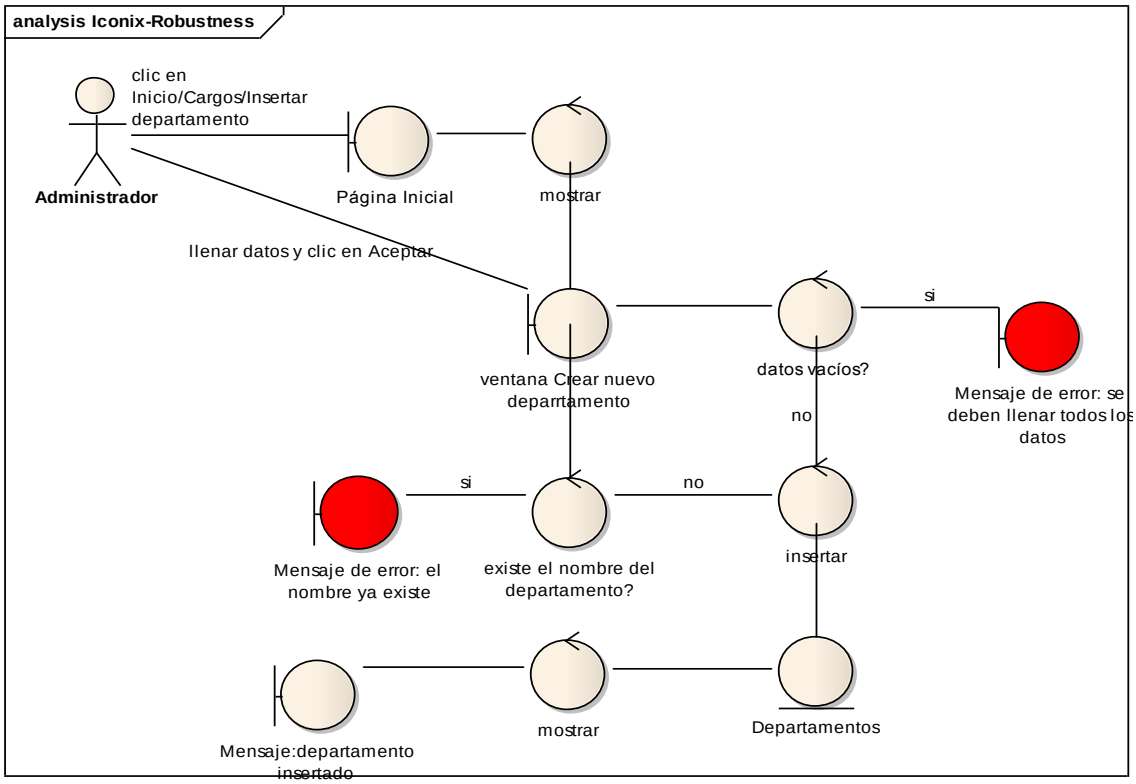



**Diagrama de Robustez del CUS: Insertar departamento**

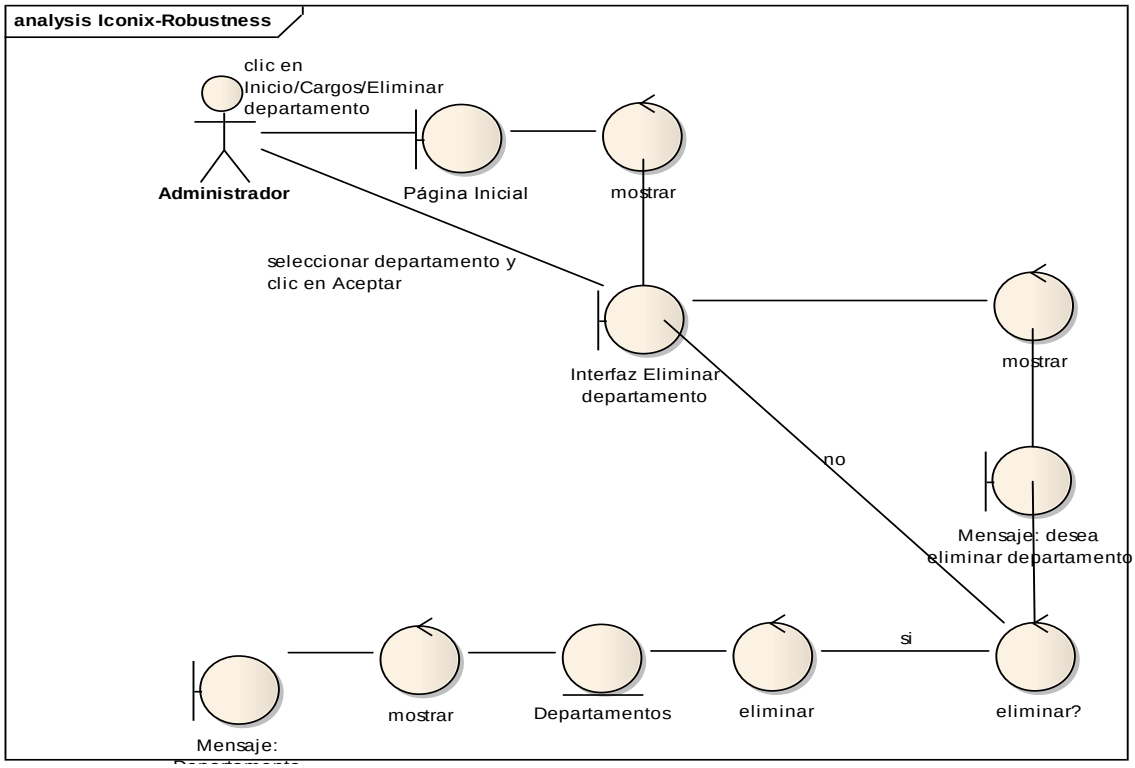



**Diagrama de Robustez del CUS: Eliminar departamento**

## Anexo 3: Diagramas de Secuencia de los CUS

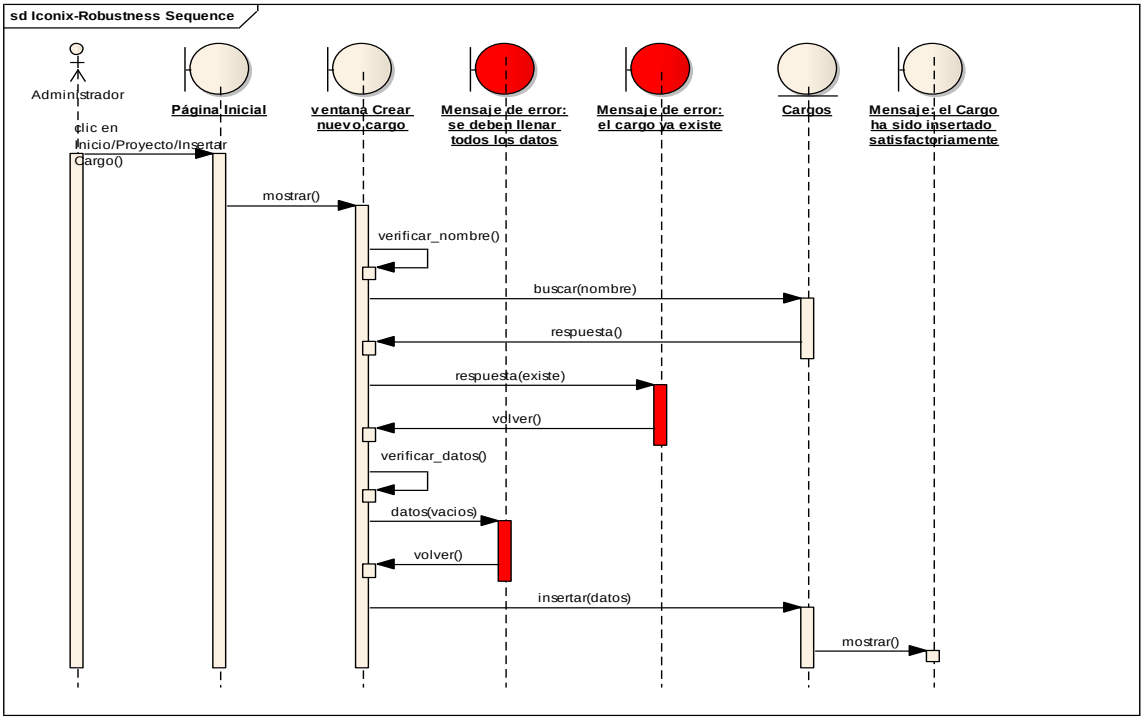



**Diagrama de Secuencia del CUS: Insertar cargo**

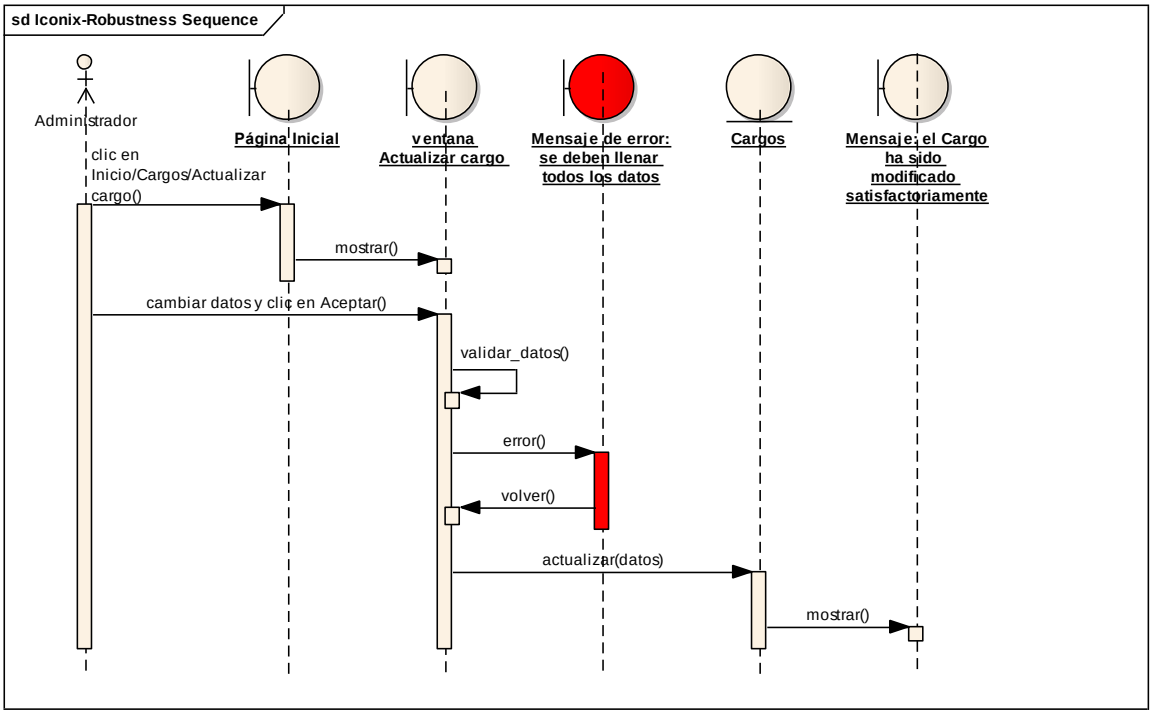



**Diagrama de Secuencia del CUS: Actualizar cargo**

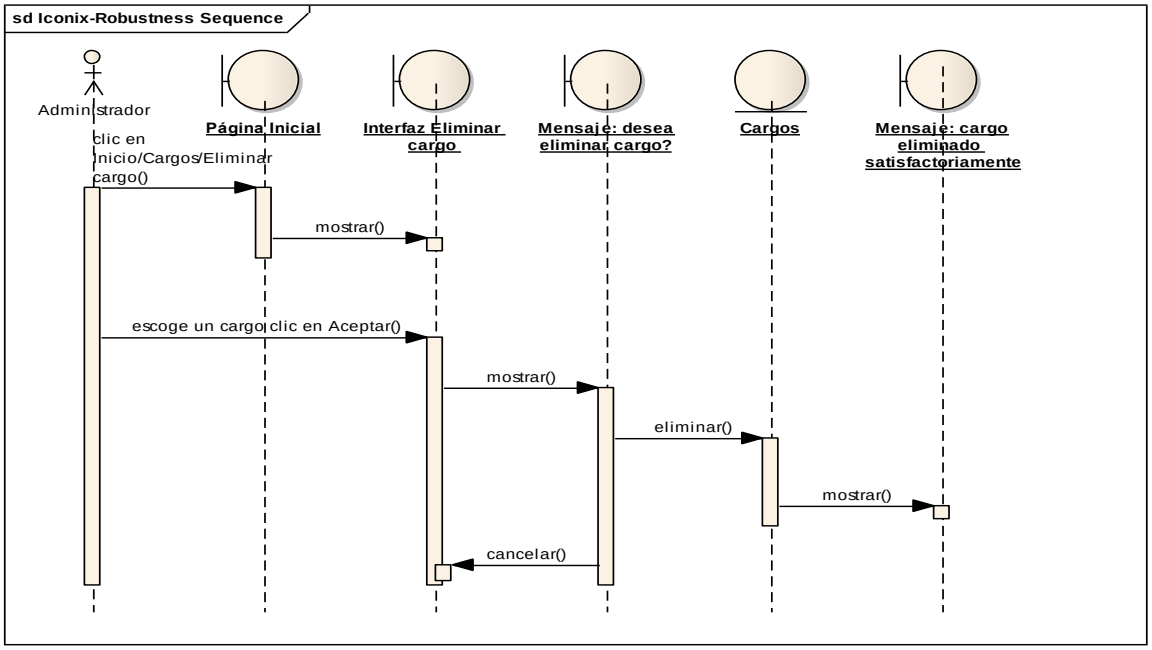



