Universidad de las Ciencias Informáticas

Facultad 5

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Capa de interfaz de usuario para el proyecto Sistema Integral de Confiabilidad Operacional

Autor: Mario Luis García Lugones

Tutor: Ing. Marisniulkis Lescaille Cos

La Habana

Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Mario Luis García Lugones	Ing. Marisniulkis Lescaille Cos
Autor	Tutor

AGRADECIMIENTOS

Agradecer primero que todo a mis padres por la educación que me han dado durante toda mi vida. A mis hermanos más queridos que son todos. A mi tía que se ha convertido en mi segunda madre. A mi familia en general agradecer su apoyo incondicional.

Agradecer a todas las personas que de una forma u otra hicieron posible la realización de este trabajo. Especialmente a mi tutora, por su ayuda, por su amistad.

DEDICATORIA

Les dedico este trabajo a mis padres por ser todo para mí. A mi madre querida que siempre está y estará presente en mí.

A mis hermanos.

RESUMEN

Actualmente en la Universidad de las Ciencias Informáticas (UCI) se llevan a cabo una serie de proyectos productivos, específicamente en la Facultad 5 donde radica el Centro de Informática Industrial (CEDIN), que cuenta, entre otras, con la línea Gestión Industrial donde se está desarrollando el proyecto Sistema Integral de Confiabilidad Operacional (SIC), con el objetivo de generar herramientas e instrumentos, que permitan mejorar la estimación del impacto de las acciones realizadas sobre los sistemas técnicos complejos durante las fases del ciclo de vida.

Este proyecto no cuenta con una interfaz gráfica de usuario que facilite su gestión, por lo que se decidió desarrollar el diseño e implementación de una interfaz web para gestionar, facilitar y mejorar el acceso a la información manejada en el sistema, cumpliendo con los estándares web internacionales, así como con los requerimientos de usuario.

En este trabajo se abordan los fundamentos teóricos, así como la descripción de la metodología de desarrollo, las herramientas y tecnologías seleccionadas para el desarrollo del sistema. Se especifica la arquitectura sobre la cual se basa el desarrollo y los diagramas que ilustran los elementos del diseño de la aplicación. El resultado final del trabajo es un sistema que sirve como interfaz de usuario a las funcionalidades que brinda el proyecto SIC.

PALABRAS CLAVE

Confiabilidad operacional, gestión, interfaz de usuario, mantenimiento.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Introducción

El desarrollo que ha alcanzado la sociedad en materia de Tecnologías de la Informática y las Comunicaciones, ha propiciado que una gran variedad de tareas o trabajos que comúnmente eran difíciles de resolver de forma manual, sin la utilización de computadoras, actualmente tengan soluciones factibles, liberando al hombre de tareas tediosas o costosas de realizar. Los sistemas de gestión y/o mantenimiento han sido entre los más implementados por las empresas en el proceso de adopción de las nuevas tecnologías. En el caso de los sistemas de mantenimiento, donde es difícil determinar una relación directa entre el tiempo de vida útil y la probabilidad de falla de los equipos, se hace necesario adoptar nuevas filosofías de mantenimiento que permitan mantener la calidad de los procesos y garantizar la integridad de los equipos.

Ante esta panorámica, una vía efectiva que le permite a las organizaciones enfrentar de manera eficiente el nuevo reto, son los conceptos y principios de Confiabilidad Operacional, la cual lleva implícita la capacidad de una instalación (procesos, tecnologías y personas) para cumplir su función dentro de sus límites de diseño y bajo un específico contexto operacional.

A lo largo de las últimas décadas se han desarrollado varios modelos, métodos y técnicas que permiten cuantificar y hacer valoraciones del impacto que tiene la confiabilidad operacional en las diversas acciones que se realizan sobre un sistema técnico complejo. Esto ha llevado a otro nivel el desarrollo de los sistemas de confiabilidad operacional donde es necesaria cada vez más una mayor calidad en sus diseños para satisfacer las necesidades del cliente. Lo que hace que estos sistemas automatizados sean cada vez más complejos, ya que las necesidades de los clientes son directamente proporcionales al desarrollo de los mismos.

Por tanto, los sistemas desarrollados para las instituciones y empresas donde se pongan en práctica los modelos, métodos y técnicas para aplicar los conceptos y principios de la confiabilidad operacional que permitan mejorar la estimación del impacto de las acciones realizadas sobre los sistemas técnicos, deben contar con una interfaz gráfica de usuario amigable y accesible a todos los usuarios, de manera que sea más natural para los usuarios la utilización del sistema y pueda ser usado con mayor facilidad por la mayor cantidad de clientes.

Una interfaz debe ser diseñada generalmente según los conocimientos que tenga el desarrollador, pero específicamente debe cumplir con los requerimientos del usuario final. Ya que el mejor sistema o herramienta, son inútiles si no se puede interactuar con ellas. (Lacalle, 2010) Las interfaces de usuario posibilitan a través de elementos visuales una interacción amigable con otros componentes más complejos de los sistemas informáticos.

La Universidad de las Ciencias Informáticas (UCI), apoyándose en sus centros productivos potencia el desarrollo de software. Específicamente en la Facultad 5 radica el Centro de Informática Industrial (CEDIN), entre las líneas asociadas a este centro se encuentra la de Gestión Industrial, donde se está desarrollando el proyecto Sistema Integral de Confiabilidad Operacional (SIC). El cual posee como objetivo generar herramientas e instrumentos que permitan mejorar la estimación del impacto de las acciones realizadas sobre los sistemas técnicos complejos, durante las fases del ciclo de vida, relacionadas con la operación y el mantenimiento, a través de la vinculación de las diferentes ramas de estudio de esta ciencia.

De este proyecto actualmente existe una versión de escritorio que fue desarrollada para Venezuela bajo una licencia propietaria que no permite la distribución del software, por lo que se decidió desarrollar una versión basada en la web para nuestro país. Pero en la etapa de desarrollo que se encuentra hoy este proyecto no cuenta con un medio de interacción entre los usuarios y los diferentes componentes de la aplicación que posibilite la gestión de la información de manera fácil, dificultando el control estricto sobre la calidad en el proceso y la integridad de los equipos durante la gestión y su mantenimiento.

Por todo lo expuesto se plantea como **problema científico** de este trabajo: ¿Cómo brindar al usuario la posibilidad de una interacción fluida y dinámica con el sistema para la gestión y mantenimiento de los equipos?

Teniendo como objeto de estudio: Proceso de desarrollo de interfaces gráficas de usuario en la web.

Y como campo de acción: Proceso de desarrollo de interfaces gráficas de usuario en la web para el proyecto SIC.

Por todo lo anterior, **se definió como objetivo general:** Desarrollar la capa de interfaz de usuario para el proyecto SIC.

Con el fin de alcanzar el objetivo planteado se definen los siguientes **objetivos específicos**:

- Diseñar la capa de interfaz de usuario para el proyecto SIC cumpliendo con los requerimientos y los casos de uso definidos para el sistema.
- Implementar las interfaces para el proyecto SIC de acuerdo con las descripciones de los casos de uso y los prototipos de interfaces disponibles.

Con el propósito de cumplir con todos los objetivos se definen las siguientes **tareas investigativas**:

- Determinación de las tendencias actuales de los sistemas de gestión y mantenimiento de equipos.
- Caracterización de los frameworks existentes que facilitan el desarrollo de la aplicación y selección del más adecuado para su creación.
- Definición de la metodología de desarrollo que va a regir el proceso de desarrollo de las interfaces.
- Análisis de los requisitos que deben cumplir las interfaces acorde con la descripción de los casos de uso y los prototipos de interfaz disponibles para un mejor desarrollo de la aplicación.
- Desarrollo de las pruebas, con los casos de prueba correspondientes, que posibilitarán la validación de las interfaces.

Para dar cumplimiento a las distintas tareas antes mencionadas, se utilizaron los siguientes métodos de investigación:

1. **Métodos teóricos:**

   - **Analítico – sintético:** Permite procesar toda la información referente al desarrollo de interfaces gráficas para poder analizar y diferenciar dicha información con el objetivo de arribar a conclusiones que sustenten la necesidad de la investigación.
   - **Inductivo – deductivo:** Permite elaborar los estándares, tanto de codificación y diseño gráfico, como las clases a utilizar por las interfaces y a su vez, determinar las especificaciones de cada una de ellas.
   - **Histórico – lógico:** Este método posibilita conocer los antecedentes y tendencias actuales de las interfaces gráficas, como también la evolución de las mismas y basado en estas observaciones tomarlas como guía para el desarrollo de la presente investigación.

2. **Métodos empíricos:**

   - **Experimento:** Este método favorece el desarrollo de las pruebas para la validación de las interfaces.

- **Revisión bibliográfica:** Se utiliza para realizar un estudio de un conjunto de fuentes de información que estén vinculadas al tema de desarrollo de interfaces, así como libros, artículos y revistas de gran utilidad para documentar la base teórica del trabajo a desarrollar.

Aporte práctico que se espera del trabajo:

El trabajo desarrollado debe dar la posibilidad de contar con una integración de la interfaz con otras unidades funcionales desarrolladas para el Sistema Integral de Confiabilidad Operacional. Además, con la implementación de una interfaz accesible y amigable para todos los usuarios.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción.

En este capítulo se realizará un estudio sobre las tendencias actuales de las interfaces gráficas de usuario, también sobre sus clasificaciones y características así como un estudio de los diferentes estándares internacionales empleados en el diseño de la capa de presentación de todo sistema de gestión. Se describirán los *frameworks*[1] más utilizados tanto en el mundo como en la universidad para la implementación de interfaces de usuario y los Entornos de Desarrollo Integrado (IDE) más utilizados en el desarrollo de las mismas. Además, se analizan las características fundamentales de las tecnologías, metodologías y herramientas de desarrollo que se utilizarán en la creación del sistema. Por tanto, se manejan un conjunto de conceptos y terminologías que se incorporan a las interfaces de usuario, realizándose a la vez breves valoraciones de las mismas que ayuden a alcanzar los objetivos propuestos con el trabajo de diploma.

1.2 Interfaz de usuario.

La idea fundamental en el concepto de interfaz es el de mediación, entre hombre y máquina. La interfaz es lo que "media", lo que facilita la comunicación, la interacción, entre dos sistemas de diferente naturaleza, típicamente el ser humano y una máquina como el computador. Esto implica, además, que se trata de un sistema de traducción, ya que los dos "hablan" lenguajes diferentes: *verbo-icónico*[2] en el caso del hombre y binario en el caso del procesador electrónico. (Toledo, 2008)

De una manera más técnica se define interfaz de usuario, como conjunto de componentes empleados por los usuarios para comunicarse con las computadoras. El usuario dirige el funcionamiento de la máquina mediante instrucciones, denominadas genéricamente “entradas”. Las entradas se introducen mediante diversos dispositivos, por ejemplo un teclado, y se convierten en señales electrónicas que pueden ser

[1] Framework: Marco de trabajo que constituye una estructura conceptual y tecnológica compuesta por componentes de software que permiten el desarrollo rápido de aplicaciones.
[2] Verbo-Icónico: Hace referencia a la comunicación mediante palabras e imágenes.

procesadas por la computadora. Estas señales se transmiten a través de circuitos conocidos como *bus*[3], y son coordinadas y controladas por la unidad de proceso central (CPU) y por un soporte lógico conocido como sistema operativo. (Toledo, 2008)

Otra definición de interfaz de usuario plantea: "La interfaz de usuario es el vínculo entre el usuario y el programa de computadora. Una interfaz es un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa." (Myers, 1996)

Debido a estos conceptos y definiciones de interfaz de usuario se puede llegar a la conclusión de que las interfaces son la parte más importante de cualquier programa ya que determinan que tan fácilmente es posible que el programa haga lo que el usuario quiere hacer.

1.2.1 Clasificación de las Interfaces.

Cuando se habla de interfaces de usuario se puede pensar en la existencia de una gran variedad de ellas, donde cada cual puede clasificarla en uno o determinados grupos, pero se pueden destacar dos formas de clasificación: (Toledo, 2008)

Interfaz de hardware: Desde el punto de vista de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla.

Interfaz de software: Destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observa habitualmente en la pantalla.

1.2.2 Tipos de Interfaces.

La evolución de las interfaces de usuario marcha en paralelo con la de los sistemas operativos; de hecho, la interfaz constituye actualmente uno de los principales elementos de un sistema operativo. (Toledo, 2008) Algunas de las distintas interfaces que han ido apareciendo desde sus inicios son las interfaces de **Línea de mandatos**. Estas poseen la desventaja de tener una ayuda difícil de leer, además de tener que memorizar un gran número de instrucciones a la hora de interactuar con ellas debido a que los nombres de las instrucciones muchas veces parecen inadecuados. Aunque pueden ser interfaces muy potentes y flexibles si son controladas por usuarios experimentados.

[3] Circuitos bus: Es una ruta compartida que conecta diferentes partes del sistema como el procesador, la memoria y los puertos de entrada, salida, permitiéndoles transmitir información.

También se pueden mencionar las **Interfaces de menús,** las cuales permiten navegar dentro de un sistema, seleccionar propiedades o acciones de un objeto. En este tipo de interfaces podemos encontrar distintos tipos de menús, como son: menús de pantalla completa, menús de barra, menús en cascada y menús contextuales.

En esta lista de interfaces se encuentran las **Interfaces gráficas de usuario (GUI),** que son las que más predominan hoy en día debido a que están basadas en íconos y elementos gráficos que logran en el usuario que interactúe con ellas un nivel mayor de aceptación. También se puede mencionar que estas interfaces no requieren un uso de la memoria a largo plazo, por lo que son muy fáciles de usar. (Toledo, 2008) Por tanto, en el desarrollo de estas interfaces es hacia donde se enfocará este trabajo.

1.2.3 Características de las GUI.

En la actualidad, una buena interfaz gráfica de usuario debe contar con un conjunto de características que le brinden al usuario comodidad y sobre todo que se sienta ubicado y guiado todo el tiempo que dure la interacción, por lo que algunas de estas características podrían sintetizarse en que:

Promueven la consistencia entre programas y permiten la transferencia de información entre los mismos. Siguen el paradigma de la interacción objeto-acción. Posibilitan a los usuarios ver en la pantalla los gráficos y textos tal como se verán impresos, proporcionan una respuesta visual de sus acciones y permiten personalizar la interfaz y la manera de interactuar con ella.

Las interfaces gráficas de usuario posibilitan además flexibilidad en el uso de dispositivos de entrada como lo son el teclado y el ratón, así como ofrecer la información visual de las acciones y modos del usuario-sistema. También deben dar la posibilidad de que se puedan manipular en la pantalla directamente los objetos y la información para que de esta forma, la puedan visualizar de manera directa como lo son los íconos y las ventanas. (Toledo, 2008)

Sin dudas una interfaz que cumpla con todas estas características logrará en el usuario un mayor nivel de aceptación, por la comodidad y la flexibilidad a la hora de la interacción usuario-sistema.

El objetivo de las interfaces es servir de puente comunicativo, de mediación entre el usuario y el sistema con el que se interactúe. Estos sistemas pueden ser aplicaciones de escritorio, o aplicaciones web, las cuales debido al desarrollo de la red de redes y a que eliminan las restricciones espacio-temporales en la

interacción con los sistemas, están siendo cada vez más usadas en el desarrollo de aplicaciones de gestión.

1.3 Aplicación Web y la Web 2.0.

Una aplicación web es un software basado en tecnologías y estándares de la World Wide Web Consortium (W3C), que provee recursos específicos tales como contenidos y servicios a través de una interfaz de usuario a la que puede accederse utilizando un navegador web. (Kappel, 2006) Intenta portar las clásicas aplicaciones de escritorio hacia entornos web, que son utilizadas a través de los distintos navegadores, agregando portabilidad y capacidad de acceder desde diferentes dispositivos. Una de las ventajas más significativas de las aplicaciones web es su forma de instalación y distribución, ya que normalmente instalar una aplicación web consiste en configurar los componentes del lado del servidor en la red y no necesariamente una instalación o configuración en el lado del cliente. Son independientes de las tecnologías de hardware y software (sistema operativo) que se utilicen.