**Diagrama de Secuencia del CUS: Eliminar cargo**

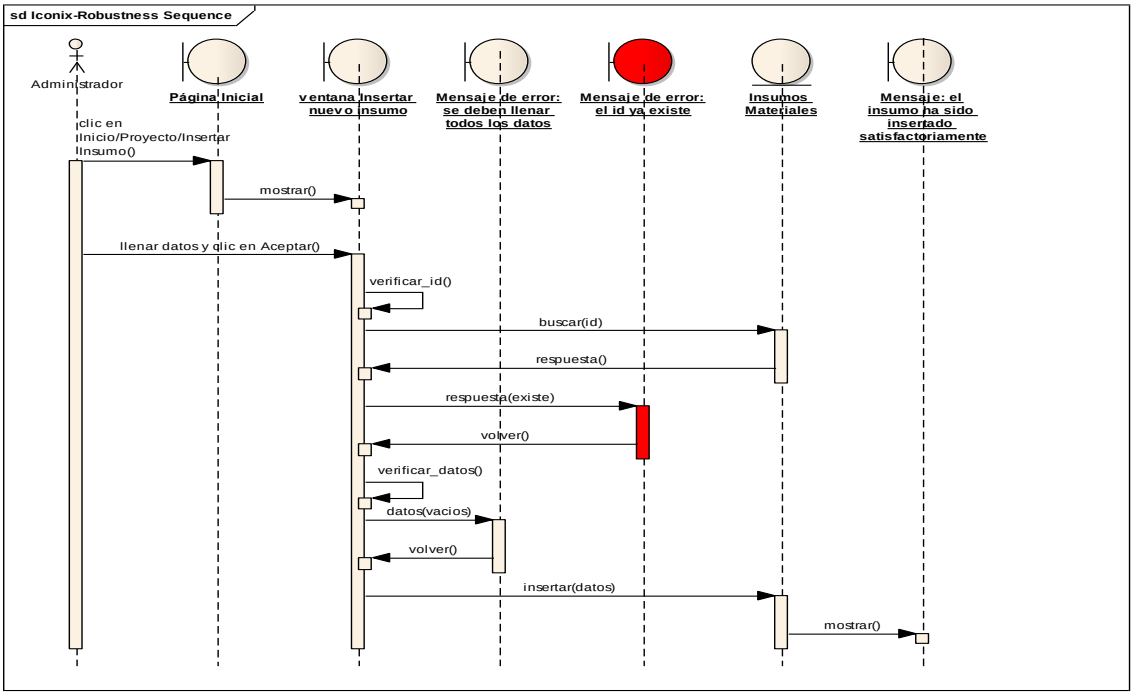



**Diagrama de Secuencia del CUS: Insertar insumo**

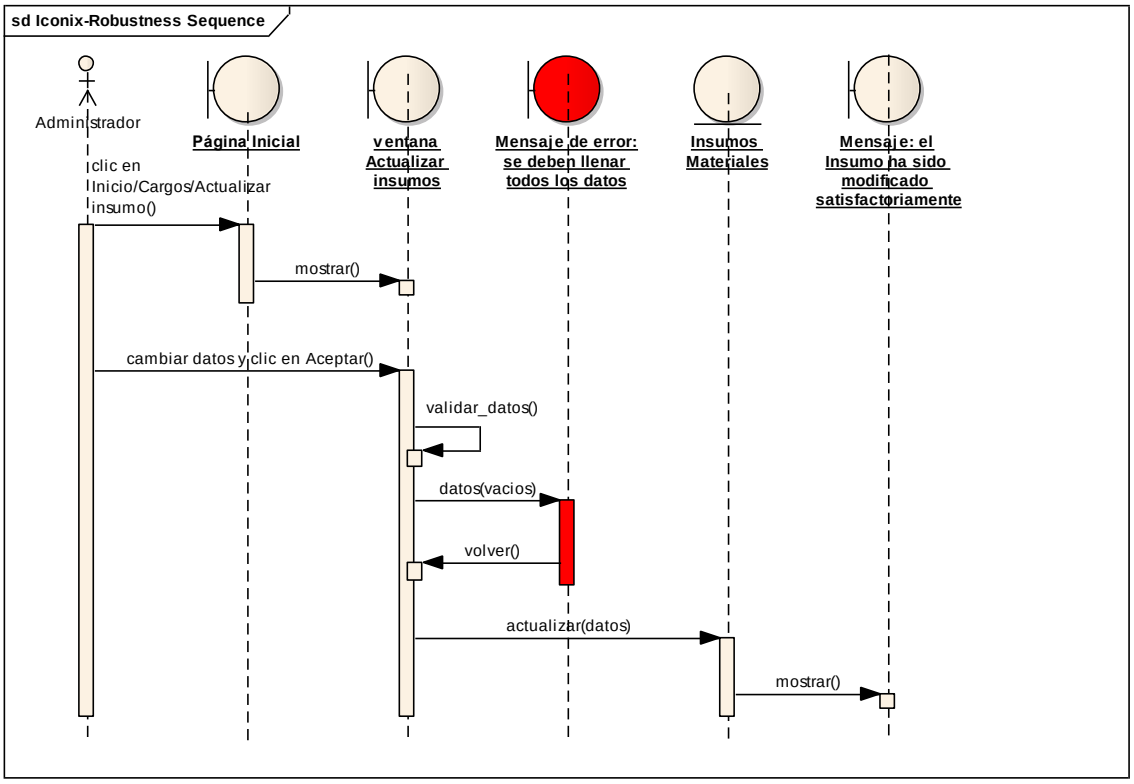



**Diagrama de Secuencia del CUS: Actualizar insumo**

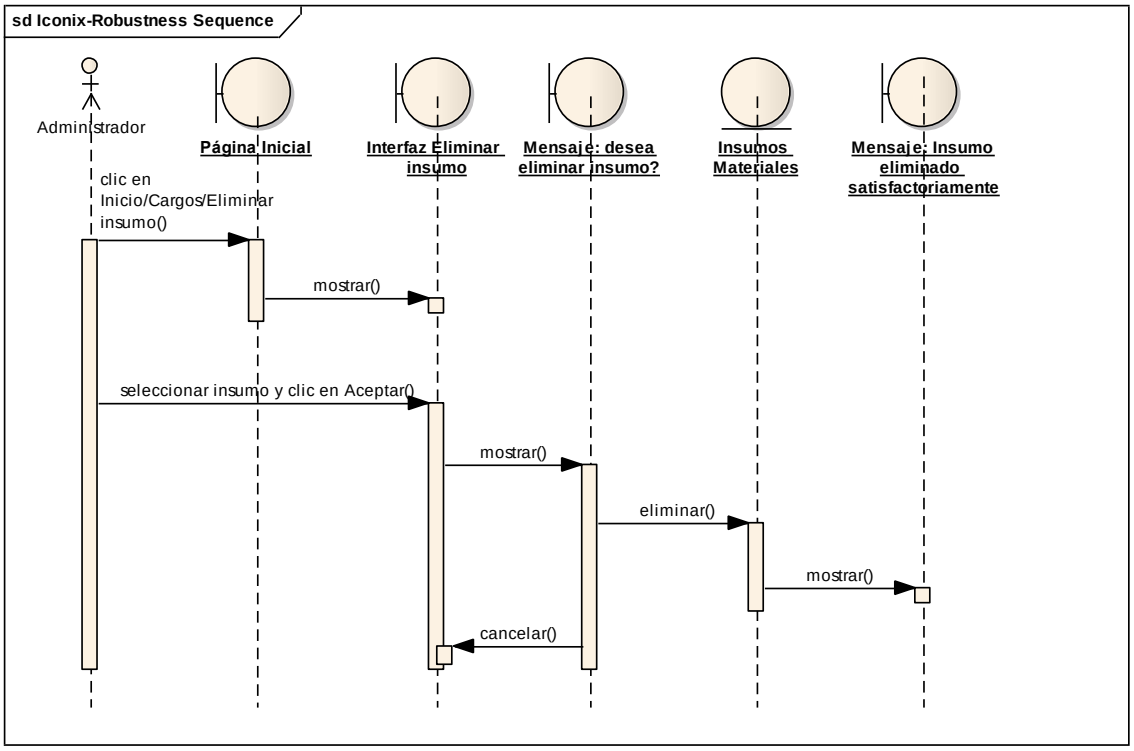



**Diagrama de Secuencia del CUS: Eliminar insumo**

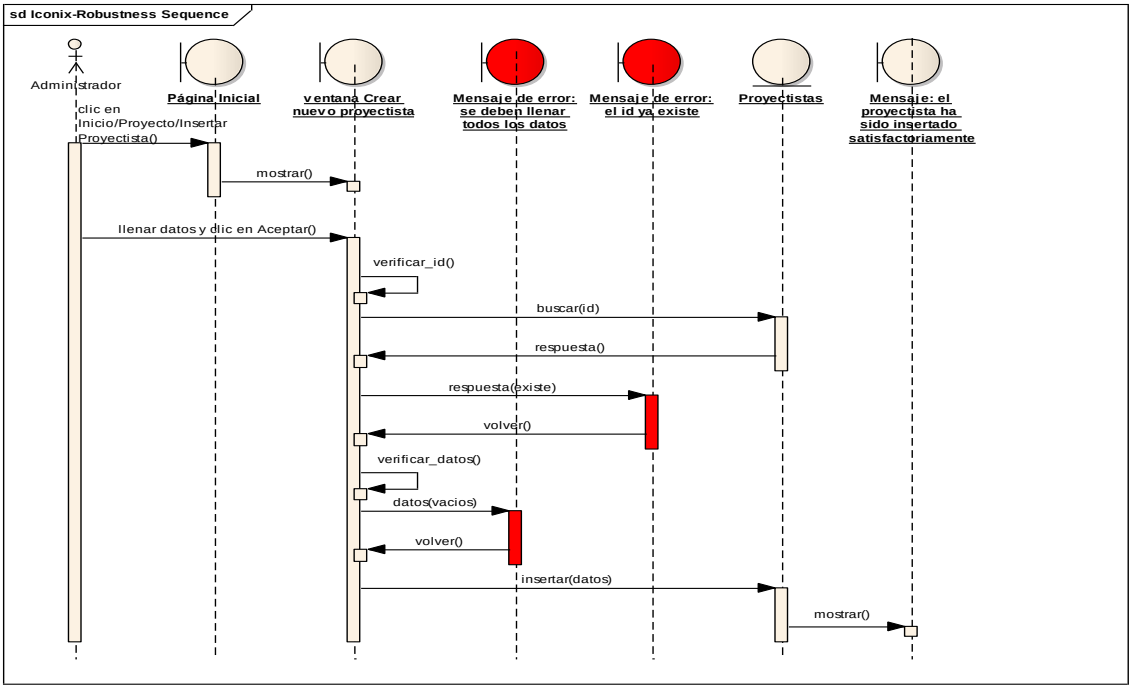



**Diagrama de Secuencia del CUS: Insertar proyectista**

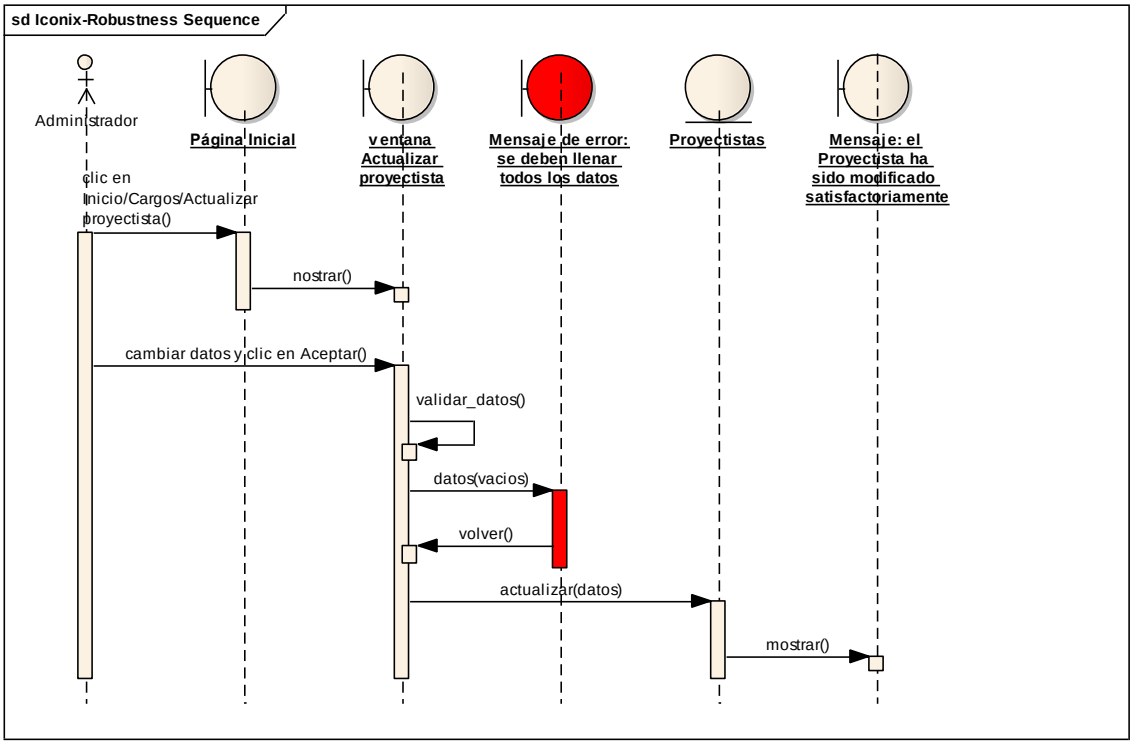