La evolución de la web ha sido continua y recientemente por la revolución que ha constituido el uso masivo de internet, un fenómeno de explosión de contenidos y de aplicaciones, en una transición no solo de tecnología, sino un cambio de actitud y una manera distinta de ver y hacer las cosas, ha generado lo que se denomina web 2.0. (Henst, 2005)

La web 2.0 tiene muchas definiciones, es un fenómeno más que una tecnología en sí, que tiene su base en la interacción de diferentes aplicaciones que facilitan el acceso y uso de la información, siendo la interoperabilidad, el diseño centrado en el usuario y la colaboración, los factores fundamentales detrás de este concepto. Ejemplos de web 2.0 son las comunidades en la red, las aplicaciones web, los servicios de red social, los servicios de alojamiento, las wikis y los blogs.

El uso de la web 2.0 permite además de localizar y obtener información, que era el paradigma de la web antes de la aparición de este, el interactuar con otros usuarios o contribuir en la creación de los contenidos de los sitios web. La web 2.0 es una forma relativamente nueva de ver la web y de usarla, no incluye formalidades técnicas. Le proporciona al usuario mayor interacción, control de sus datos y proyección en la web e inserta nuevos modelos de negocio en la red.

1.4 Estándares web de la W3C.

Los estándares web son tecnologías, establecidas por la **W3C** y otros organismos de normalización, que se utilizan para crear e interpretar contenido basado en web. Estas tecnologías están diseñadas para el futuro de prueba de los documentos publicados en la web y hacer los documentos accesibles al mayor número de personas y dispositivos como sea posible. (Johansson, 2008)

Existen varios estándares que garantizan una mejor calidad, accesibilidad y rapidez de las aplicaciones web. Un estándar no es más que un modelo a seguir al realizar una acción. Son documentos que dan los detalles técnicos y las reglas necesarias para que un producto o tecnología se use correctamente. (Johansson, 2008)

Cuando se habla de estándares web no se pueden dejar de mencionar todas las ventajas que a la hora de desarrollar aplicaciones web hacen que deban tenerse en cuenta para el desarrollo. Dentro de esas ventajas puede mencionarse la insistencia en que se escriba código limpio, válido, modular y semánticamente correcto.

Estos no se aplican solo para la compatibilidad al futuro, también funcionan con los navegadores más viejos donde el sitio se va a degradar levemente para producir un resultado aceptable en esos navegadores.

Potencian que el contenido sea accesible a una gama más amplia de usuarios, incluidos aquellos con discapacidades.

Cuando el código de marcas (o etiquetas) está más limpio, los dispositivos especializados para las personas con problemas visuales pueden leer el contenido más fácilmente. Esto también permite a los usuarios personalizar las páginas a sus necesidades utilizando los conmutadores de estilo y ofrece una versión para imprimir de las páginas web.

Permiten mejor optimización de motores de búsqueda (*Search Engine Optimization*), es decir, se puede lograr un mejor ranking en los motores de búsqueda. Si el código está muy cargado (resultado de no

aplicar estos estándares), será más difícil para las arañas *(Spiders[4])* de los motores de búsqueda localizar e indexar el contenido.

Otra de las ventajas es que simplifican el mantenimiento del sitio, porque al utilizar CSS (*Cascading Style Sheets,* Hojas de Estilo en Cascada) para separar el contenido de la presentación, gran parte de los cambios se hacen mucho más fáciles, es decir, solo se tendrá una hoja de estilos en lugar de estilos en todas las páginas. El ahorrar ancho de banda también figura entre una de sus cualidades ya que con menos caché de código CSS y archivos, se podrá ahorrar un ancho de banda considerable. (Jorge, 2012)

Entre los principales estándares de la W3C para el desarrollo de aplicaciones web se encuentran: (Alonso, 2005)

- HTML: Lenguaje de Etiquetado de Hipertexto (*HyperText Markup Language*). Es un lenguaje comúnmente utilizado para la publicación de hipertexto en la Web y desarrollado con la idea de que cualquier persona o tipo de dispositivo pueda acceder a la información en la web. HTML utiliza etiquetas que marcan elementos y estructuran el texto de un documento.
- XML: Lenguaje de Etiquetado Extensible (*Extensible Markup Language*). Es un lenguaje con una importante función en el proceso de intercambio, estructuración y envío de datos en la web. Describe los datos de tal manera que es posible estructurarlos utilizando para ello etiquetas, como lo hace HTML, pero que no están predefinidas, delimitando de esta manera los datos, a la vez que favorece la interoperabilidad de los mismos.
- XHTML: Lenguaje de Etiquetado de Hipertexto Extensible(*Extensible HyperText Markup Language*). Es una versión más estricta y limpia de HTML que nace ante la limitación de uso de este con las herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñada para describir los datos.
- DOM: Modelo de Objetos del Documento (*Document Object Model*). Es una plataforma que proporciona un conjunto estándar de objetos a través del cual se pueden crear documentos HTML y XML, navegar por su estructura y modificar, añadir y borrar tanto elementos como contenidos. Al no apoyarse en un lenguaje de programación en particular, DOM facilita el diseño de páginas web activas, proporcionando una interfaz estándar para que otro software manipule los documentos.

[4]Los Spiders o arañas: Son robots automáticos que utilizan la mayoría de los grandes buscadores internacionales y conocidos para indexar los sitios en sus motores de búsqueda.

- CCS: Hojas de Estilo en Cascada (*Cascading Style Sheets*). Es un mecanismo para dar estilo a documentos HTML y XML, que consiste en reglas simples a través de las cuales se establece cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores control total sobre el estilo y formato de sus documentos separando contenido y presentación.
- XSLT: Transformaciones del Lenguaje de Hojas de Estilo Extensible (*Extensible Stylesheet Language Transformations*). Es un lenguaje que permite la transformación de la estructura de un documento XML en otro documento XML con estructura diferente.

Indiscutiblemente con el uso de estos estándares se garantiza una mejor calidad, accesibilidad y rapidez en las aplicaciones web desarrolladas, así como mejor interoperabilidad y usabilidad.

1.5 Herramientas y tecnologías utilizadas.

1.5.1 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, bibliotecas y un lenguaje de *scripting*[5] entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. (Maldonado, 2007)

Frameworks de JavaScript:

Para este trabajo se realizó en estudio sobre los frameworks más usados de JavaScript en el mundo de la web 2.0, ya que estos hoy en día se han popularizado en el desarrollo de aplicaciones web. Entre los principales frameworks en la actualidad para la construcción de aplicaciones web con JavaScript se encuentran: (AjaxLine, 2010)

JQuery, este framework permite la interactividad y modificación del árbol DOM, además de la manipulación de la hoja de estilos CSS. Brinda una gama amplia de efectos y animaciones

[5]Un lenguaje de scripting o lenguaje de script: Es un lenguaje de programación que soporta la escritura de guiones

personalizadas. Posee utilidades como por ejemplo; obtener información del navegador, operar con objetos y vectores. También es compatible con los navegadores Mozilla Firefox, Internet Explorer, Safari Opera 10.6 y Google Chrome 8 y versiones superiores a las descritas en estos navegadores. Aunque JQuery por sus características, es un framework potente para el desarrollo de aplicaciones web, posee desventajas que llevaron a que se desechara en este trabajo, algunas de las más importantes son; es orientado al DOM en lugar de a JavaScript, y presenta problemas en las funciones de *callback*[6]. (AjaxLine, 2010)

MooTools es un framework que le aporta al usuario muchas ventajas como otras tantas bibliotecas de JavaScript, entre estas se encuentra que es un framework modular y extensible, el desarrollador puede elegir qué componentes usar y cuáles no. Es orientado a objetos y sigue los principios DRY (*Don't Repeat Yourself*, No te repitas) por sus siglas en inglés. Posee componentes avanzados en efectos, con transiciones de función parabólica, optimizados y utilizados por la comunidad de desarrolladores Flash.

Este framework posee interesantes ventajas como las que se reflejaron anteriormente que lo hacen un framework importante y que se debe tenerse en cuenta a la hora de querer desarrollar una aplicación web, pero también posee desventajas que permiten desechado si aún tenemos la posibilidad de contar con otros. Una de las desventajas más importantes de este framework es que carece de widgets avanzados para la realización de una buena interfaz gráfica. (AjaxLine, 2010)

ExtJS, tiene la ventaja de que su código es reutilizable, es orientado a la programación de interfaces tipo desktop en la web. Otra de sus ventajas es que la API (*Application Programming Interface*, Interfaz de aplicación programable), es homogenizada e independiente del adaptador usado. También este framework posee una extensa comunidad de usuarios. (Vassallo, 2010)

Pero a pesar de que ExtJS tiene una gran popularidad para desarrollar aplicaciones web, necesita de una plataforma, pues se depende del paquete ExtJS para obtener los resultados deseados. Además posee una licencia de código abierto compatible con la GNU GPL v3[7] la cual exige que se comparta el código fuente de la aplicación desarrollada bajo esta licencia a los usuarios para que puedan ser libres de modificar la aplicación para sus propias necesidades. Lo que conlleva, a que si no se desea compartir el código de la aplicación, se está obligado a comprar una licencia comercial. (Vassallo, 2010)

[6] Callback: Secuencia de llamadas a funciones que se ejecutarán una detrás de otra.
[7] GNU GPL v3: "Licencia General Pública GNU versión 3, es una licencia libre, copyleft para software y otros tipos de trabajos."

Dojo, posee una total independencia del intérprete, unifica estándares de codificación y brinda utilidades de accesibilidad, incluyendo navegación por teclas y soporte para usuarios con visibilidad limitada. También provee una completa librería de componentes UI[8], pero requiere el uso de Entornos de Desarrollo Integrado (IDE) avanzados para el desarrollo del contenido web. (AjaxLine, 2010)

GWT (*Google Web Toolkit*), permite escribir aplicaciones AJAX en Java y compila a código JavaScript optimizado para los principales navegadores, posee componentes de interfaz de usuario dinámicos y re-utilizables. (Cristian Castiblanco, 2007)
Permite crear y colocar los widgets automáticamente en paneles además de empaquetar los widgets en archivos (**.jar**) para ser reutilizados.

Entre las características de este framework se encuentran:

- RPC (*Remote Procedure Call*, Llamada a Procedimiento Remoto) realmente fácil: Permite la comunicación desde el navegador que lanza la aplicación con el servidor web, solamente se necesitará definir clases de Java para las peticiones y respuestas. En producción, GWT serializa automáticamente las peticiones del navegador y de-serializa las respuestas desde el servidor web. El mecanismo de RPC de GWT puede incluso manejar jerarquía de polimorfismo en clases, y manejar las posibles excepciones. (Cristian Castiblanco, 2007)
- Depuración en tiempo real: Mientras se desarrolla la aplicación el código se compila sobre una máquina virtual de Java (JVM), lo que significa que en la fase de desarrollo permite depurar la aplicación con los avanzados sistemas de *debugging*[9] y manipulación de excepciones incluidos en los entornos de desarrollo integrado como Eclipse. (Cristian Castiblanco, 2007)
- Compatibilidad con los navegadores: Las aplicaciones en GWT serán automáticamente soportadas por navegadores como Firefox, Internet Explorer, Mozilla, Safari, y Opera sin ningún tipo de operación para la detección de los mismos, en la mayoría de los casos, ya que GWT compila el código Java en archivos JavaScript independientes para cada uno de los navegadores que estarán disponibles a través de cualquier servidor web. (Cristian Castiblanco, 2007)

[8] Componente UI: Un objeto mantenido en el servidor que provee funcionalidades para interactuar con el usuario final.
[9] Debugging: Proceso metódico de la búsqueda y reducción en el número de errores o defectos, en un programa de computadora.

- Integración con JUnit: Mediante la integración de JUnit en GWT se podrán probar las aplicaciones y depurarlas en un navegador mientras se construyen, incluso, se pueden probar llamadas asíncronas a procedimientos remotos (RPC). (Cristian Castiblanco, 2007)
- Internacionalización: Permite crear aplicaciones y bibliotecas de internacionalización rápida y fácilmente.
- GWT es un proyecto de código abierto: El código de GWT está disponible bajo la licencia Apache 2.0.

Las características mencionadas anteriormente convierten a GWT en un framework potente para el desarrollo de aplicaciones en la web, pero además de las características podemos mencionar algunas ventajas que lo confirman:

Permite a los desarrolladores crear aplicaciones JavaScript con interfaces complejas a partir de código Java. También brinda facilidad a la hora de depurar el correcto funcionamiento del código. Es más fácil depurar en Java que en JavaScript. El compilador de GWT se encarga de optimizar las clases Java para crear el mínimo de JavaScript necesario para depurar las aplicaciones, también crea código JavaScript optimizado para los distintos navegadores. Esto significa que no se necesita escribir distinto código JavaScript en función del navegador web utilizado. Otra importante ventaja de desarrollar con GWT es que da la posibilidad de usar cualquiera de los IDE para Java. (Izabel, 2007)

Sin duda cada uno de estos frameworks tienen peculiaridades diferentes con respecto a los otros, y todos tienen ventajas y desventajas para el desarrollo de software. Es por esta razón que después de haberse realizado un estudio detallado de las características de cada uno de ellos, se escogió GWT para el desarrollo de la capa de presentación de la aplicación, teniendo en cuenta que este es un framework desarrollado en Java de código abierto, que permite escapar de la “matriz” de tecnologías usadas actualmente para escribir aplicaciones AJAX, las cuales son difíciles de manejar y propensas a errores, atendiendo a la experiencia del equipo de desarrollo con la tecnología Java.

1.5.2 Entorno de desarrollo Integrado de Programación (IDE).

Un Entorno de Desarrollo Integrado, *Integrated Development Environment* (IDE) por sus siglas en inglés, es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea,

consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante *plugins*[10] se le puede añadir soporte de lenguajes adicionales. A continuación se describirán algunos de los IDE más populares para el desarrollo de aplicaciones: (EcuRed, 2011)

Eclipse: Entorno de desarrollo integrado de código abierto multiplataforma para desarrollar proyectos. Esta plataforma ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java, llamado Java Development Toolkit (JDT) y el Eclipse Compiler for Java (ECJ), que se entrega como parte de Eclipse. También se puede usar para otros tipos de aplicaciones cliente. En Eclipse se pueden usar diferentes lenguajes de programación como: Java, ANCI C, C++, JSP, Perl, PHP.

NetBeans: Brinda la posibilidad de programar en distintos lenguajes, es ideal para trabajar con el lenguaje de desarrollo Java y todos sus derivados, además ofrece un excelente entorno para programar en PHP. También se puede descargar una vez instalado NetBeans, los complementos para programar en C++. NetBeans tiene un excelente balance entre una interfaz con múltiples opciones y un aceptable completamiento de código.

Geany: Este entorno es muy sencillo, pero proporciona las funcionalidades necesarias para desarrollar aplicaciones sin problemas. Su interfaz está dividida en tres zonas: panel lateral con el árbol de carpetas y documentos abiertos, sección principal para el código y panel inferior para los mensajes de la aplicación, compilación, etc. Este IDE permite programar en diferentes lenguajes como: C, C++, Java, Python, Pascal, SQL o HTML.

CodeRun: Es un IDE que permite programar en línea varios lenguajes, entre ellos PHP, Ajax, C#, CSS, JavaScript y HTML. Funciona perfectamente, aunque está en inglés, es útil para quien no disponga de un buen editor a mano.

Spring Source Tool Suite (STS): Es un IDE basado en la versión Java EE de Eclipse, pero personalizado de manera especial para trabajar con el Framework Spring. Con su consola Spring Insight proporciona una interfaz gráfica en tiempo real de las métricas de rendimiento en las aplicaciones, que permite a los desarrolladores identificar y diagnosticar problemas desde sus escritorios. Unido a esto STS apoya las

[10] Plugin: módulo de hardware o software que añade funcionalidades a un sistema más grande.

aplicaciones dirigidas a los servidores locales, virtuales y basadas en nube. Es de libre acceso para el desarrollo y uso interno de las operaciones de negocios sin límites de tiempo.

Además, STS proporciona herramientas para el trabajo con varias tecnologías enfocadas al desarrollo sobre la plataforma JavaEE como **Grails** que en este caso se utilizará para el negocio de la aplicación y el lenguaje dinámico Groovy que es un lenguaje de programación con una sintaxis de Java, desarrollado desde y para Java, que se compila en *bytecode*[11] igual que Java y se ejecuta en la Máquina Virtual de Java (JVM).