**Diagrama de Secuencia del CUS: Actualizar proyectista**

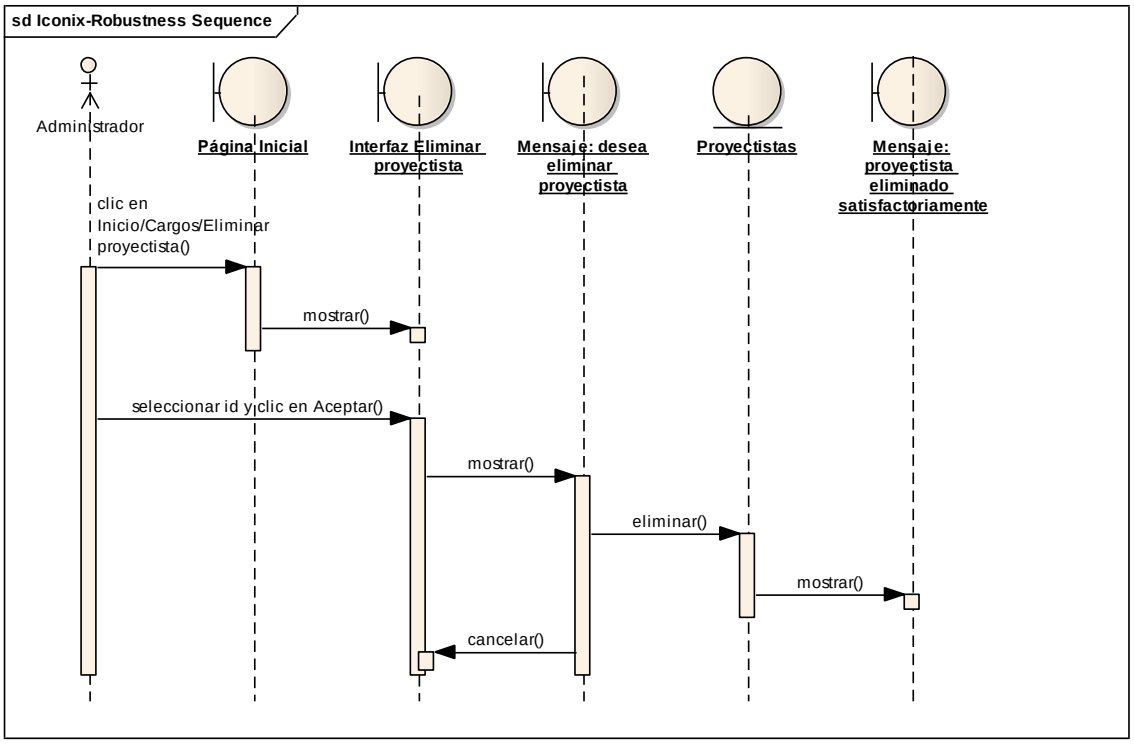



**Diagrama de Secuencia del CUS: Eliminar proyectista**

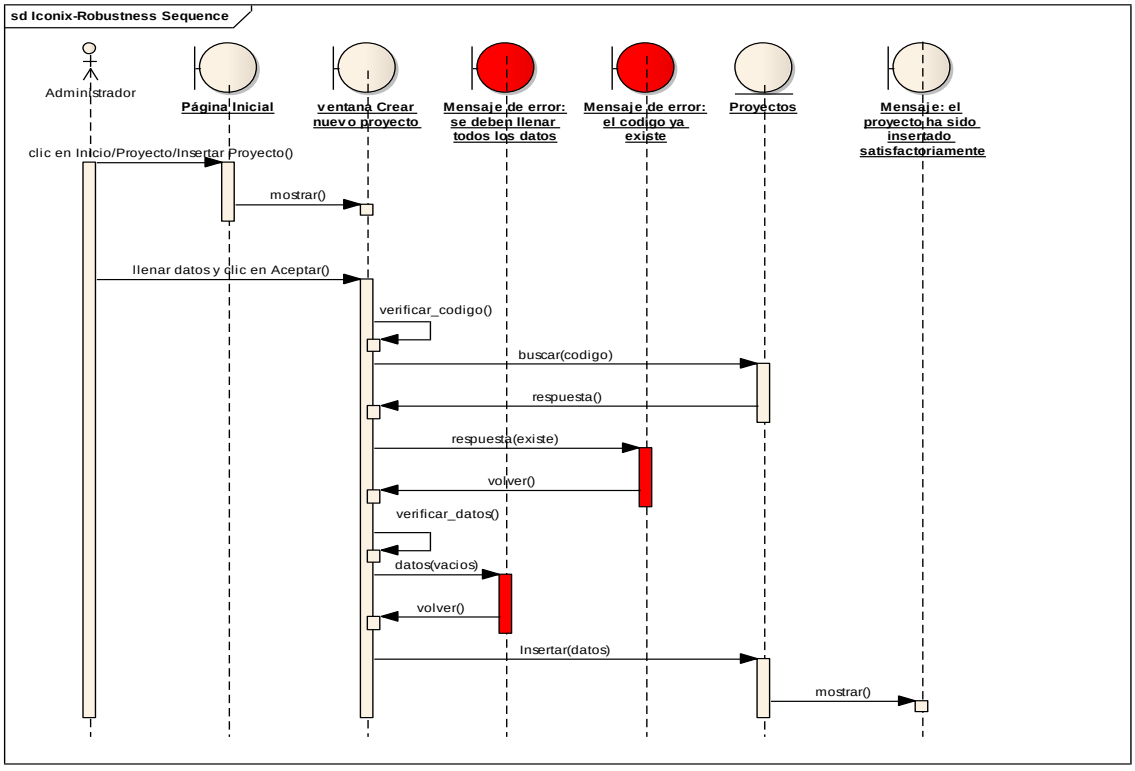



**Diagrama de Secuencia del CUS: Insertar proyecto**

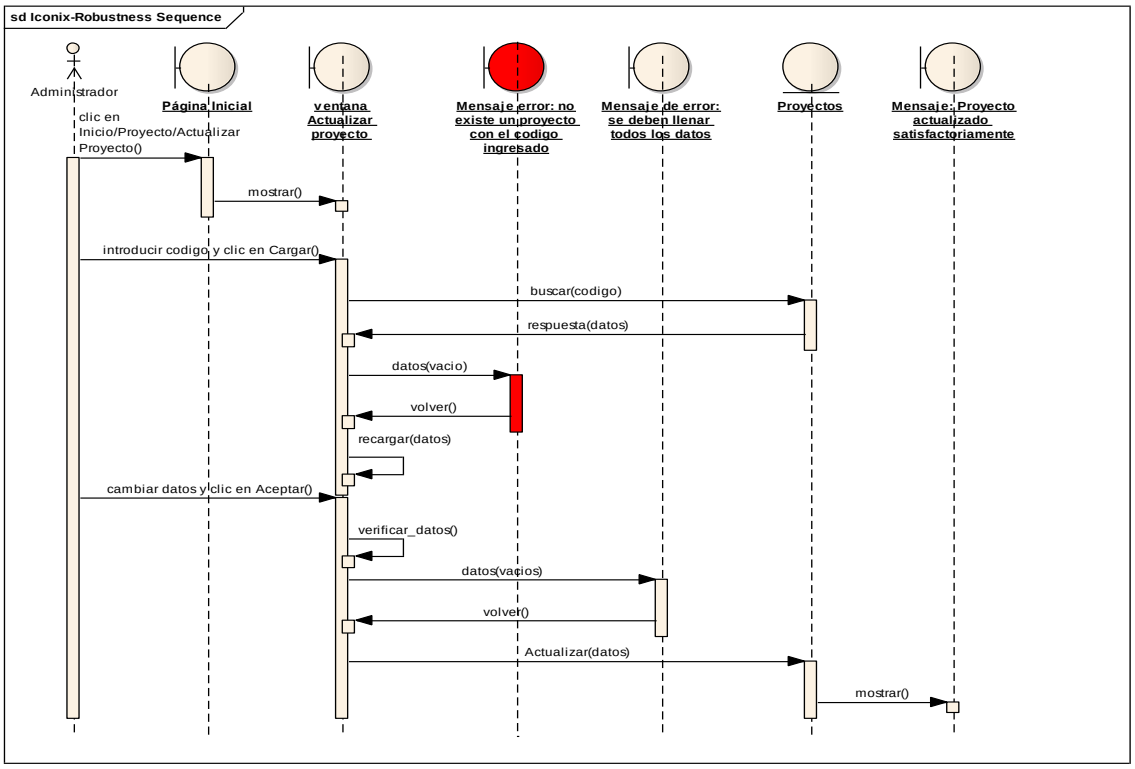



**Diagrama de Secuencia del CUS: Actualizar proyecto**

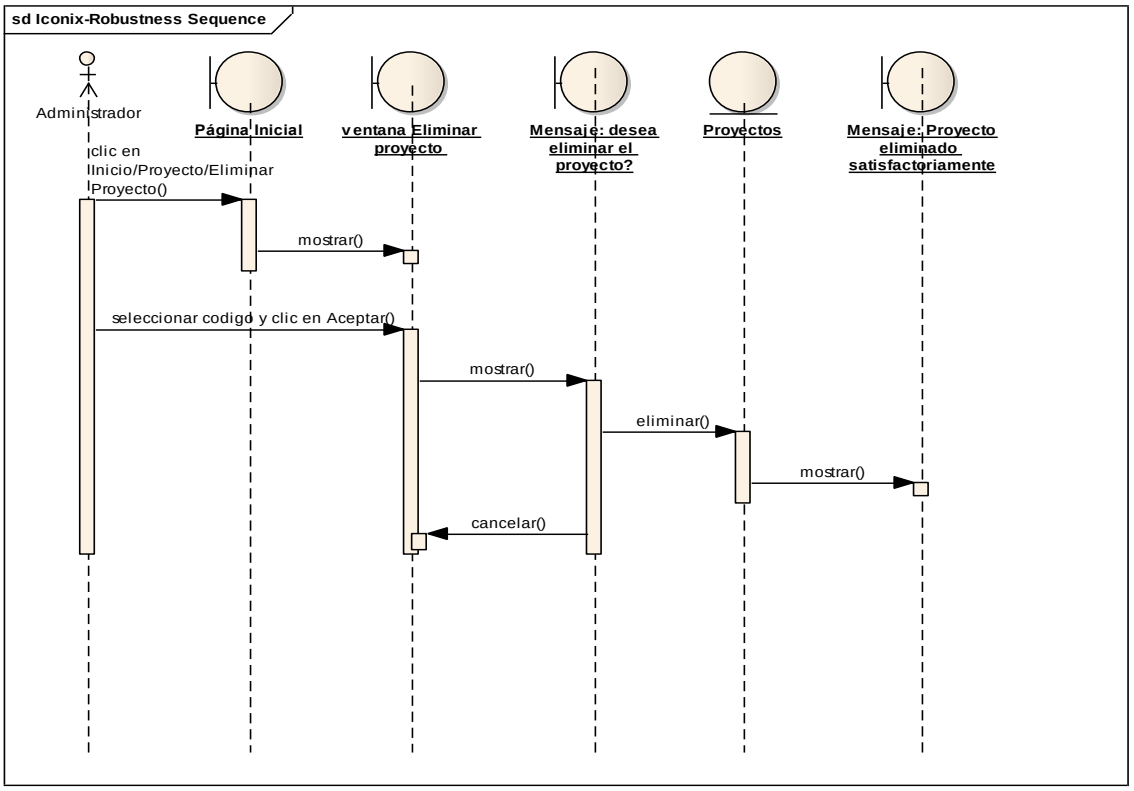



**Diagrama de Secuencia del CUS: Eliminar proyecto**

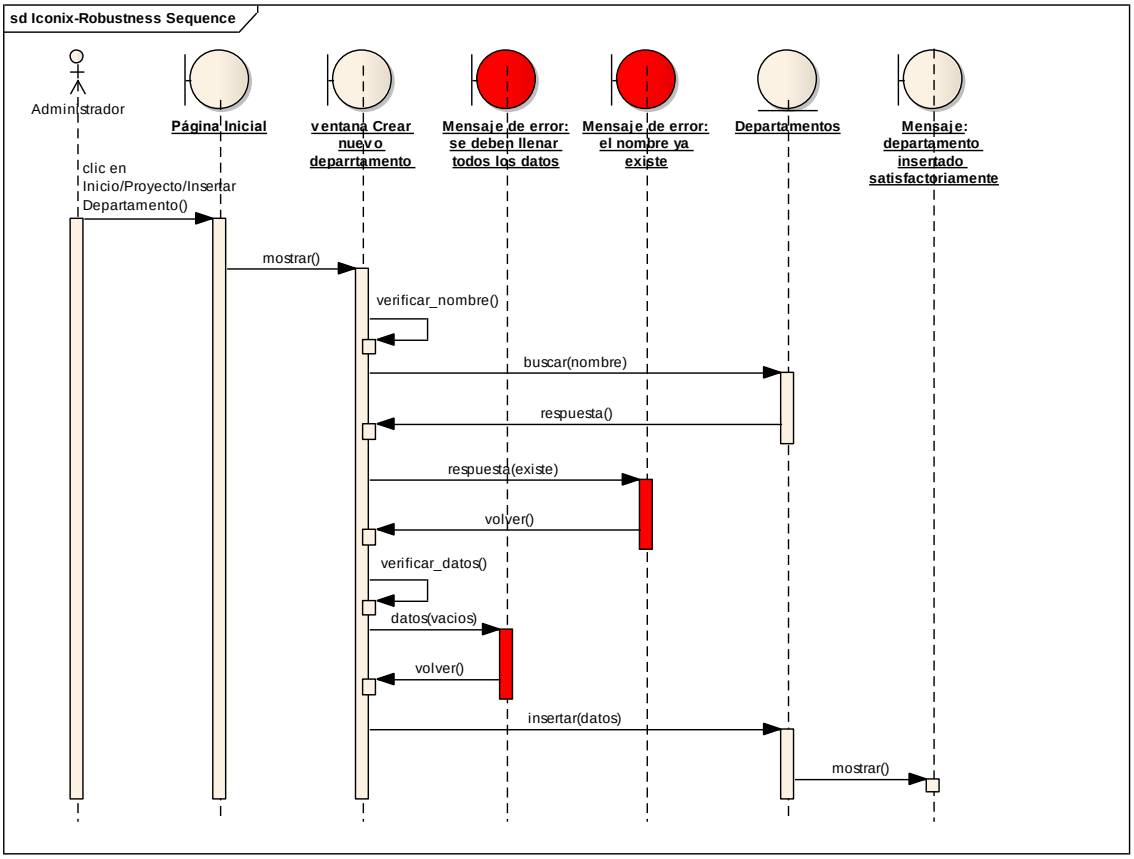