También, **Groovy** se integra perfectamente con Java y permite la mezcla entre ambos; al punto de ser posible instanciar objetos Java desde Groovy y viceversa, es mucho más flexible y potente, además de ser totalmente compatible con sus librerías. STS tiene incorporado por defecto el plugin para trabajar con el framework de Grails. Posee una consola de comandos donde se pueden ejecutar comandos Grails y permite ejecutar y depurar aplicaciones Grails.

Cuando la aplicación es ejecutada el STS brinda una URL **(Uniform Resource Locator)** a la cual se accede para ver el resultado de la aplicación. Además entre las más importantes ventajas que tiene este IDE es que permite la incorporación del plugin de Google para Eclipse (GPE). Este plugin brinda a los desarrolladores la posibilidad de crear aplicaciones con calidad de manera rápida y fiable, además de permitir el completamiento de código y la realización de pruebas.

Después de conocer algunas de las características de las herramientas analizadas anteriormente se decidió utilizar la herramienta Spring Source Tool Suite versión 2.5 para darle solución al problema planteado ya que tiene un soporte aceptable para todas las tecnologías que se utilizarán para la implementación de las interfaces de usuario.

1.6 Metodologías de Desarrollo de Software.

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de

[11] Bytecode: Es el código independiente del hardware generado por el compilador Java™ y ejecutado por su intérprete.(JVM)

metodologías para la creación de software. Las cuales se podrían clasificar en dos grandes grupos: (Isaías Carrillo Pérez, 2008)

- Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas **Metodologías Pesadas.**
- Las metodologías orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas, **Metodologías ágiles.**

1.6.1 Proceso Unificado de Desarrollo (Rational Unified Process, RUP).

RUP es una metodología de desarrollo preparada para proyectos grandes y complejos, se encuentra en el grupo de las metodologías pesadas. Utiliza como lenguaje de modelado UML, es orientada a objetos y genera una gran cantidad de documentación, elemento importante para los equipos de desarrollo cuyo personal varía constantemente y en proyectos con una gran duración. RUP es una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cómo y cuándo). Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales, respetando cronograma y presupuesto. El proceso de desarrollo de RUP tiene tres características esenciales: está dirigido por casos de uso, centrado en la arquitectura, y es iterativo e incremental. (Sánchez, 2012)

La metodología RUP divide en 4 fases el desarrollo del software:

- **Inicio**: El objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: En esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción**: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- **Transmisión:** El objetivo es llegar a obtener el *release*[12] del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

[12] El release de un software: Es la distribución del mismo, su documentación y materiales de soporte.

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

Disciplina de Desarrollo

- **Ingeniería de Negocios:** Entendiendo las necesidades del negocio.
- **Requerimientos:** Trasladando las necesidades del negocio a un sistema automatizado.
- **Análisis y Diseño:** Trasladando los requerimientos dentro de la arquitectura de software.
- **Implementación:** Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- **Pruebas:** Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

Disciplina de Soporte

- **Configuración y administración del cambio:** Guardando todas las versiones del proyecto.
- **Administrando el proyecto:** Administrando horarios y recursos.
- **Ambiente:** Administrando el ambiente de desarrollo.
- **Distribución:** Hacer todo lo necesario para la salida del proyecto.

1.6.2 Programación Extrema (Extreme Programming, XP).

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. (Sánchez, 2012)

Esta metodología se basa en:

Pruebas Unitarias: Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueden hacer pruebas de las fallas que pudieran ocurrir. Es como si se pudieran obtener los posibles errores antes de que ocurran.

Refabricación: Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Después de haber realizado un breve estudio sobre las características que nos brindan estas metodologías para el desarrollo del software, se decidió escoger RUP por ser una metodología que entre sus principales características brinda una gran cantidad de documentación lo que permite que el software esté bien documentado, lo cual como bien se explica en las características es un elemento importante para aquel equipo de desarrollo cuyo personal es muy cambiante, como es el caso del proyecto SIC. Elemento importante pues en este caso el cliente no forma parte del equipo de desarrollo e interesa por tanto generar la mayor documentación posible.

1.7 Herramientas Case.

Se puede definir a las Herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadoras) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida del desarrollo de un software. (Murillo, 1999)

1.7.1 Visual Paradigm.

Visual Paradigm para UML es una herramienta CASE multiplataforma que da soporte al modelado visual con UML 2.0. Además, es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (Sierra, 2007)

1.7.2 Rational Rose.

Rational Rose es la herramienta CASE que soporta de forma completa la especificación de UML. Propone la utilización de cuatro tipos de modelados para realizar un diseño del sistema utilizando una vista estática y otra dinámica de los modelos del sistema: uno lógico y otro físico. Cubre todo el ciclo de vida del

proyecto y está disponible para la plataforma Windows. Esta herramienta permite la generación de código a partir de un diseño en UML para lenguajes como: C++, Java, Visual Basic, Ada, CORBA. Además, permite realizar ingeniería inversa por lo que se puede obtener un diseño a partir del código fuente de un programa. (Rouse, 2005)

Una vez analizadas las características de las herramientas CASE se escogió Visual Paradigm como herramienta para el modelado, por ser esta una herramienta multiplataforma.

1.8 Conclusiones parciales.

En este capítulo se realizó un estudio de los conceptos necesarios para el desarrollo de la aplicación. Teniendo en cuenta las características de las metodologías de desarrollo, herramientas y tecnologías descritas, y que se adecuan correctamente al tipo de desarrollo a realizar, se escoge como plataforma de desarrollo Java y el framework Google Web Toolkit para el desarrollo de la capa de presentación. Se escogió STS versión 2.5 como Entorno de Desarrollo Integrado y como metodología de desarrollo RUP. Además se escogió Visual Paradigm como herramienta de modelado.

Capítulo 2: Descripción y Diseño del sistema.

2.1 Introducción.

En este capítulo se realiza una breve descripción del sistema. Se presentan los requisitos funcionales y no funcionales, se definen los actores y casos de uso, y se hace un resumen breve de estos. Se describe la arquitectura base del sistema a partir de las tecnologías seleccionadas para su construcción, además de hacer un minucioso estudio sobre algunos de los patrones de arquitectura utilizados para la creación de aplicaciones web.

2.2 Descripción del sistema a desarrollar.

El sistema a desarrollar tiene como objetivo fundamental proporcionar una interfaz amigable para la gestión de los servicios del proyecto SIC. Debe ser capaz de gestionar la información de los servicios que ofrece este proyecto, así como de los contenidos y su información complementaria asociada. El sistema debe permitir el acceso a los usuarios desde cualquier sitio físico, garantizando que los mismos puedan gestionar la información disponible en el sistema desde cualquier área o local donde se encuentren.

2.2.1 Requerimientos del Sistema.

Un requerimiento puede definirse como un atributo necesario dentro de un sistema, que puede representar una capacidad, una característica o un factor de calidad del sistema de tal manera que le sea útil a los clientes o a los usuarios finales. (Gallego, 2006)

A continuación se presentan los requerimientos funcionales y no funcionales capturados para el desarrollo del sistema.

Requisitos funcionales

RF1. Crear ubicaciones técnicas

RF2. Buscar ubicaciones técnicas

RF3. Crear equipos técnicos

RF4. Buscar equipos técnicos

RF5. Aplicar metodología Acia

RF6. Gestionar Plan de mantenimiento

6.1 Insertar plan de mantenimiento

6.2 Modificar plan de mantenimiento

6.3 Buscar plan de mantenimiento

6.4 Visualizar plan de mantenimiento

6.5 Eliminar plan de mantenimiento

Requisitos no funcionales:

Usabilidad

RNF1. La interfaz de usuario debe ser de fácil operación y basada en áreas de trabajo.

RNF2. El sistema deberá ser utilizado por usuarios que tengan conocimiento acerca del funcionamiento y procesamiento de la información con que se trabaja en el sistema.

RNF3. El sistema estará desarrollado en una plataforma web la cual permitirá al usuario poder tener acceso a la aplicación desde cualquier navegador con la facilidad de no tener que instalar la aplicación en cada puesto de trabajo.

Eficiencia

RNF4. El sistema debe estar en capacidad de dar respuesta al acceso de todos los usuarios definidos en el sistema con tiempo de respuesta aceptable y uniforme, en la medida de las posibilidades tecnológicas, en períodos de alta, media y baja demanda del uso del sistema.

Soporte

Restricciones de diseño e implementación:

RNF5. El sistema debe ser basado en la arquitectura Cliente-Servidor.

RNF6. El diseño de la aplicación, se hará aplicando el paradigma de programación orientada a objetos.

RNF7. Se utilizarán las tecnologías que brindan los frameworks definidos, así como sus arquitecturas correspondientes.

RNF8. El sistema será implementado en el lenguaje Java. Se debe utilizar como IDE de desarrollo Spring Source Tool Suite (STS).

RNF9. El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

Interfaz

RNF10. El sistema debe tener interfaces gráficas amigables.

El sistema debe tener una interfaz intuitiva, organizada y de fácil entendimiento para el usuario que combine correctamente los colores, tipo de letra y tamaño y que los íconos estén en correspondencia con lo que representan.

Hardware

RNF11. Tener un mínimo PC Pentium 4, 2.4 GHz, 512 MB de memoria RAM.

Software

RNF12. El sistema debe ser multiplataforma, especialmente para Windows XP y Linux.

RNF13. El sistema debe visualizarse correctamente en Internet Explorer, Mozilla, Opera y Chrome.

Requisitos de Licencia

RNF14. Los requisitos legales, de derecho de autor y otros se establecen a través del Centro de Informática Industrial (CEDIN).

2.2.2 Actores del Sistema.

Tabla 1 Actores del Sistema

Actores	Descripción
Ingeniero de Confiabilidad	Crea, modifica y elimina planes de mantenimiento de manera básica y avanzada dentro del Repositorio de Información Unificada (RIU) según

	criterios establecidos. Aplica la Metodología de Análisis de Criticidad Integral de Activos (ACIA) a los Equipos Técnicos. Gestiona la información de los equipos y las ubicaciones técnicas dentro del RIU.

2.2.3 Casos de Uso del Sistema.

A continuación se muestra un resumen de los casos de uso definidos en la etapa del análisis para la versión de escritorio del proyecto SIC con el objetivo de lograr un entendimiento más claro sobre el dominio de aplicación de este trabajo.

Tabla 2 Casos de Uso del Sistema

CU-1	**Crear Ubicación Técnica**
Actor	Ingeniero de confiabilidad
Resumen	El actor realiza la acción de insertar ubicaciones técnicas en el RIU.
Referencia	RF1

CU-2	**Buscar Ubicaciones Técnicas**
Actor	Ingeniero de confiabilidad.
Resumen	El actor realiza la acción de buscar ubicaciones técnicas dentro del RIU
Referencia	RF2

CU-3	Crear Equipos
Actor	Ingeniero de confiabilidad
Resumen	Este caso de uso le permite al ingeniero de confiabilidad insertar equipos en el RIU.
Referencia	RF3

CU-4	Buscar Equipos
Actor	Ingeniero de confiabilidad
Resumen	Este caso de uso le permite al Ingeniero de confiabilidad buscar equipos en el RIU.
Referencia	RF4

CU-5	Aplicar Metodología Acia
Actor	Ingeniero de confiabilidad
Resumen	Este caso de uso le permite al Ingeniero de confiabilidad aplicar la Metodología de Análisis de Criticidad Integral de Activos (ACIA) a los Equipos.
Referencia	RF5

CU-6	Gestionar plan de mantenimiento
Actor	Ingeniero de confiabilidad
Resumen	Este caso de uso le permite al Ingeniero de confiabilidad crear, modificar y eliminar Planes

	Genéricos de Mantenimiento (plantillas) dentro del Repositorio de Información Unificada (RIU).
Referencia	RF6

2.2.4 Diagrama de casos de uso.

En la figura 1 se muestra el diagrama que responde a los casos de uso definidos para la aplicación. Este diagrama es obtenido de la etapa del análisis para la versión de escritorio del proyecto SIC.

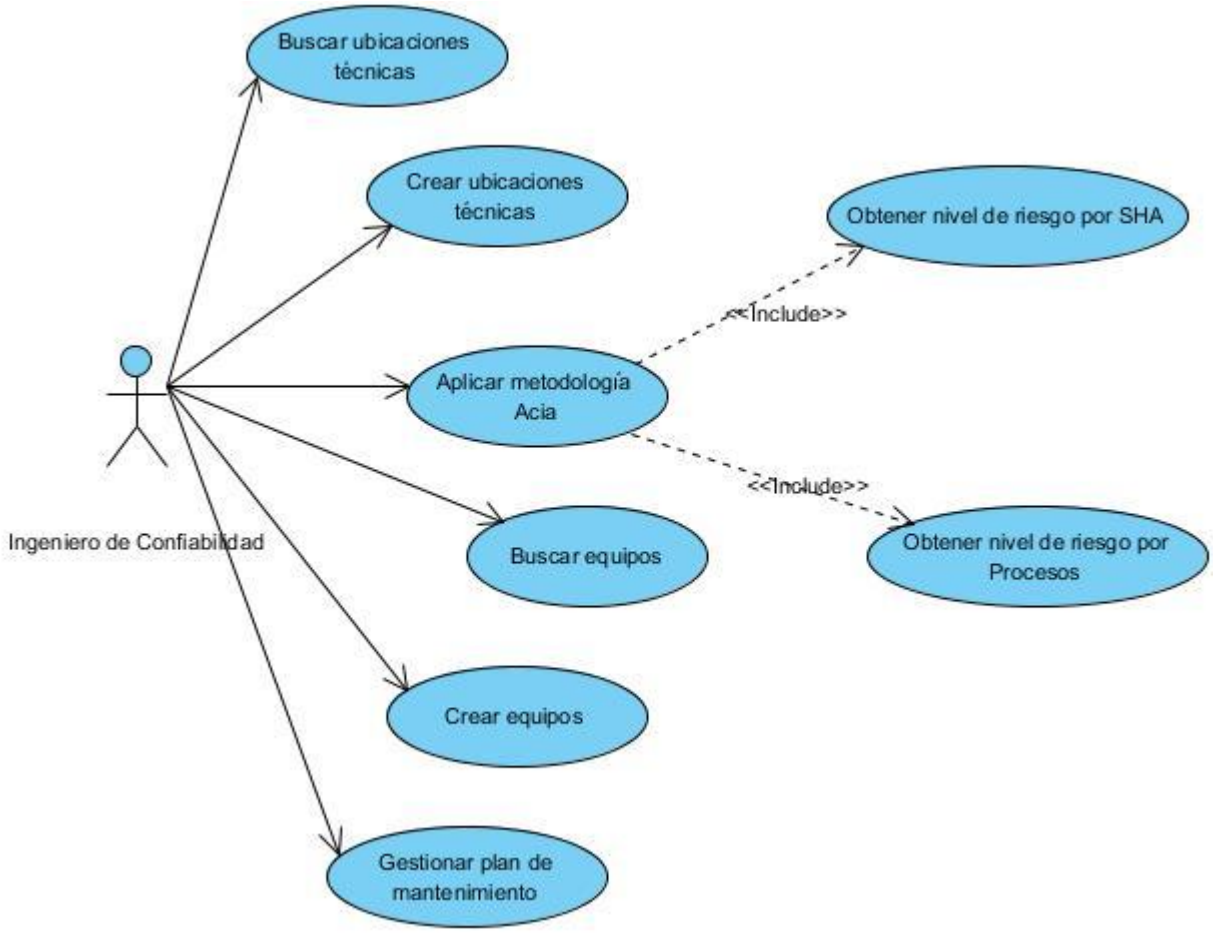


Fig. 1 DCU para el Ingeniero de confiabilidad

2.3 Descripción de la Arquitectura.

La arquitectura de software es importante como disciplina debido a que los sistemas de software crecen de forma tal que resulta muy complicado que sean diseñados, especificados y entendidos por un solo individuo. Uno de los aspectos que motivan el estudio en este campo es el factor humano, en términos de aspectos como inspecciones de diseño, comunicación a alto nivel entre los miembros del equipo de desarrollo, reutilización de componentes y comparación a alto nivel de diseños alternativos. (Erika Camacho, 2004)

Una de las definiciones más completas y por la cual se guía el diseño de la arquitectura del sistema, es la formulada por la IEEE 1471-2000: “La Arquitectura de Software es la organización fundamental de un sistema, encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución”. (IEEE, 2000)

2.3.1 Arquitectura Cliente-Servidor.

La tecnología Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales como podemos apreciar en la figura 2. Desde el punto de vista funcional, se puede definir la tecnología Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aún en entornos multiplataforma. Se trata pues, de la arquitectura más extendida en la realización de Sistemas Distribuidos. (Oposiciones TIC © 2010, 2010)

De una forma más simple también se puede decir que la arquitectura cliente-servidor se encuentra dentro de la clasificación del estilo **Llamada y retorno,** y es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. (Campo, 2008)

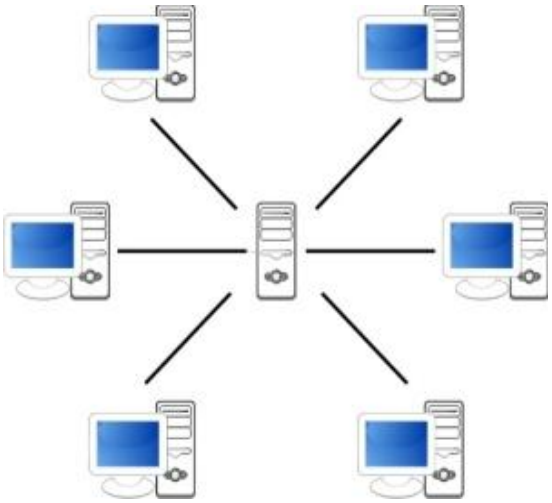


Fig. 2 Estilo arquitectónico cliente-servidor

2.3.2 Modelo – Vista – Controlador.

La arquitectura Modelo Vista Controlador, (*Model View Controller,* MVC), fue introducida como parte de la versión *Smalltalk-80* del lenguaje de programación *Smalltalk*. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que; el Modelo, las Vistas y los Controladores se tratan como entidades separadas, esto hace que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas. Este modelo de arquitectura presenta varias ventajas: posee una clara separación entre los componentes de un programa, lo cual permite implementarlos por separado. También cuenta con una API (*Application Programming Interface*) muy bien definida; cualquiera que use la API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad. La conexión entre el Modelo y sus Vistas es dinámica, se produce en tiempo de ejecución, no en tiempo de compilación. (Catalani, 2012)

El MVC es un patrón de arquitectura que considera dividir una aplicación en tres módulos claramente identificables y con funcionalidad bien definida como se puede observar en la figura 5, a continuación se hace una breve descripción de cada uno de estos módulos:

Modelo: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

Vista: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

Controlador: Recibe las entradas, usualmente como eventos que codifican los movimientos o pulsaciones de los botones del ratón y de teclas. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista.

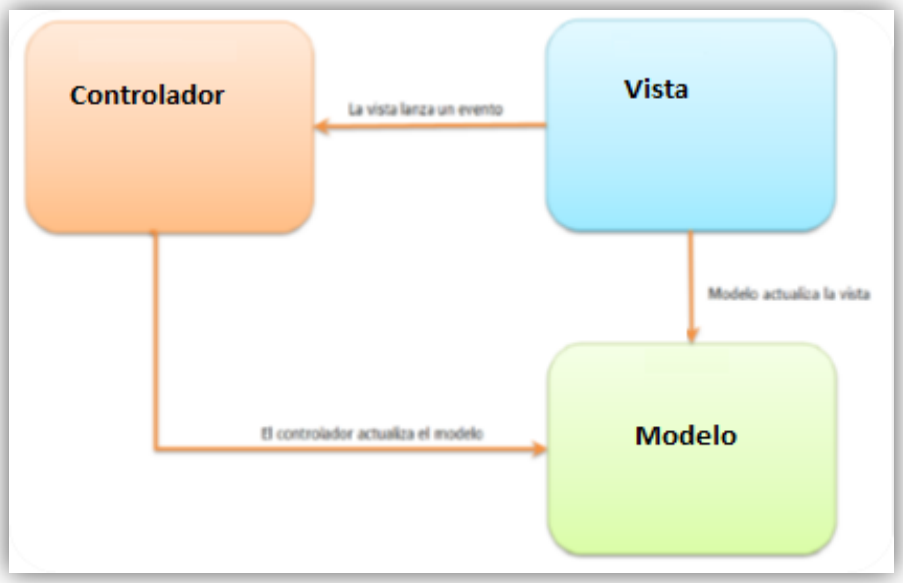


Fig. 3 Patrón MVC (Modelo Vista Controlador)

2.3.3 Modelo – Vista – Presentador.

Este patrón se considera una derivación del MVC. El Modelo-Vista-Presentador, (*Model View Presenter,* MVP), por sus siglas en inglés separa el modelo del dominio, la presentación y las acciones basadas en la interacción con el usuario en tres clases separadas como se muestra en la figura 6. La vista le delega a su presentador, toda la responsabilidad del manejo de los eventos del usuario. El presentador se encarga de actualizar el modelo cuando surge un evento en la vista, pero también es responsable de actualizar a la vista cuando el modelo le indica que ha cambiado. El modelo no conoce la existencia del presentador. Por lo tanto, si el modelo cambia por acción de algún otro componente que no sea el presentador debe disparar un evento para que el presentador se entere. A la hora de implementar este patrón, se identifican los siguientes componentes: (César de la Torre Llorente, 2010)

IView: Es la interfaz con la que el Presentador se comunica con la vista.

Vista: Vista que implementa la interfaz IView y se encarga de manejar los aspectos visuales. Mantiene una referencia a su Presentador al cual le delega la responsabilidad del manejo de los eventos.

Presentador: Contiene la lógica para responder a los eventos y manipula el estado de la vista mediante una referencia a la interfaz IView. El Presentador utiliza el modelo para saber cómo responder a los eventos. El presentador es responsable de establecer y administrar el estado de una vista.

Modelo: Está compuesto por los objetos que conocen y manejan los datos dentro de la aplicación. Por ejemplo, pueden ser las clases que conforman el modelo del negocio.

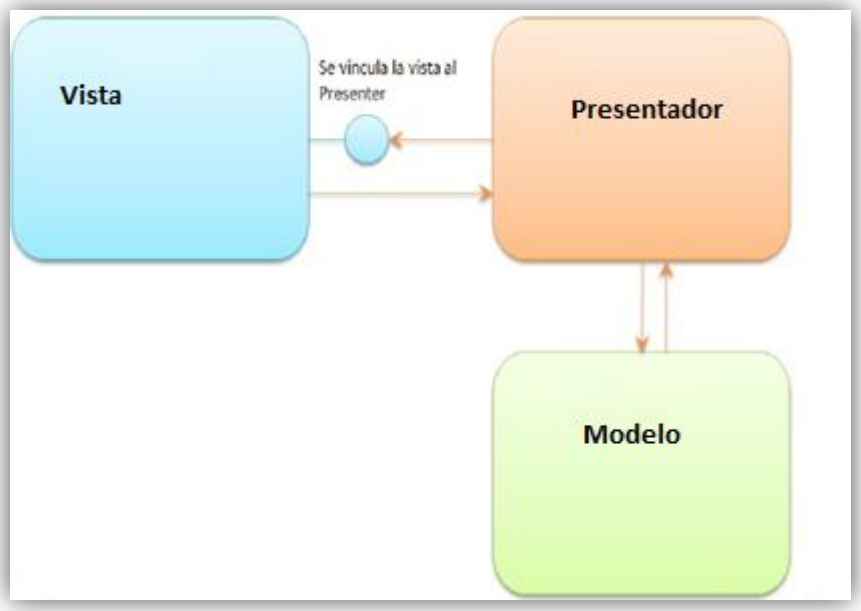


Fig. 4 Patrón MVP (Modelo Vista Presentación)

El concepto de este patrón es bastante sencillo. Por un lado está la vista, que se encarga de mostrar la información al usuario y de interactuar con él para hacer ciertas operaciones. Por otro lado, tenemos el modelo que, ignorando cómo la información es mostrada al usuario, realiza toda la lógica de las aplicaciones usando las entidades del dominio. Y por último tenemos al presentador que es el que "presenta" a ambos actores sin que haya ningún tipo de dependencia entre ellos.

2.3.4 Modelo Vista Presentador aplicado a GWT.

En cuanto a GWT se refiere, MVP parece ser el patrón más comúnmente utilizado por los desarrolladores para la creación de aplicaciones eficientes, amigables y dinámicas, en muchas plataformas, tales como *Microsoft*.*NET* o *Android* y *Windows*. Pero existen varios patrones arquitectónicos para elegir, como lo son el *Presentation Abstraction Control* (PAC), MVC, y por supuesto el MVP. Pero si bien cada modelo tiene sus ventajas, el MVP funciona mejor cuando se desarrollan aplicaciones con GWT por dos razones principales: el modelo de MVP, al igual que otros patrones de diseño, desacopla el desarrollo de una manera que permite a varios desarrolladores trabajar simultáneamente y permite minimizar el uso de los casos de pruebas para GWT (*GWTTestCase*), que se basan en la presencia de un navegador. Pero lo principal de este patrón es la separación de las funcionalidades en componentes que lógicamente tienen sentido, en este caso GWT tiene un enfoque claro; que es hacer la vista tan simple como sea posible para minimizar la dependencia de GWTTestCase y reducir el tiempo en las pruebas de funcionamiento. (Ramsdale, 2010)

El MVP en GWT contiene los mismos componentes originarios, es decir, el modelo; que incluye los objetos del negocio. La vista, que contiene todos los componentes de interfaz de usuario que componen la aplicación, y es bueno aclarar que aquí también las vistas son responsables de la distribución de los componentes de interfaz de usuario y no tienen noción del modelo, y por último el presentador, que es el que contiene toda la lógica de la aplicación, la sincronización y la transición de los datos a través de RPC con el servidor. Lo que en este caso se suma otro componente más, que cumple con una importante función dentro de GWT y es el AppController, este componente contiene la gestión de la historia y la lógica de la transición de la vista. (Ramsdale, 2010)

2.3.5 Diseño de la Arquitectura del sistema.

El sistema es una aplicación web; proponiendo así, que los usuarios puedan acceder a través de un navegador al mismo, por lo que el sistema será basado en el estilo arquitectónico Cliente-Servidor y este a su vez seguirá los principios del patrón MVC, el cual tiene como idea principal la Presentación Separada, que consiste en hacer una división clara entre objetos del dominio y la presentación. Los objetos del dominio deben ser completamente auto contenidos y trabajar sin referirse a la presentación, también

deben ser capaces de soportar presentaciones múltiples que pudieran llegar a ser simultáneas. (César de la Torre Llorente, 2010)

Este patrón arquitectónico establece que los componentes de un sistema de software deben organizarse en tres capas distintas según su misión:

Modelo, o capa de datos: Contiene los componentes que representan y gestionan los datos manejados por la aplicación.

Vista, o capa de presentación: Los componentes de esa capa son responsables de mostrar al usuario el estado actual del modelo de datos, y presentarle las distintas acciones disponibles.

Capa de control: Contendrá los componentes que reciben las órdenes del usuario, gestionan la aplicación de la lógica de negocio sobre el modelo de datos, y determinan que vista deben mostrarse a continuación.

2.4 Modelo de diseño.

2.4.1 Modelo de diseño del sistema.

La aplicación cuenta con 3 paquetes fundamentales como quedan descritos a continuación y se puede observar en la figura 7:

Controlador: Este paquete cuenta con un subsistema que agrupan las clases encargadas de la selección de los eventos para cada Caso de Uso, darle interpretación a estos eventos y darle a las otras dos partes que conforman el sistema los mensajes de la acción a realizar, previamente procesada por la clase correspondiente dentro de este paquete. Estas acciones a realizar van desde informar a las interfaces que deben mostrar y en qué momento hasta informarle a las partes del modelo las acciones a realizar.

Vista: Este paquete cuenta con la agrupación de tres subsistemas que responden a la aplicación del patrón MVP en la capa de presentación. Estos subsistemas van a agrupar las clases que se utilizan para mostrar la información que desea ver el usuario y los recursos que necesitan para la actualización de las vistas. Estos tres subsistemas que se agrupan en este paquete son los encargados de manejar la interacción del usuario con la aplicación.

Modelo: Este paquete cuenta con un subsistema que es el encargado de representar detalladamente la información con la que el sistema debe operar.

2.4.1.1 Diagrama de Paquetes.

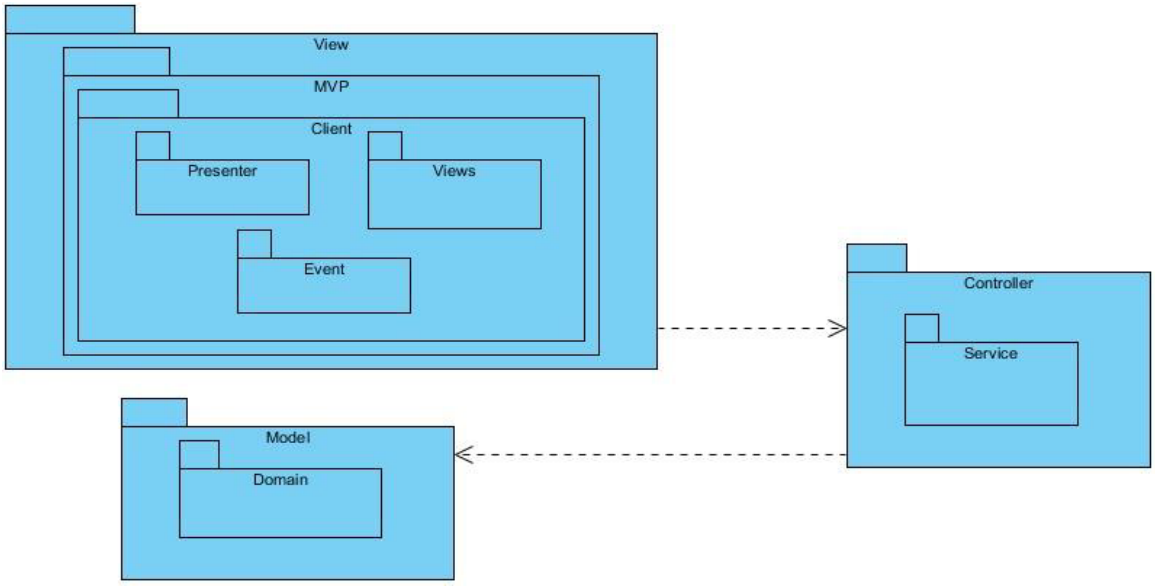


Fig. 5 Diagrama de Paquetes del Sistema

Existe una relación lógica entre los diferentes subsistemas que conforman la aplicación la cual es mostrada a continuación:

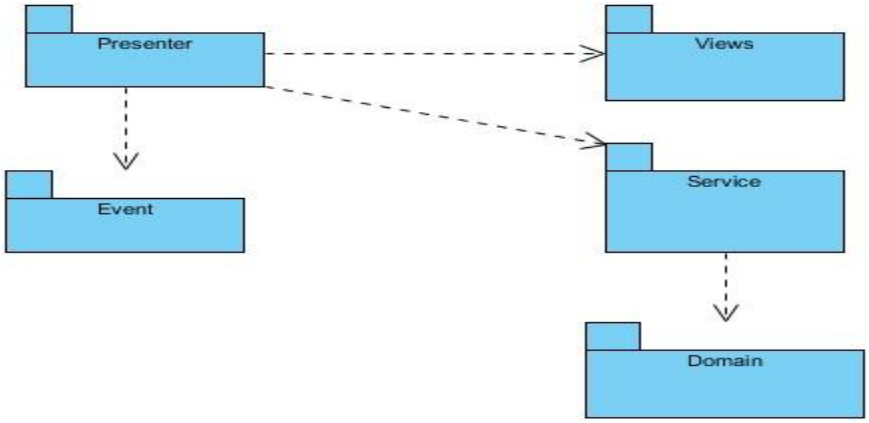


Fig. 6 Relación entre los subsistemas de diseño.

Service: Contiene un grupo de clases que garantizan para cada uno de los Casos de Uso el tratamiento y selección de los eventos, interpretan las acciones del usuario, accediendo a las operaciones de negocio de la aplicación y modificando a partir de sus resultados el estado del modelo y la navegación entre vistas.