**Diagrama de Secuencia del CUS: Insertar departamento**

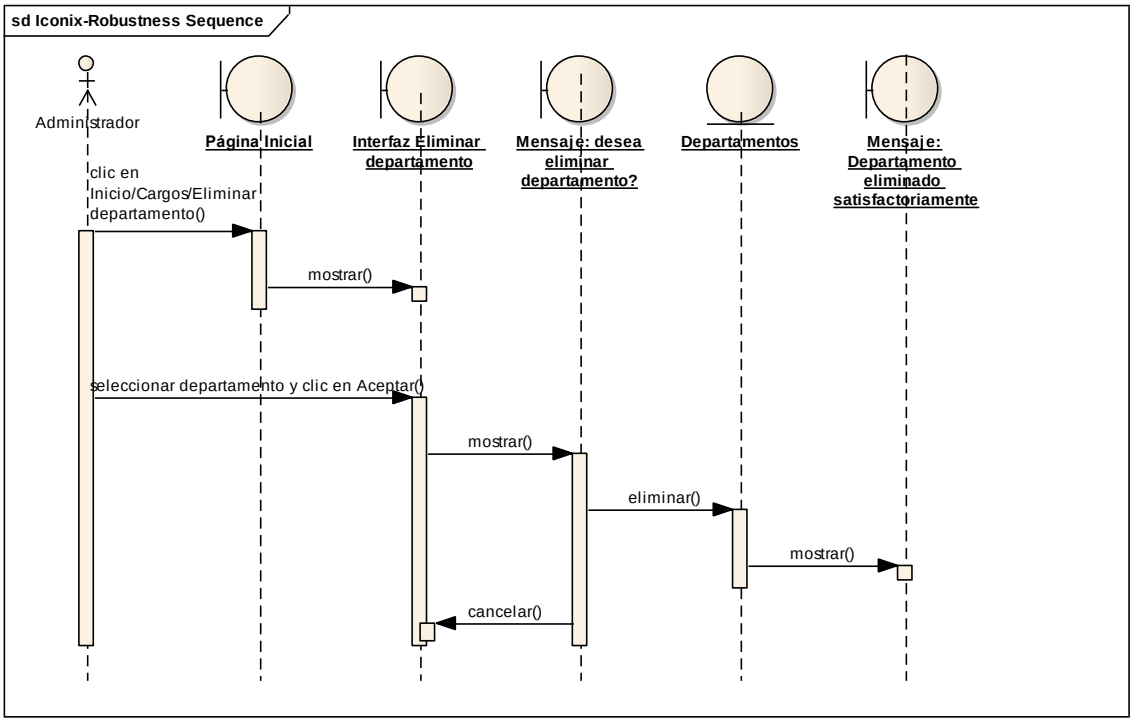
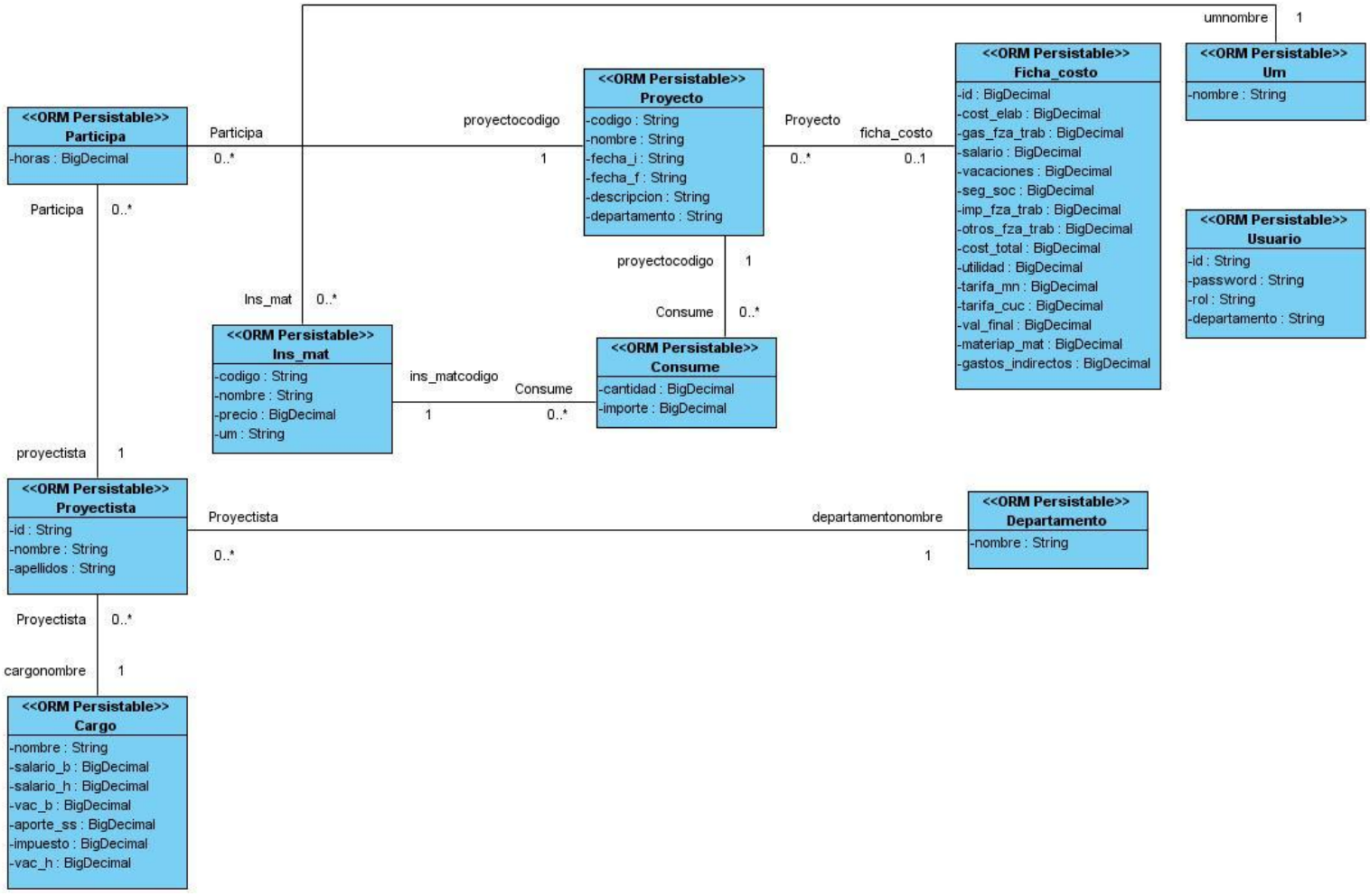



**Diagrama de Secuencia del CUS: Eliminar departamento**

## Anexo 4: Diagrama de Clases Persistentes



## Anexo 5: Diagrama de Clases

## Anexo 6: Estándar de Código

Historial de versiones del documento:

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 9/1/2013 | 1.0 | Estándar de código para la tesis | Edder Yoel Peña Rodríguez |

### 1- Organización del código

### 1.1- Aspectos generales

- Idioma:

El idioma que se va a emplear para nombrar los distintos elementos va a ser el español.

- Identación:

La identación va ser fija para todos elementos a 5 espacios.

- Anidamiento:

El anidamiento va a ser absoluto para cualquier tipo de instrucción a no más de 5 niveles.

- Tamaño máximo de líneas:

Las líneas de código no deben exceder los 80 caracteres. Por su parte, las de ruptura (continuación de una línea de código que excedió los 80 caracteres) no deben exceder a 35 caracteres.

- Módulos:

Los módulos no deben contener más de 10000 líneas de código.

- Apertura y cierre de ámbito (*begin-end*, {-}, etc):

Los ámbitos van a ser abiertos y cerrados en una línea aparte a la sentencia que los precede.

### 1.2 Líneas y espacios en blanco

- Líneas en blanco:

Se deben usar líneas en blanco antes y después de:

1- La declaración de una estructura o una clase.

2- La implementación del método de una clase.