Views: Contiene la agrupación de las clases encargadas de la visualización de los datos así como brindar las interfaces de los formularios con las cuales interactuarán los clientes de la aplicación.

Domain: El subsistema agrupa dentro del, todas las entidades correspondientes a los casos de uso con que cuenta la aplicación.

Presenter: Contiene la agrupación de las clases encargadas de actualizar las vistas y gestionar los eventos de las mismas.

Event: Contiene la agrupación de las clases encargadas de registrar los eventos de cada una de las vistas.

2.4.2 Diagrama de clases del diseño.

Un diagrama de Clases del Diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Una clase es una descripción de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. (Riesco, 2009)

La figura 9 muestra la interacción entre las clases que conforman los subsistemas de la aplicación como lo son; la vista, el presentador y los eventos, para los casos de uso “Crear Equipos Técnicos”, “Crear Ubicaciones Técnicas”, “Buscar Equipos Técnicos” y “Buscar Ubicaciones Técnicas”.

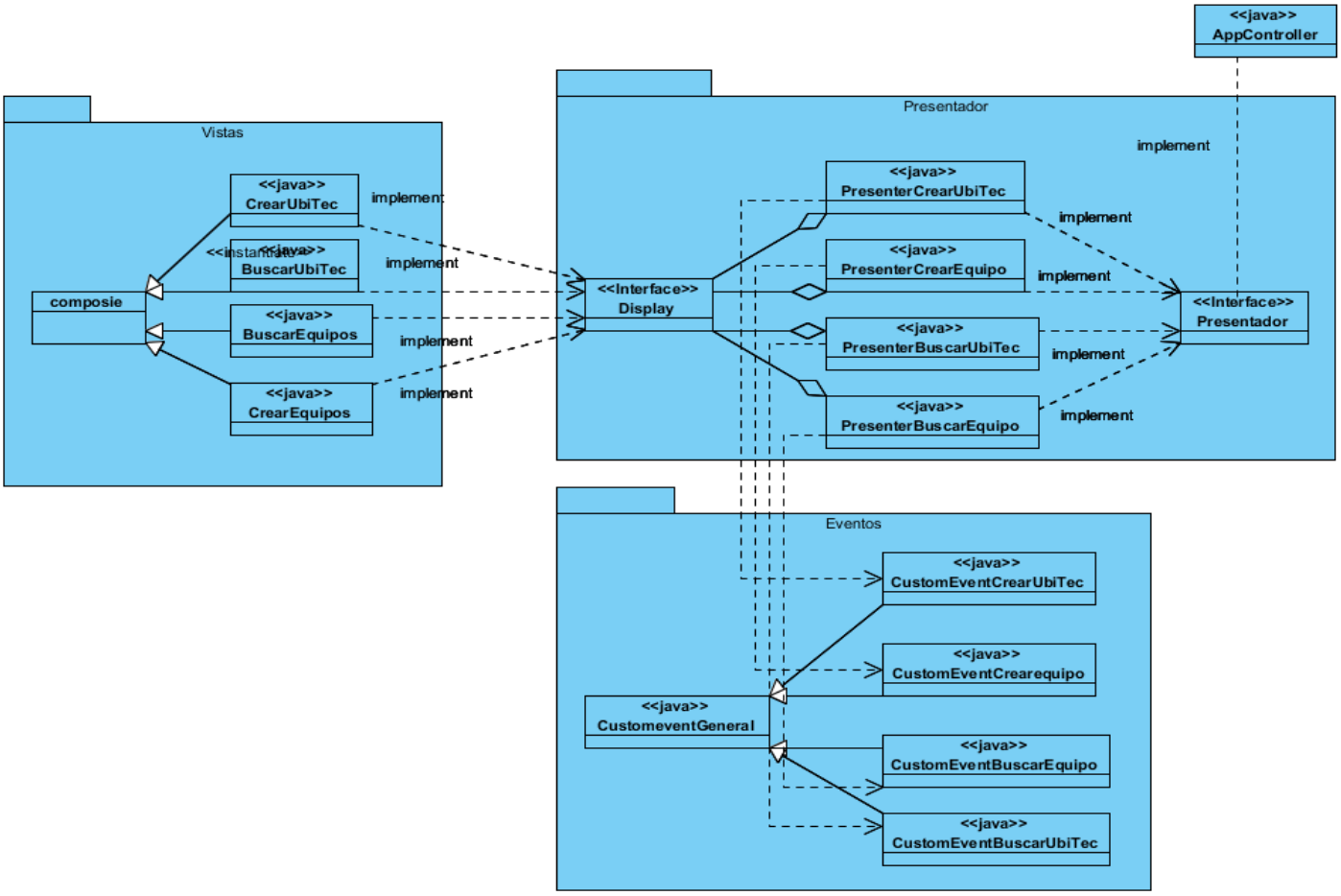


Fig. 7 Diagrama de Clases del Diseño para los casos de uso "Crear Equipos Técnicos", "Crear Ubicaciones Técnicas", "Buscar Equipos Técnicos" y "Buscar Ubicaciones Técnicas".

2.4.2.1 Descripción de las clases del diseño.

En este epígrafe se describen las principales clases del diseño que responden a los casos de uso "Crear Equipos Técnicos", "Crear Ubicaciones Técnicas", "Buscar Equipos Técnicos" y "Buscar Ubicaciones Técnicas".

En el paquete **Vistas** se encuentran las siguientes clases:

CrearUbiTec: clase que implementa la interfaz Display del presentador **PresenterCrearUbiTec** y se encarga de mostrar al usuario los componentes visuales que responden al caso de uso "Crear Ubicación Técnica".

BuscarUbiTec: clase que implementa la interfaz Display del presentador **PresenterBuscarUbiTec** y se encarga de mostrar al usuario los componentes visuales que responden al caso de uso “Buscar Ubicación Técnica”.

CrearEquipos: clase que implementa la interfaz Display del presentador **PresenterCrearEquipo** y se encarga de mostrar al usuario los componentes visuales que responden al caso de uso “Crear Equipos Técnicos”.

BuscarEquipos: clase que implementa la interfaz Display del presentador **PresenterBuscarEquipo** y se encarga de mostrar al usuario los componentes visuales que responden al caso de uso “Buscar Equipos Técnicos”.

En el paquete **Presentador** se encuentran las siguientes clases:

PresenterCrearUbiTec: clase que implementa la interfaz Presentador y que se encarga de responder a los eventos de la vista **CrearUbiTec**.

PresenterBuscarUbiTec: clase que implementa la interfaz Presentador y que se encarga de responder a los eventos de la vista **BuscarUbiTec**.

PresenterCrearEquipo: clase que implementa la interfaz Presentador y que se encarga de responder a los eventos de la vista **CrearEquipos**.

PresenterBuscarEquipo: clase que implementa la interfaz Presentador y que se encarga de responder a los eventos de la vista **BuscarEquipos**.

El paquete **Eventos** contiene las siguientes clases:

CustomEventGeneral: clase de la cual heredan las clases **CustomEventCrearUbiTec**, **CustomEventBuscarEquipo**, **CustomEventBuscarObjTec**, **CustomEventCrearEquipo**.

CustomEventCrearUbiTec: clase que se encarga de registrar todos los eventos de la vista **CrearUbiTec**.

CustomEventBuscarEquipo: clase que se encarga de registrar todos los eventos de la vista **BuscarEquipos**.

CustomEventBuscarObjTec: clase que se encarga de registrar todos los eventos de la vista **BuscarUbiTec**.

CustomEventCrearEquipo: clase que se encarga de registrar todos los eventos de la vista **CrearEquipos**.

La clase **AppController**: que se encarga de definir las acciones a desarrollar una vez lanzado un evento en el sistema.

2.5 Diagrama de componentes.

Un diagrama de componentes muestra las dependencias lógicas entre componentes de software, ya sean componentes fuentes, binarios o ejecutables, e ilustran las piezas del software. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema, es decir, para describir la vista de implementación estática de un sistema. Los diagramas de componentes se relacionan con los diagramas de clases, ya que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones, pero un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. (Ramírez., 2009)

En la figura que se muestra a continuación se representa la distribución de los componentes de la aplicación.

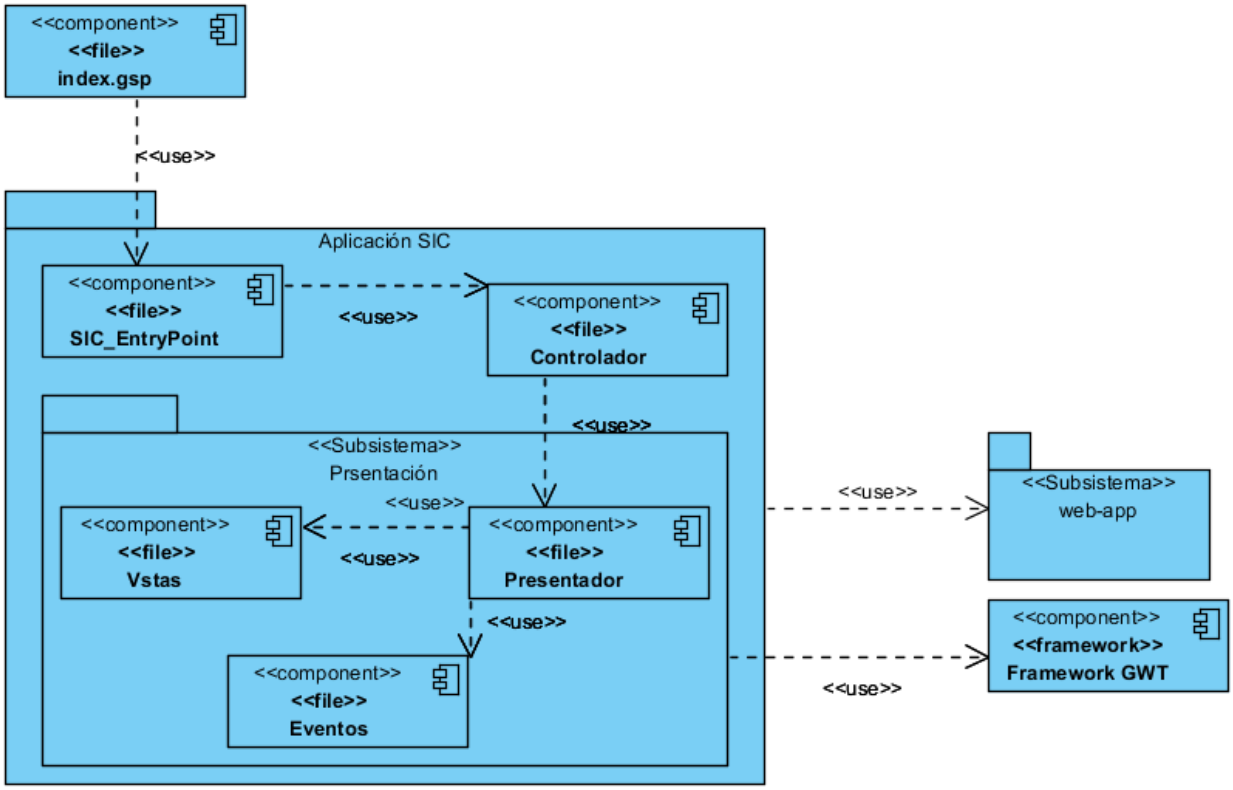


Fig. 8 Diagrama de componentes del sistema.

2.5.1 Descripción de los componentes.

Componente index.gsp: Componente mediante el cual se accede a la aplicación.

Componente SIC_EntryPoint: Este componente usa al componente **Controlador** y es en el cual se renderizan todos los componentes de la aplicación.

Componente Controlador: Este componente es el encargado de manejar todo el historial de la aplicación así como la transición lógica de todas las clases que conforman el componente **Vistas**.

Componente Presentador: Este componente usa el componente **Eventos** y es el que se encarga de responder a todos los eventos de las clases que conforman el componente **Vistas.**

Componente Vistas: Este componente es el encargado de contener todas las clases con extensión .java que se encargan de mostrarle toda la información necesaria al usuario a través de componentes visuales.

Componente Eventos: Este componente es usado por el componente **Presentador** y es el encargado de registrar todos los eventos de las vistas que conforman la aplicación cuando estas sean lanzadas a través de algún evento realizado en el sistema.

Subsistema web-app: Este subsistema agrupa a los elementos de estilos (CSS) e imágenes predeterminados que usa GWT para la construcción de los componentes visuales.

Componente framewok GWT: Este componente incluye las clases del framework GWT necesarias para la ejecución de las funciones que fueron usadas para la implementación de la capa de presentación del sistema.

2.6 Conclusiones parciales.

En este capítulo se resumieron los requisitos funcionales y las cualidades que el producto debe tener, recogidas en los requisitos no funcionales. Se definió una arquitectura base flexible y adaptada a las funcionalidades del sistema. Se estudiaron y aplicaron los patrones arquitectónicos MVC como patrón de la estructura general de la aplicación y MVP para la capa de presentación. Se modeló el diagrama de clases del diseño para los casos de uso más significativos, describiéndose además cada una de las clases que lo conforman. Se expuso el diagrama de paquetes del diseño y el diagrama de componentes del sistema, obteniendo así el conjunto de artefactos que describen la solución desarrollada.

Capítulo 3: Validación de la solución propuesta.

3.1 Introducción.

En este capítulo se diseñan los casos de prueba que permitirán evaluar y valorar la calidad del producto.

3.2 Modelo de prueba.

Las pruebas de software consisten en una serie de actividades en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, realizando una evaluación del sistema o del componente probado. Su principal objetivo es evaluar o valorar la calidad del producto. Las pruebas constituyen un elemento crítico para la garantía de la calidad del software. (Pressman, 2002)

3.2.1 Métodos de pruebas.

Para realizarles las pruebas a las interfaces se decidió utilizar el método de pruebas de caja negra, en lo adelante se describirá en que consisten estas pruebas:

Pruebas de caja negra: Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. (Pressman, 2002)

3.2.2 Diseño de casos de prueba.

Los casos de prueba que se utilizarán para las pruebas de la aplicación, están diseñados para verificar los requerimientos del usuario. A continuación se muestran cuatro de los casos de uso a los que se le realizaron las pruebas. Estos casos de uso son arquitectónicamente los más significativos del sistema porque a través de ellos se desarrollan los demás casos de uso de la aplicación.

1. CU Crear Ubicaciones técnicas.
2. CU Buscar Ubicaciones Técnicas.

3. CU Crear Equipos Técnicos
4. CU Buscar Equipos Técnicos.

Tabla 3 Prueba de caja negra del Caso de uso Crear "Ubicaciones Técnicas"

Condición de entrada	Casos válidos	Casos no válidos
Nombre de la Ubicación Técnica	**Cadena de caracteres en mayúscula separadas por guión**	**Dejar el campo del nombre vacío**
Denominación de la Ubicación Técnica	**Cadena de caracteres se incluyen números y letras**	**Dejar el campo de la denominación vacío**
Ubicación Técnica Superior	**Cadena de caracteres en mayúscula separadas por guión**	**Dejar el campo vacío**

Tabla 4 Prueba de caja negra del Caso de uso Crear "Ubicaciones Técnicas" insertando correctamente los datos

Caso de uso	Crear "Ubicaciones Técnicas"
Caso de prueba	Permitir insertar una nueva Ubicación Técnica en la base de datos introduciendo **correctamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos correctamente de la forma: **Nombre Ubicación técnica:** XX-XXX-XXXX-XXXXXX **Denominación:** Ubicación Técnica 1 **Ubicación técnica Superior:** XX-XXX-XXXX
Resultado	El sistema introduce una nueva Ubicación Técnica en la base de datos
Condiciones	El sistema muestra un cartel avisando que los datos han sido guardados

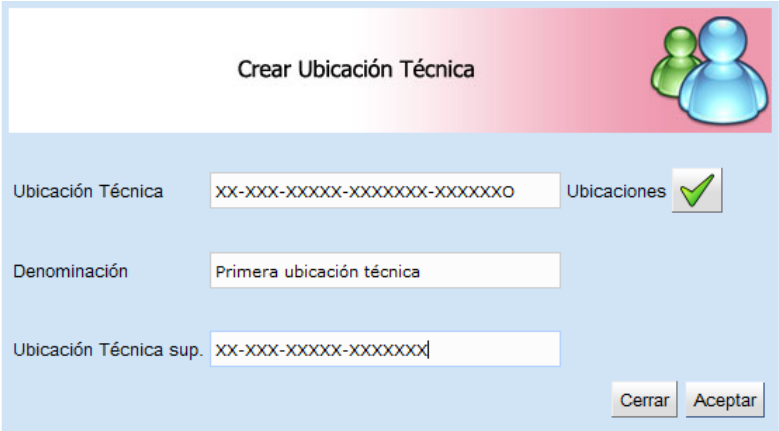


Fig. 9 Crear ubicación técnica de forma correcta

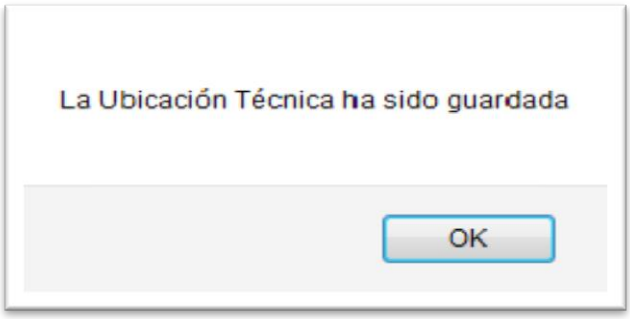


Tabla 5 Prueba de caja negra del Caso de uso Crear "Ubicaciones Técnicas" insertando incorrectamente los datos

Caso de uso	**Crear "Ubicaciones Técnicas"**
Caso de prueba	Permitir insertar una nueva Ubicación técnica en la base de datos introduciendo **incorrectamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos incorrectamente o deja algún campo vacío
Resultado	El sistema muestra un mensaje de error especificando que los datos que se han introducido son incorrectos o debe llenar todos los campos
Condiciones	Existen campos vacíos

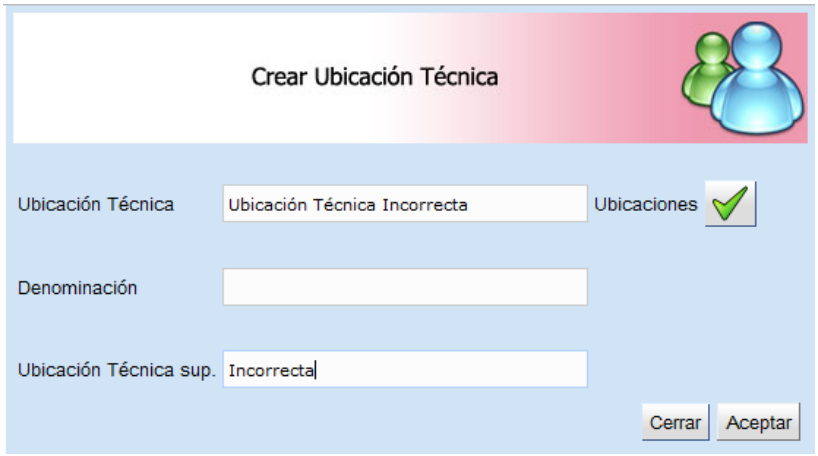


Fig. 10 Crear Ubicación técnica de forma incorrecta

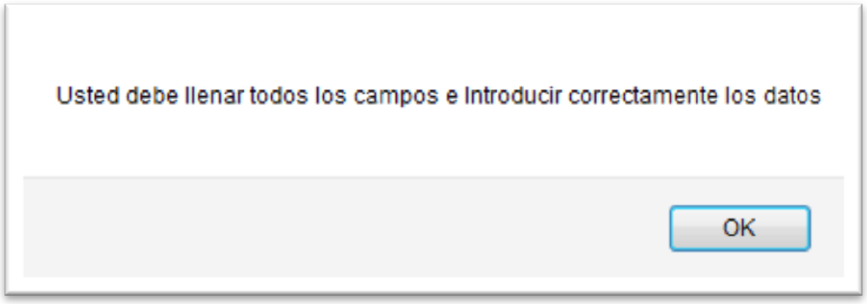


Tabla 6 Prueba de caja negra del Caso de uso Buscar "Ubicaciones Técnicas"

Condición de entrada	Casos válidos	Casos no válidos
Denominación de la Ubicación Técnica	**Cadena de caracteres se incluyen números y letras**	**Introducir la denominación de la Ubicación Técnica de forma incorrecta**

Tabla 7 Prueba de caja negra del Caso de uso Buscar "Ubicaciones Técnicas" insertando correctamente los datos

Caso de uso	Buscar "Ubicaciones Técnicas"
Caso de prueba	Permitir buscar una Ubicación Técnica en la base de datos introduciendo los datos de forma **correcta** en el campo de criterio de búsqueda

Entrada	El ingeniero de confiabilidad introduce la denominación de la Ubicación Técnica correctamente de la forma: **Denominación:** Ubicación Técnica 1
Resultado	El sistema muestra la Ubicación Técnica que corresponde con la denominación entrada por el ingeniero de confiabilidad
Condiciones	La Ubicación Técnica es mostrada de forma correcta

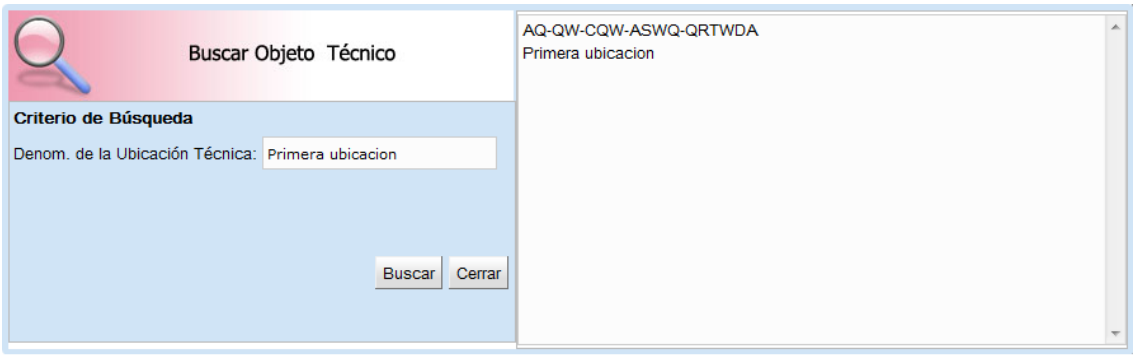


Fig. 11 Buscar ubicación técnica de forma correcta

Tabla 8 Prueba de caja negra del Caso de uso Buscar "Ubicaciones Técnicas" insertando incorrectamente los datos

Caso de uso	**Buscar "Ubicaciones Técnicas"**
Caso de prueba	Permitir buscar una Ubicación Técnica en la base de datos introduciendo los datos de forma **incorrecta** en el campo de criterio de búsqueda
Entrada	El ingeniero de confiabilidad introduce una denominación que no corresponde a ninguna de las Ubicaciones Técnicas existentes en la base de datos
Resultado	El sistema muestra un mensaje de error alertando de que no existen Ubicaciones Técnicas con dicha denominación en la base de datos
Condiciones	La denominación es incorrecta

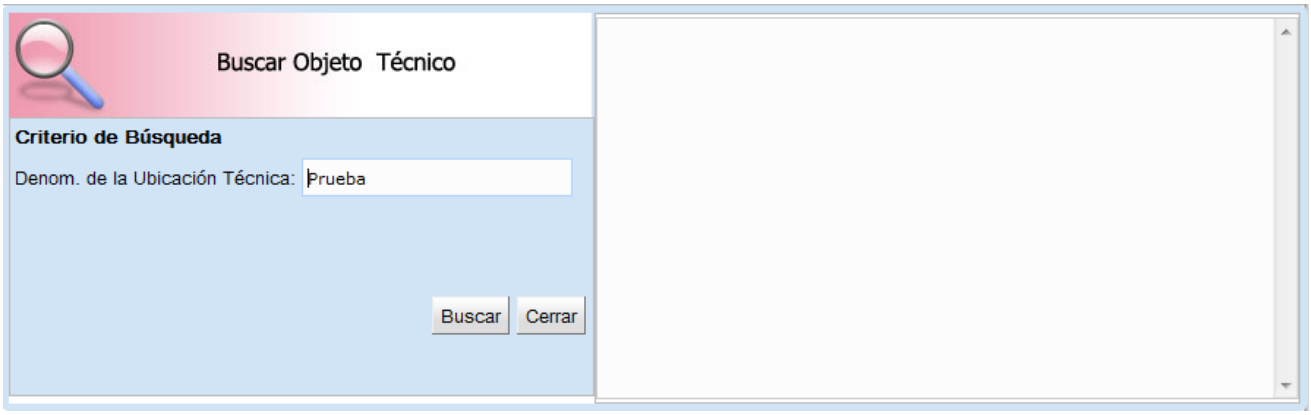


Fig. 12 Buscar ubicación técnica de forma incorrecta

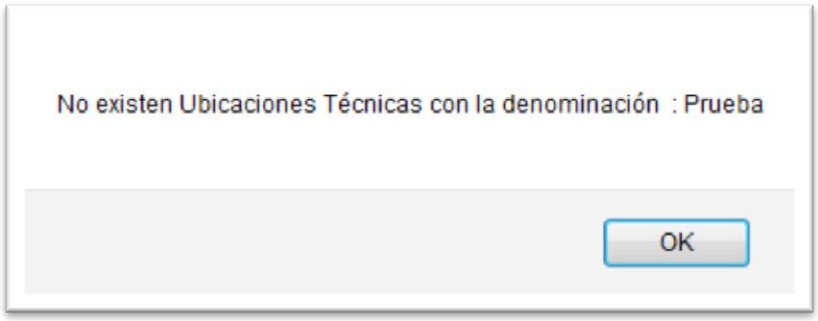


Tabla 9 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar Ubicación Técnica del Equipo"

Condición de entrada	Casos válidos	Casos no válidos
Nombre de la Ubicación Técnica	**Cadena de caracteres en mayúscula separadas por guión**	**Dejar el campo del nombre vacío**
Denominación de la Ubicación Técnica	**Cadena de caracteres se incluyen números y letras**	**Dejar el campo de la denominación vacío**
Sociedad	**Cadena de caracteres**	**Dejar el campo vacío**

	alfabéticos en mayúscula	
División	**Cadena de caracteres alfabéticos en mayúscula**	**Dejar el campo vacío**
Centro de Emplazamiento	**Cadena de caracteres alfabéticos en mayúscula**	**Dejar el campo vacío**
Indicador de Estructura	**Cadena caracteres alfanuméricos en mayúscula**	**Dejar el campo vacío**
Centro de Costo	**Cadena caracteres alfanuméricos**	**Dejar el campo vacío**
Grupo de Autorización	**Cadena de caracteres alfabéticos en mayúscula**	**Dejar el campo vacío**
Emplazamiento	**Cadena de caracteres alfabéticos en mayúscula**	**Dejar el campo vacío**
Área de Empresa	**Cadena de caracteres alfabéticos en mayúscula**	**Dejar el campo vacío**
Puesto de Trabajo	**Cadena de caracteres en mayúscula separados por guión**	**Dejar el campo vacío**
Puesto de Trabajo responsable	**Cadena de caracteres en**	**Dejar el campo vacío**

	mayúscula separados por guión	
Criticidad	**Valor numérico romano predeterminado por el sistema del I a IV**	**Dejar el campo vacío**
Grupo Planificador	**Cadena de caracteres alfabéticos en mayúscula**	**Dejar el campo vacío**

Tabla 10 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar Ubicación Técnica del Equipo" de forma correcta

Caso de uso	**Crear "Equipos Técnicos" Escenario "Insertar la ubicación técnica de los Equipos"**
Caso de prueba	Permitir insertar la ubicación técnica del Equipo en la base de datos introduciendo **correctamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos correctamente de la forma: **Nombre Ubicación técnica:** XX-XXX-XXXX-XXXXXX **Denominación:** Ubicación Técnica 1 Sociedad: RFCF División: ROCC Centro Emplazamiento: CEAZ Emplazamiento: CEAZ Indicador: FR001 Centro de Costo: 12052012 Grupo de Autorización: DRCF

	Área de la empresa PDMA Grupo Planificador: DIRF Puesto de Trabajo OFCE-CF Puesto de Trabajo responsable: OFCE-CF Criticidad: III
Resultado	El sistema introduce la ubicación técnica de ese Equipo Técnico en la base de datos
Condiciones	El sistema muestra un cartel avisando que los datos han sido guardados

Fig. 11 Insertar la ubicación técnica de un equipo de forma correcta

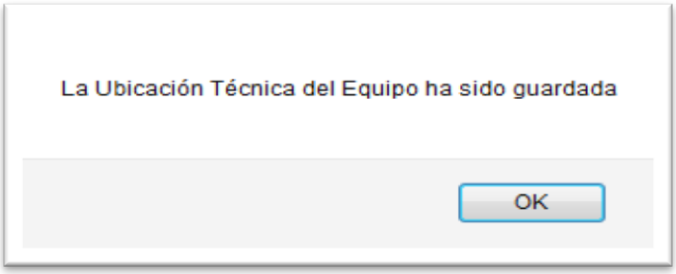


Tabla 11 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar Ubicación Técnica del Equipo" de forma incorrecta

Caso de uso	**Crear "Equipos Técnicos" Escenario "Insertar la ubicación técnica de los Equipos"**
Caso de prueba	Permitir insertar la ubicación técnica del Equipo en la base de datos introduciendo **incorrectamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos incorrectamente o deja algún campo vacío
Resultado	El sistema muestra un mensaje de error especificando que los datos que se han introducido son incorrectos o debe llenar todos los campos
Condiciones	Existen campos vacíos o datos mal introducidos

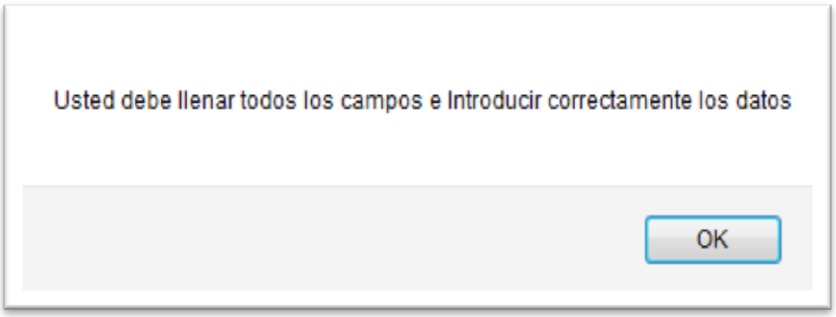


Fig. 12 Insertar datos del equipo de forma incorrecta

Tabla 12 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información general del Equipo"

Condición de entrada	**Casos válidos**	**Casos no válidos**
Número del equipo	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres**	**Dejar el campo vacío e introducir letras**
Tag del equipo	**Cadena de**	**Dejar el campo vacío**

	alfanuméricos	
Clase del equipo	**Cadena de caracteres alfanuméricos**	**Dejar el campo vacío**
Tipo del equipo	**Cadena de caracteres alfabéticos**	**Dejar el campo vacío e introducir números**
Grupo del equipo	**Cadena de caracteres alfabéticos y numéricos en mayúscula separados por guión**	**Dejar el campo vacío**
Serial del equipo	**Cadena caracteres alfanuméricos de hasta 50 caracteres**	**Dejar el campo vacío**
Fabricante	**Cadena caracteres alfanuméricos**	**Dejar el campo vacío**
Modelo	**Cadena de caracteres alfanuméricos**	**Dejar el campo vacío**

Tabla 13 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información general del Equipo" de forma correcta

Caso de uso	**Caso de uso Crear "Equipos Técnicos " Escenario "Insertar información general del Equipo"**
Caso de prueba	Permitir insertar la información general del Equipo en la base de datos introduciendo **correctamente** los datos

Entrada	El ingeniero de confiabilidad introduce los datos correctamente de la forma: Número del equipo: 00001 Tag del equipo: Equipo01 Clase del equipo: AA01 Tipo del equipo: Tipo de equipo Grupo del equipo: AA-001 Serial del equipo: ZXY0001 Fabricante: SONY Modelo: Atomic
Resultado	El sistema introduce la información general de ese Equipo en la base de datos
Condiciones	El sistema muestra un cartel avisando que el Equipo ha sido guardado