3- Comentarios no relacionados con el código.

4- Bloques de códigos complejos.

- Espacios en blanco:

Se deben usar espacios en blanco antes y después de:

1- Operadores lógicos.

2- Operadores matemáticos.

### 1.3 Organización del código (Ficheros)

La carpeta raíz del proyecto se va a llamar Todavía sin definir. El sistema va a quedar subdividido en subsistemas. A continuación la información de cada uno de estos subsistemas:

## 2 Comentarios

Los comentarios serán escritos en español.

### 2.1 Comentarios de una línea

Se deben usar este tipo de comentarios en los siguientes casos:

- Explicación de métodos complejos.

### 2.2 Comentarios en bloque

Estos comentarios se aplicarán en los casos que se listan a continuación, y con la estructura mostrada:

Inicio de cada *unit*.

Estructura:

// Nombre del Producto

// Copyright (C) <lista de años> <nombre de la Empresa>

// Creado por:

```
// Versión Fecha de la última versión
// Cambios realizados con fecha
//-------------------------------------------------------------------
```

Después de la declaración de la interfaz de un método en la implementación.

Estructura:

```
void TClass::Method()
// Precondiciones: estado en que debe estar la clase antes de ejecutar
// Post condiciones: estado en que queda la clase después de ejecutar
// el método
// Si es necesario, un breve comentario del funcionamiento del método.
código1;
```

**3 Nombres**

**3.1 Nombre para Paquetes**

Los Paquetes no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

**3.2 Nombre para Unidades**

Las Unidades no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

**3.3 Nombre para Tipos**

Los Tipos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.4 Nombre para *Namespaces*

Los *Namespaces* no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.5 Nombre para Interfaces

Las Interfaces no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.6 Nombre para Clases

Las Clases no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.7 Nombre para Objetos

Van a mantener el mismo nombre de la clase de la que son tipo, quitándole el sufijo de clase.

### 3.8 Nombre para Atributos pasivos

Los Atributos pasivos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.9 Nombre para Funciones de Acceso

Las Funciones de Acceso no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre

indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.10 Nombre para Métodos

Los Métodos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.11 Nombre para Constructores

Los Constructores no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.12 Nombre para Parámetros

Los Parámetros no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.13 Nombre para Excepciones y sus objetos

Las Excepciones y sus objetos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.14 Nombre para Constantes

Las Constantes no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.15 Nombre para Arreglos

Los Arreglos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.16 Nombre para Variables locales

Las Variables locales no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

### 3.17 Nombre para Contadores de ciclo

Se van a utilizar letras, comenzando por la Q.

## 4 Declaraciones e inicializaciones

### 4.1 Aspectos generales

#### 4.1.1 Tipos de datos

No va a importar el lugar donde sean declarados los tipos de datos

### 4.2 Declaración de clases

#### 4.2.1 Orden de declaración

Por niveles de visibilidad

Los niveles de visibilidad van a ser declarados en orden descente, o sea, primero *public*, luego *protected*, y por último *private*).

Dentro de un nivel de visibilidad deben declarar primero los atributos y luego los métodos.

En grupos de elementos van a declarar los elementos en orden alfabético.

### 4.2.2 Constructores y destructores

Los constructores y destructores serán declarados encabezando el segmento de los métodos.

Se debe:

- Definir un constructor copia.
- No re declarar parámetros para que contengan un valor por defecto.
- Definir constructores virtuales para súper clases.
- Evitar muchos parámetros simples para los constructores.

### 4.2.3 Funciones sobrecargadas

Para estas funciones no va a importar el orden de declaración.

- Por último, no se debe declarar atributos pasivos como *protected* o *public*.

## 4.3 Declaración de funciones

### 4.3.1 Parámetros

El orden de declaración de los parámetros que no son por de defecto va a ser en orden decreciente de importancia. Por otro lado, en caso de que los parámetros sean por defecto, serán declarados primero los más propensos a ser utilizados.

Adicionalmente para los parámetros se debe:

- Proveer nombres formales en la declaración.
- Mismo nombre en la declaración y en la declaración.
- Evitar paso de parámetros por valor.
- Usar parámetros por defecto en lugar de sobrecarga de funciones.

### 4.3.2 Aspectos de complejidad

La complejidad ciclomática tiene que ser menor que 10. Mientras que la complejidad ciclomática extendida que 15. El número máximo de líneas que puede contener el cuerpo de una función es de 60, y la cantidad máxima de puntos de retorno va a ser 1.

**5 Base de Datos**

**5.1 Nombre para Base de Datos**

Las Bases de Datos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

**5.2 Nombre para Tablas**

Las Tablas no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

**5.3 Nombre para Campos**

Los Campos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

**5.4 Nombre para Procedimientos**

Los Procedimientos no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

**5.5 Nombre para *Triggers***

Los *Triggers* no requieren prefijo.

Las letras de la palabra estarán en Notación Camell. Es necesario que el nombre indique propósito, que contenga letras, y la longitud de este no debe exceder los 10 caracteres.

## Anexo 7: Estudio de Factibilidad con COCOMO

**Entradas externas (EI):** son todas aquellas entradas que le son proporcionadas al sistema.

| Nombre de las entradas externas | Cantidad de ficheros | Cantidad de elementos de datos | Clasificación |
|---|---|---|---|
| Entrar al sistema | 1 | 2 | Bajo |
| Insertar nuevo cargo | 1 | 2 | Bajo |
| Editar un cargo | 1 | 2 | Bajo |
| Eliminar un cargo | 1 | 1 | Bajo |
| Insertar un nuevo departamento | 1 | 1 | Bajo |
| Eliminar un departamento | 1 | 1 | Bajo |
| Insertar un nuevo insumo | 1 | 4 | Bajo |
| Actualizar un insumo | 1 | 4 | Bajo |
| Eliminar un insumo | 1 | 1 | Bajo |
| Insertar un nuevo proyectista | 1 | 5 | Bajo |
| Actualizar un proyectista | 1 | 5 | Bajo |
| Eliminar un proyectista | 1 | 1 | Bajo |

| | | | |
|---|---|---|---|
| Insertar un nuevo proyecto | 1 | 6 | Bajo |
| Actualizar un proyecto | 1 | 6 | Bajo |
| Eliminar un proyecto | 1 | 1 | Bajo |
| Insertar un nuevo usuario | 1 | 4 | Bajo |
| Actualizar un usuario | 1 | 4 | Bajo |
| Eliminar un usuario | 1 | 1 | Bajo |
| Crear valoración de un proyecto | 2 | 2 | Bajo |
| Modificar valoración de un proyecto | 2 | 2 | Bajo |
| Crear Cierre de producción | 2 | 1 | Bajo |

**Tabla Entradas externas (EI)**

Salidas externas (Norte): son las salidas asociadas al sistema que tiene elementos de filtraje de información

| Nombre de las peticiones | Cantidad de ficheros | Cantidad de elementos de datos | Clasificación |
|---|---|---|---|
| Imprimir una Valoración | 4 | 1 | Bajo |
| Imprimir Estado de la producción | 4 | 2 | Medio |
| Buscar un proyecto | 2 | 1 | Bajo |
| Imprimir cierre de Producción | 4 | 2 | Bajo |

**Tabla Salidas externas (Norte)**

Ficheros lógicos internos (ILF): son los ficheros lógicos o de almacenamiento de información que pertenecen al sistema

| Ficheros internos | Cantidad de ficheros | Cantidad de elementos de datos | Clasificación |
|---|---|---|---|
| cargo | 1 | 7 | Bajo |
| consume | 1 | 4 | Bajo |
| departamento | 1 | 1 | Bajo |
| ficha_costo | 1 | 15 | Bajo |
| ins_mat | 1 | 4 | Bajo |
| participa | 1 | 3 | Bajo |
| proyectista | 1 | 5 | Bajo |
| proyecto | 1 | 7 | Bajo |
| um | 1 | 1 | Bajo |
| usuario | 1 | 4 | Bajo |

**Tabla Ficheros internos (ILF)**

**Anexo 8: Encuesta para la determinación del coeficiente de competencias de expertos.**

**Nombre y apellidos: _____.**

**Cargo que desempeña: _____.**

Usted ha sido seleccionado como posible experto para ser consultado respecto al Sistema de gestión de información para la valoración económica de proyectos de la ENPA Holguín.