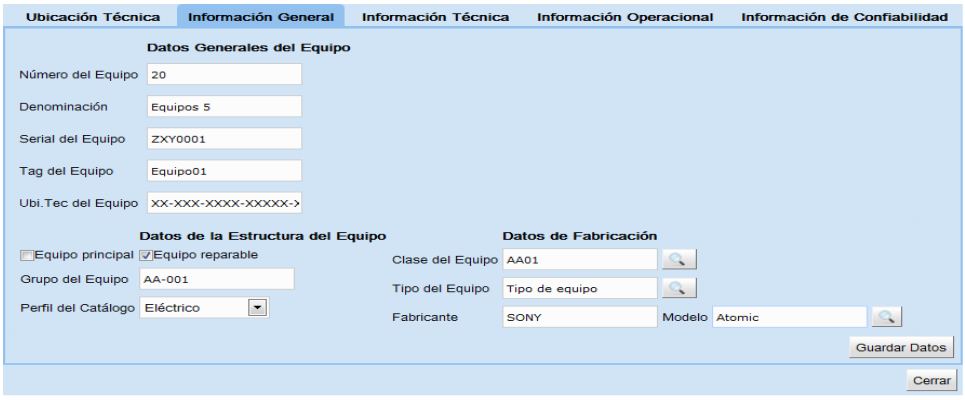


Fig. 13 Insertar información general del equipo de forma correcta

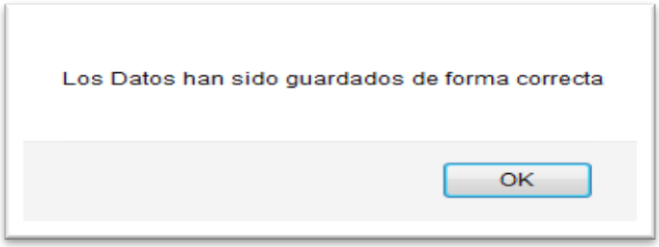


Tabla 14 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información general del Equipo" de forma incorrecta

Caso de uso	**Crear "Equipos Técnicos" Escenario "Insertar la información general del Equipo"**
Caso de prueba	Permitir insertar la información general del Equipo en la base de datos introduciendo **incorrectamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos incorrectamente o deja algún campo vacío
Resultado	El sistema muestra un mensaje de error especificando que los datos que se han introducido son incorrectos o debe llenar todos los campos
Condiciones	Existen campos vacíos o datos mal introducidos

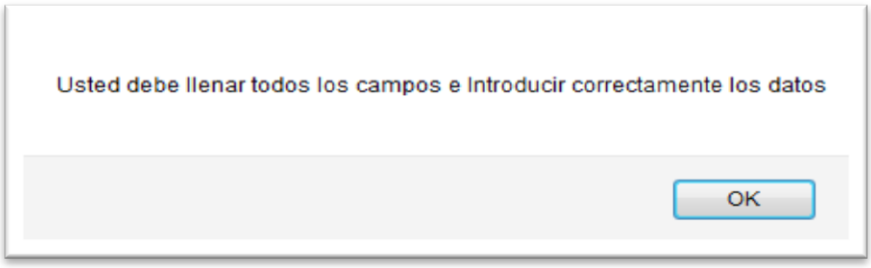


Fig. 14 Insertar la información general del equipo de forma incorrecta

Tabla 15 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información técnica del Equipo"

Condición de entrada	**Casos válidos**	**Casos no válidos**
Número del equipo	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres**	**Dejar el campo vacío e introducir letras**
Velocidad de rotación	**Cadena de caracteres numéricos de hasta 6**	**Dejar el campo vacío e introducir letras**

	caracteres permitiendo el uso de un decimal	
Potencia	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres, permitiendo el uso de un decimal**	**Dejar el campo vacío e introducir letras**
Número de etapas	**Cadena de caracteres numéricos**	**Dejar el campo vacío e introducir letras**
Sistema de lubricación	**Cadena de caracteres alfabéticos**	**Dejar el campo vacío e introducir números**
Tipo de cojetines radiales	**Cadena de caracteres alfabéticos**	**Dejar el campo vacío e introducir números**
Tipo de cojetines axiales	**Cadena caracteres alfabéticos**	**Dejar el campo vacío e introducir números**
Tipo de bombas	**Cadena de caracteres alfanuméricos de hasta 8 caracteres**	**Dejar el campo vacío**
Número de sello	**Cadena de caracteres alfanuméricos de hasta 8 caracteres**	**Dejar el campo vacío**

Tabla 16 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información técnica del Equipo" de forma correcta

Caso de uso	**Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información técnica del Equipo"**

Caso de prueba	Permitir insertar la información técnica del Equipo en la base de datos introduciendo **correctamente** los datos.
Entrada	El ingeniero de confiabilidad introduce los datos correctamente de la forma: **Número del equipo:** 110 Velocidad de rotación (rpm): 12.25 Potencia: 98.02 Número de etapas: 5 Sistemas de lubricación : Sistema de lubricación x Tipo de cojetines radiales: tipo de cojinete radial x Tipo de cojetines axiales: tipo de cojinete axial y Tipo de Bombas: BombaY2 Número de sello: API068
Resultado	El sistema introduce la información técnica de ese Equipo en la base de datos
Condiciones	El sistema muestra un cartel avisando que los datos han sido guardados

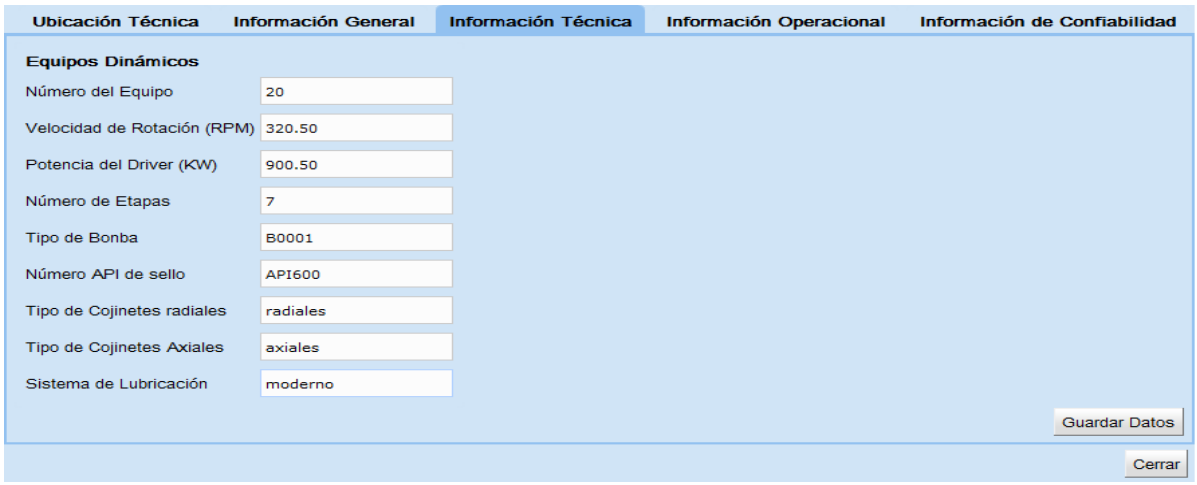


Fig. 15 Insertar información técnica del equipo de forma correcta

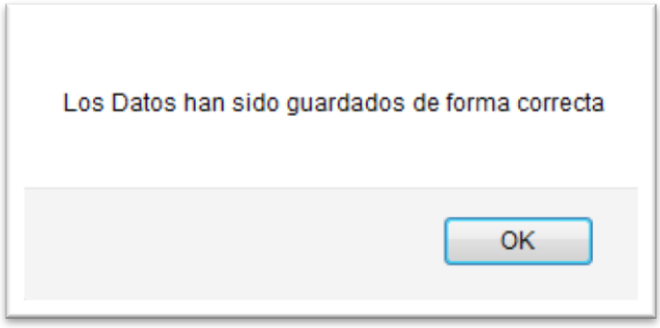


Tabla 17 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información técnica del Equipo" de forma incorrecta

Caso de uso	**Crear "Equipos Técnicos" Escenario "Insertar la información técnica del Equipo"**
Caso de prueba	Permitir insertar la información técnica del Equipo en la base de datos introduciendo **incorrectamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos incorrectamente o deja algún campo vacío
Resultado	El sistema muestra un mensaje de error especificando que los datos que se han introducido son incorrectos o debe llenar todos los campos
Condiciones	Existen campos vacíos o datos mal introducidos

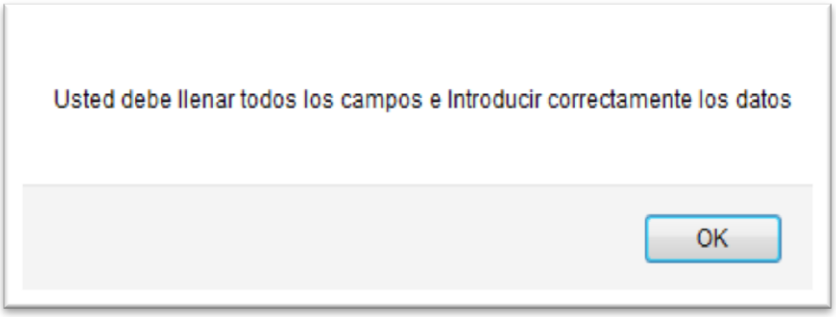


Fig. 16 Insertar información técnica del equipo de forma incorrecta

Tabla 18 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información operacional del Equipo"

Condición de entrada	Casos válidos	Casos no válidos
Número del equipo	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres**	**Dejar el campo vacío e introducir letras**
Servicio	**Cadena de caracteres alfabéticos de hasta 60 caracteres**	**Dejar el campo vacío e introducir números**
Presión operacional	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres, permitiendo el uso de un decimal**	**Dejar el campo vacío e introducir letras**
Temperatura de operación	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres, permitiendo el uso de un decimal**	**Dejar el campo vacío e introducir letras**
Humedad	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres, permitiendo el uso de un decimal**	**Dejar el campo vacío e introducir letras**

Tabla 19 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información operacional del Equipo" de forma correcta

Caso de uso	**Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información operacional del Equipo"**
Caso de prueba	Permitir insertar la información operacional del Equipo en la base de datos introduciendo **correctamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos correctamente de la forma: Número del equipo: 00001 Servicio: Servicio X Presión operacional: 78.09 Temperatura de operación : 50.8 Humedad: 0.2
Resultado	El sistema introduce la información operacional de ese Equipo en la base de datos
Condiciones	El sistema muestra un cartel avisando que los datos han sido guardados

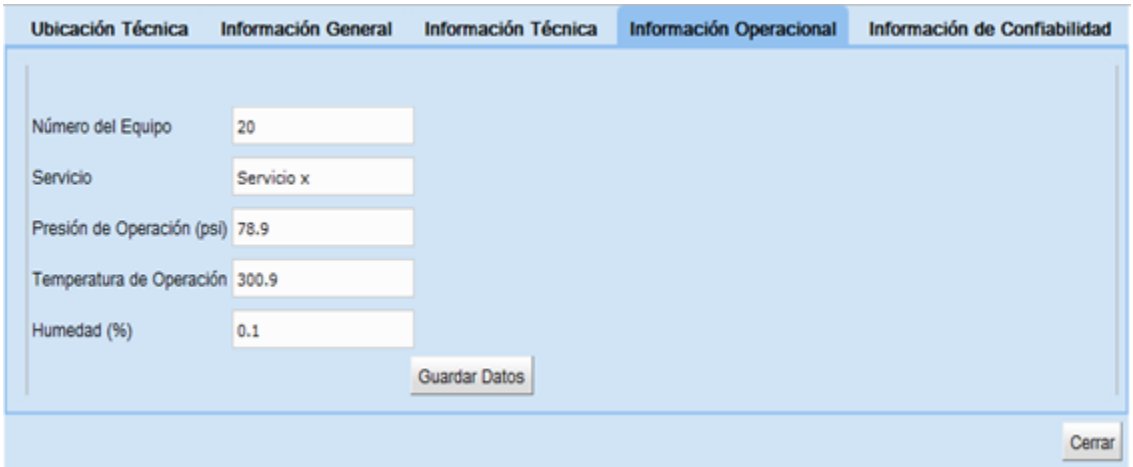


Fig. 17 Insertar información operacional de forma correcta

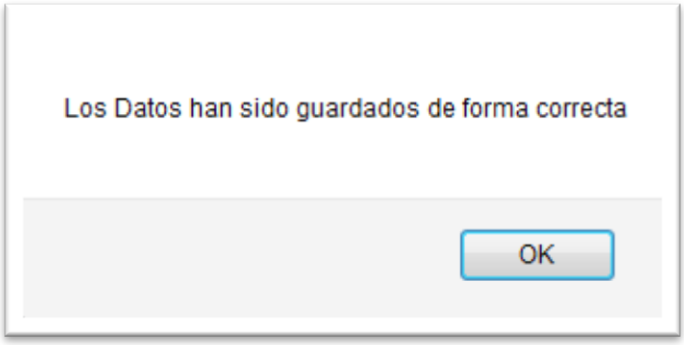


Tabla 20 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información operacional del Equipo" de forma incorrecta

Caso de uso	**Crear "Equipos Técnicos" Escenario "Insertar la información operacional del Equipo"**
Caso de prueba	Permitir insertar la información operacional del Equipo en la base de datos introduciendo **incorrectamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos incorrectamente o deja algún campo vacío
Resultado	El sistema muestra un mensaje de error especificando que los datos que se han introducido son incorrectos o debe llenar todos los campos
Condiciones	Existen campos vacíos o datos mal introducidos

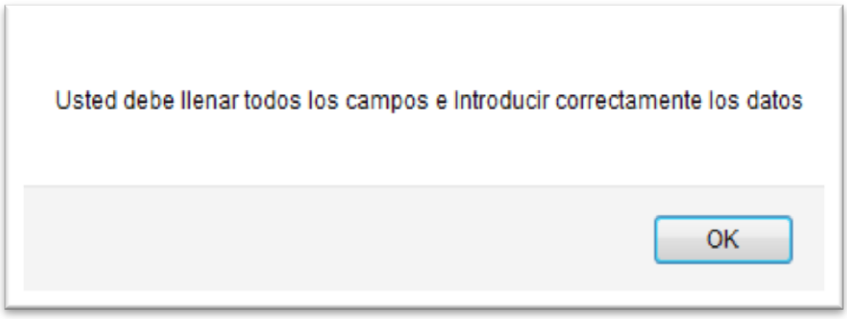


Fig. 18 Insertar información operacional del equipo de forma incorrecta

Tabla 21 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información de confiabilidad del Equipo"

Condición de entrada	Casos válidos	Casos no válidos
Número del equipo	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres**	**Dejar el campo vacío e introducir letras**
% de pérdida de producción	**Campo numérico de 5 caracteres con un valor mínimo de cero (0).**	**Dejar el campo vacío e introducir letras**
Tiempo promedio de operación entre fallas	**Cadena de caracteres numéricos**	**Dejar el campo vacío e introducir letras**
Tiempo promedio fuera de servicio	**Cadena de caracteres numéricos**	**Dejar el campo vacío e introducir letras**
Impacto ambiental	**Cadena de caracteres alfabéticos predefinidos por el sistema**	**Dejar el campo vacío**

Tabla 22 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información de confiabilidad del Equipo" de forma correcta

Caso de uso	Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información de confiabilidad del Equipo"
Caso de prueba	Permitir insertar la información de confiabilidad del equipo en la base de datos introduciendo **correctamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos correctamente de la forma: Número del equipo: 00001 % de perdida de producción: 4

	Tiempo promedio de operación entre fallas (meses): 5 Tiempo promedio fuera de servicio (hrs): 8 Impacto ambiental: Impacto severo
Resultado	El sistema introduce la información de confiabilidad de ese Equipo en la base de datos
Condiciones	El sistema muestra un cartel avisando que los datos han sido guardados