Necesitamos, antes de realizarle la consulta correspondiente, como parte del método empírico de investigación "consulta a expertos", determinar su coeficiente de competencia en este tema, a los efectos de reforzar la validez del resultado de la consulta que realizaremos. Por esta razón, se le solicita que responda las siguientes preguntas de la forma más objetiva que le sea posible.

1.- Marque con una cruz (X), en la tabla siguiente, el valor que se corresponde con el grado de conocimientos que usted posee sobre el tema. Considere que la escala que le presentamos es ascendente, es decir, el conocimiento sobre el tema referido va creciendo desde 0 hasta 10.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

2.- Realice una autovaloración del grado de influencia que cada una de las fuentes que le presentamos a continuación y ha tenido en su conocimiento y criterio sobre los indicadores puestos a su consideración. Para ello marque con una cruz (X), según corresponda, en **A** (alto), **M** (medio) o **B** (bajo).

| **Fuentes de argumentación.** | Grado de influencia de cada una de las fuentes. | | |
|---|---|---|---|
| | **A** (alto) | **M** (medio) | **B** (bajo) |
| *Análisis teóricos realizados por usted.* | | | |
| *Su experiencia obtenida.* | | | |
| *Trabajo de autores nacionales.* | | | |
| *Trabajo de autores extranjeros.* | | | |
| *Su propio conocimiento del estado del problema en el extranjero.* | | | |
| *Su intuición.* | | | |

Muchas Gracias

## Anexo 9: Encuesta aplicada a expertos

El Departamento de Proyectos de la ENPA de Holguín realiza un estudio para valorar el grado de satisfacción de los usuarios para el Sistema de gestión de información económica para la valoración de proyectos. Le agradeceríamos que contestara cuidadosamente el siguiente cuestionario y exprese su criterio. Considere que la opinión que usted aporte contribuirá mucho para nosotros. Le damos las gracias de antemano.

1. Actualmente usted es:

□ Directivo

□ Trabajador

2. Las preguntas que se realizan a continuación se consideran importantes para la evaluación del Sistema de gestión de información para la valoración económica de proyectos. Para ello se le proponen las siguientes categorías: Muy Adecuado (MA), Bastante Adecuado (BA), Adecuado (A), Poco Adecuado (PA), No Adecuado (NA). Marque con una **X** la categoría que considera adecuada para cada criterio:

| **CONCLUSIONES GENERALES** | | | | | |
|---|---|---|---|---|---|
| **Criterios** | **MA** | **BA** | **A** | **PA** | **NA** |
| 1. ¿Cómo evalúa la estructura organizativa del sistema? | | | | | |
| 2. ¿Cómo aprecia el diseño de las interfaces del sistema? | | | | | |
| 3. ¿Cómo aprecia el uso de los colores e imágenes? | | | | | |
| 4. ¿El sistema actual facilita la valoración económica de proyectos en la ENPA Holguín? | | | | | |
| 5. ¿Cómo evalúa la seguridad del sistema? | | | | | |
| 6. ¿Cómo evalúa la disponibilidad de los datos brindada por el sistema informático? | | | | | |

*Gracias por su amable opinión.*

**Anexo 10: Procesamiento de la encuesta de opinión de los expertos aplicando el método *Delphy***

| **Tabla de frecuencia absoluta** | | | | | | |
|---|---|---|---|---|---|---|
| **Criterios** | **MA** | **BA** | **A** | **PA** | **NA** | **TOTAL** |
| 1. ¿Cómo evalúa la estructura organizativa del sistema? | 11 | 5 | 4 | 0 | 0 | 20 |
| 2. ¿Cómo aprecia el diseño de las interfaces del sistema? | 10 | 8 | 2 | 0 | 0 | 20 |
| 3. ¿Cómo aprecia el uso de los colores e imágenes? | 7 | 8 | 5 | 0 | 0 | 20 |
| 4. ¿El sistema actual facilita la valoración económica de proyectos en la ENPA Holguín? | 13 | 5 | 2 | 0 | 0 | 20 |
| 5. ¿Cómo evalúa la seguridad del sistema? | 11 | 5 | 4 | 0 | 0 | 20 |
| 6. ¿Cómo evalúa la disponibilidad de los datos brindado por el Sistema *Web* en la aplicación? | 7 | 9 | 4 | 0 | 0 | 20 |

| **TABLA DE FRECUENCIA ABSOLUTA ACUMULADA** | | | | | |
|---|---|---|---|---|---|
| ***Criterios*** | **MA** | **BA** | **A** | **PA** | **NA** |
| 1. ¿Cómo evalúa la estructura organizativa del sistema? | 11 | 16 | 20 | 20 | 20 |
| 2. ¿Cómo aprecia el diseño de las interfaces del sistema? | 10 | 18 | 20 | 20 | 20 |
| 3. ¿Cómo aprecia el uso de los colores e imágenes? | 7 | 15 | 20 | 20 | 20 |
| 4. ¿El sistema actual facilita la valoración económica de proyectos en la ENPA Holguín? | 13 | 18 | 20 | 20 | 20 |
| 5. ¿Cómo evalúa la seguridad del sistema? | 11 | 18 | 22 | 24 | 20 |
| 6. ¿Cómo evalúa la disponibilidad de los datos brindado por el Sistema *Web* en la aplicación? | 11 | 16 | 20 | 24 | 24 |

| TABLA DEL INVERSO DE LA FRECUENCIA ABSOLUTA ACUMULADA | | | | |
|---|---|---|---|---|
| **Criterios** | **MA** | **BA** | **A** | **PA** |
| 1. ¿Cómo evalúa la estructura organizativa del sistema? | 0,55 | 0,8 | 1 | 1 |
| 2. ¿Cómo aprecia el diseño de las interfaces del sistema? | 0,5 | 0,9 | 1 | 1 |
| 3. ¿Cómo aprecia el uso de los colores e imágenes? | 0,35 | 0,75 | 1 | 1 |
| 4. ¿El sistema actual facilita la valoración económica de proyectos en la ENPA Holguín? | 0,65 | 0,9 | 1 | 1 |
| 5. ¿Cómo evalúa la seguridad del sistema? | 0,55 | 0,9 | 1,1 | 1,2 |
| 6. ¿Cómo evalúa la disponibilidad de los datos brindado por el Sistema *Web* en la aplicación? | 0,4583 | 0,6667 | 0,83333 | 1 |

| TABLA DE DEREMINACIÓN DE LOS PUNTOS DE CORTES | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Criterios** | **MA** | **BA** | **A** | **PA** | **Suma** | **Promedio** | **N - Prom.** |
| 1. ¿Cómo evalúa la estructura organizativa del sistema? | 0,13 | 0,84 | 3,49 | 3,49 | 7,95 | 1,99 | -1,99 |
| 2. ¿Cómo aprecia el diseño de las interfaces del sistema? | 0 | 1,28 | 3,49 | 3,49 | 8,26 | 2,07 | -1,99 |
| 3. ¿Cómo aprecia el uso de los colores e imágenes? | -0,39 | 0,67 | 3,49 | 3,49 | 7,26 | 1,82 | -2,07 |
| 4. ¿El sistema actual facilita la valoración económica de proyectos en la ENPA Holguín? | 0,39 | 1,28 | 3,49 | 3,49 | 8,65 | 2,16 | -1,82 |
| 5. ¿Cómo evalúa la seguridad del sistema? | 0,13 | 1,28 | 1,28 | 3,49 | 8,65 | 2,16 | -2,16 |
| 6. ¿Cómo evalúa la disponibilidad de los datos brindado por el Sistema *Web* en la aplicación? | -0,1 | 0,43 | 0,97 | 3,49 | 4,79 | 1,2 | -2,16 |

| **CONCLUSIONES GENERALES** | | | | | |
|---|---|---|---|---|---|
| **Criterios** | **MA** | **BA** | **A** | **PA** | **NA** |
| 1. ¿Cómo evalúa la estructura organizativa del sistema? | **Si** | **-** | | | |
| 2. ¿Cómo aprecia el diseño de las interfaces del sistema? | **Si** | **-** | | | |
| 3. ¿Cómo aprecia el uso de los colores e imágenes? | **Si** | **-** | | | |
| 4. ¿El sistema actual facilita la valoración económica de proyectos en la ENPA Holguín? | **Si** | **-** | | | |
| 5. ¿Cómo evalúa la seguridad del sistema? | **Si** | **-** | | | |
| 6. ¿Cómo evalúa la disponibilidad de los datos brindado por el Sistema *Web* en la aplicación? | **Si** | **-** | | | |