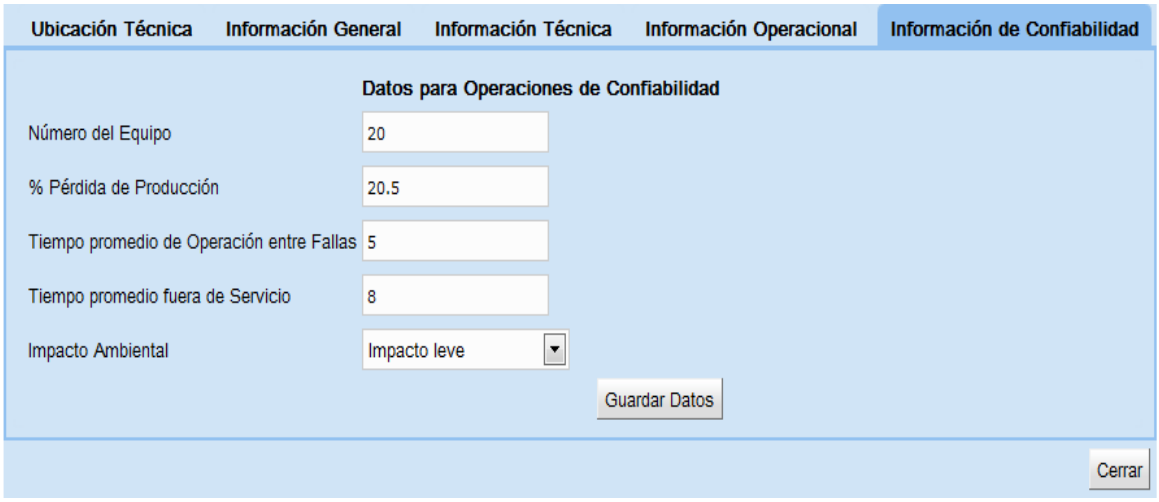


Fig. 19 Insertar información de confiabilidad de forma correcta

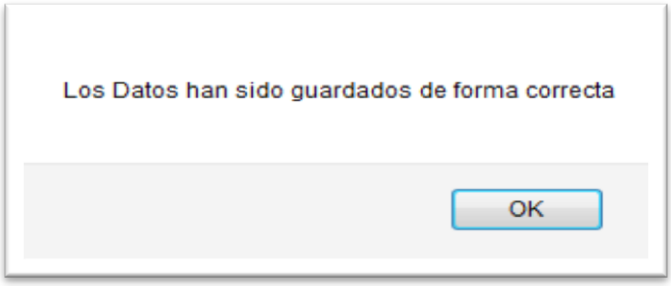


Tabla 23 Prueba de caja negra del Caso de uso Crear "Equipos Técnicos" Escenario "Insertar información de confiabilidad del Equipo" de forma incorrecta

Caso de uso	**Crear "Equipos Técnicos" Escenario "Insertar la información de confiabilidad de los equipos"**
Caso de prueba	Permitir insertar la información de confiabilidad del Equipo en la base de datos introduciendo **incorrectamente** los datos
Entrada	El ingeniero de confiabilidad introduce los datos incorrectamente o deja algún campo vacío
Resultado	El sistema muestra un mensaje de error especificando que los datos que se han introducido son incorrectos o debe llenar todos los campos
Condiciones	Existen campos vacíos o datos mal introducidos

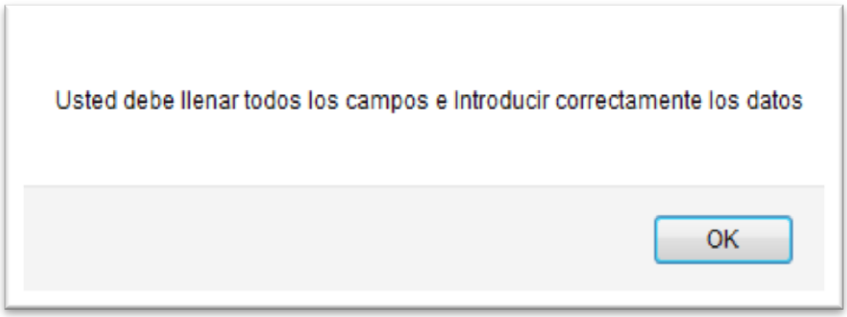


Fig. 20 Insertar datos de confiabilidad de forma incorrecta

Tabla 24 Prueba de Caja Negra del Caso de Uso "Buscar Equipo Técnico"

Condición de entrada	**Casos Válidos**	**Casos no Válidos**
Número del Equipo Técnico	**Cadena de caracteres numéricos con longitud máxima de 6 caracteres**	**Dejar el campo vacío e introducir un número incorrecto**

Tabla 25 Prueba de Caja Negra del Caso de Uso "Buscar Equipo Técnico" de forma correcta

Caso de Uso	Caso de Uso "Buscar Equipo Técnico"
Caso de Prueba	Permitir buscar un Equipo Técnico en la base de datos introduciendo **correctamente** el número del Equipo en el campo del criterio de búsqueda
Entrada	El ingeniero de Confiabilidad introduce el número del Equipo de la forma: **Número del Equipo:** 20
Resultado	El sistema muestra el Equipo Técnico que corresponde con el número entrado por el ingeniero de confiabilidad
Condiciones	El sistema muestra correctamente el Equipo Técnico

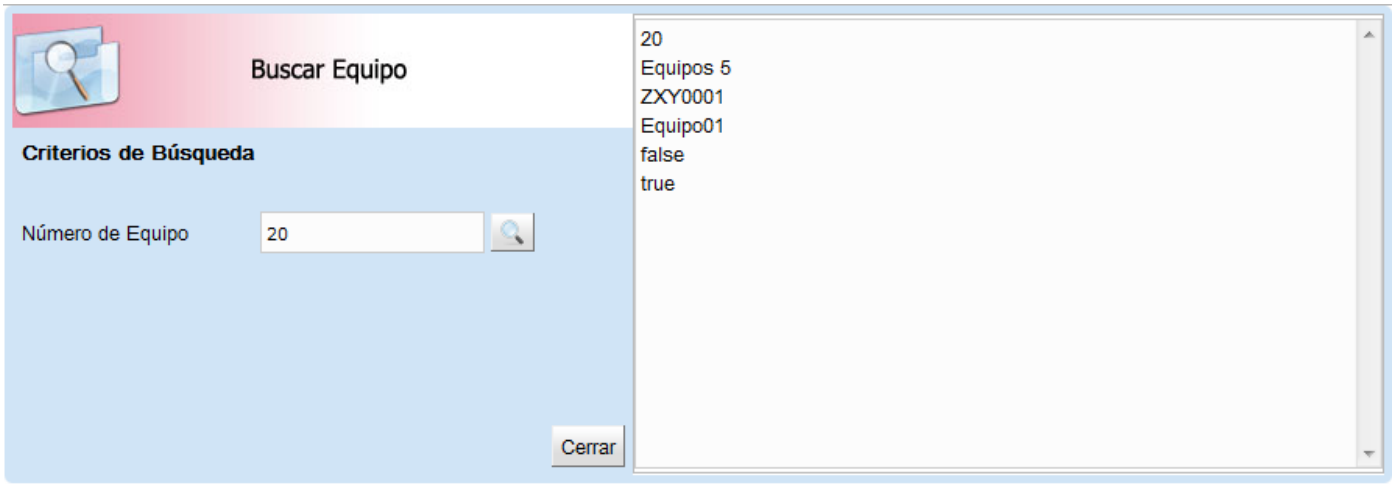


Fig. 21 Buscar un equipo de forma correcta

Tabla 26 Prueba de Caja Negra del Caso de Uso "Buscar Equipo Técnico" de forma incorrecta

Caso de Uso	Caso de Uso "Buscar Equipo Técnico"
Caso de Prueba	Permitir buscar un Equipo Técnico en la base de datos introduciendo

	incorrectamente el número del Equipo en el campo del criterio de búsqueda
Entrada	El ingeniero de confiabilidad introduce un número de equipo que no corresponde a ningún Equipo Técnico existente en la base de datos
Resultado	El sistema muestra un mensaje de error alertando de que no existen Equipos Técnicos con dicho número en la base de datos
Condiciones	El número del equipo es incorrecto

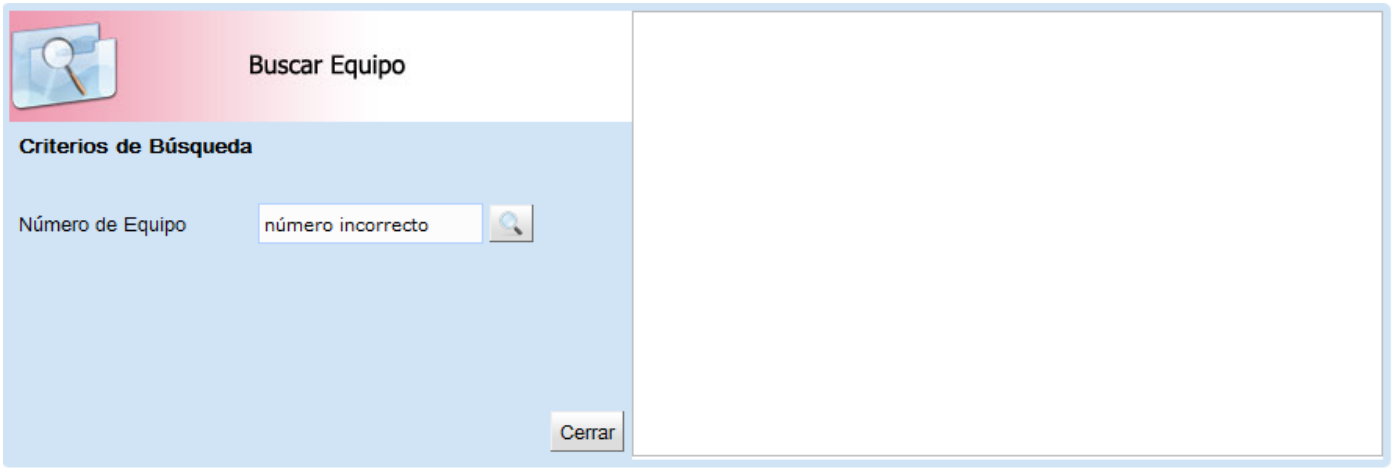
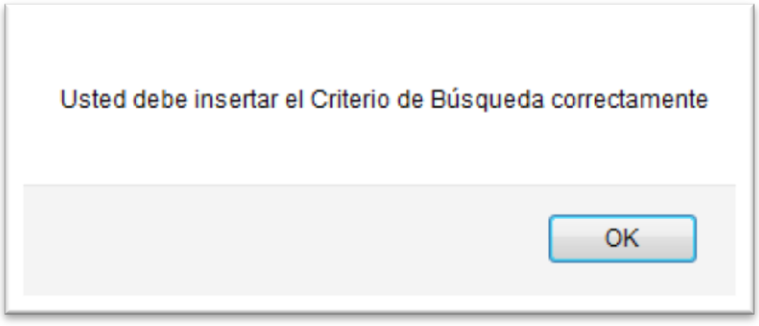


Fig. 22 Buscar un equipo introduciendo un número incorrecto



La tabla 27 muestra los resultados arrojados por la aplicación después de haberse realizado las pruebas de caja negra a los casos de prueba definidos. Como se puede observar en la tabla, se realizaron tres

iteraciones en las cuales se detectaron no conformidades, siendo en la última iteración donde se lograron resolver todas las no conformidades detectadas, dejando de esta forma al producto libre de anomalías.

Tabla 27 Resultado de las pruebas realizadas

	Iteración 1	Iteración 2	Iteración 3
No Conformidades	8	5	3
Resueltas	5	3	3
Pendientes	3	2	0

3.3 Conclusiones parciales.

En este capítulo se realizó la validación de la solución propuesta, a partir de la realización de pruebas de caja negra partiendo de los casos de prueba definidos para cuatro de los casos de uso más significativos de la aplicación, mostrando que las funcionalidades de estos casos de uso no presentan anomalías en su funcionamiento ya que realizan un correcto tratamiento de los datos que manejan y responden satisfactoriamente a los eventos del usuario. Por lo que se concluye que el resultado de las pruebas realizadas ha sido exitoso y que la aplicación cumple con las especificaciones definidas.

Conclusiones generales

Al finalizar el desarrollo de la presente investigación se concluye que se cumplieron los objetivos planteados pues:

- Se aplicaron patrones arquitectónicos, potenciando la reutilización de experiencias positivas anteriores, lo que contribuyó al ahorro de tiempo en el desarrollo de las interfaces y a una mejor estructuración del código.
- Se obtuvieron las interfaces gráficas de usuario para el proyecto SIC de acuerdo con los requisitos funcionales y no funcionales planteados y teniendo en cuenta estándares de diseño de interfaces web, así como su fácil utilización y entendimiento por los usuarios.
- Se realizaron pruebas a la aplicación, permitiendo corregir errores y verificar el funcionamiento correcto de las interfaces y su integración con otros componentes del proyecto SIC.

Recomendaciones

Se recomienda:

- Confeccionar el manual de usuario y la ayuda del sistema.
- Integrar otras metodologías de análisis de criticidad de equipos además de la metodología ACIA (Análisis de Criticidad Integral de Activos).

Bibliografía

AjaxLine. 2010. *Most Popular JavaScript Frameworks.* [Online] marzo 10, 2010. [Cited: Diciembre 12, 2012.] http://www.ajaxline.com/10-most-popular-javascript-frameworks.

Alonso, José Manuel. 2005. *El W3C –Oficina Española.* [Online] marzo 14, 2005. [Cited: Diciembre 14, 2011.] http://www.w3c.es/Presentaciones/2005/0314-estandares-JA/.

Campo, Jimmy. 2008. *Sistemas de Información – Arquitectura Cliente-Servidor .* [Online] 2008. [Cited: febrero 12, 2012.] http://www.slideshare.net/jcampo/cliente-servidor-307243.

Catalani, Exequiel. 2012. *Arquitectura Modelo/Vista/Controlador .* [Online] 2012. [Cited: febrero 20, 2012.] http://exequielc.wordpress.com/2007/08/20/arquitectura-delovistacontrolador/.

César de la Torre Llorente, Unai Zorrilla Castro, Javier Calvarro Nelson, Miguel Ángel Ramos Barroso. 2010. *Guía de Arquitectura N-Capas Orientada al Dominio con .NET 4.0.* [Online] 2010. http://www.scribd.com/doc/60751645/170/Patron-MVP-Modelo-Vista-Presentador.

Cristian Castiblanco. 2007. *Introduccion-al-Google-Web-Toolkit.* [Online] 2007. [Cited: Noviembre 25, 2011.] http://www.scribd.com/doc/48999317/.

EcuRed. 2011. *IDE de Programación Citado el.* [Online] septiembre 26, 2011. [Cited: febrero 26, 2012.] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

Erika Camacho, Fabio Cardoso- Gabriel Núñez. 2004. *Arquitecturas de Software .* [Online] 2004. [Cited: marzo 6, 2012.] http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf.

Gallego, Juan Pablo Gómez. 2006. *Capítulos de Ingeniería de Requerimientos.* [Online] 2006. [Cited: abril 24, 2012.] http://es.scribd.com/doc/270431/Ingenieria-requerimientos.

Henst, Christian Van Der. 2005. ¿Qué es la Web 2.0? [Online] 2005. [Cited: 5 12, 2012.] http://www.maestrosdelweb.com/editorial/web2/.

IEEE, 1471. 2000. *Arquitectura.* [Online] 2000. [Cited: marzo 6, 2012.] http://www.sei.cmu.edu/architecture/definitions.html.

Isaías Carrillo Pérez, Aureliano David Rodríguez Martín, Rodrigo Pérez González. 2008. *Metodología de Desarrollo del Software [En Línea]. Citado el 6-5-2012.* [Online] octubre 15, 2008. [Cited: mayo 6, 2012.]

Izabel. 2007. GWT Tutorial – Introduction to GWT. [Online] 2007. [Cited: Diciembre 11, 2011.]

Johansson, Roger. 2008. *Developing with Web Standards .* [Online] 2008. [Cited: abril 12, 2012.] http://www.456bereastreet.com/lab/developing_with_web_standards/webstandards/.

Jorge. 2012. W3C- Ventajas de los Estándares Web. [Online] 2012. [Cited: Diciembre 12, 2011.] http://blog.ivetres.com.ar/index.php/360/las-ventajas-de-los-estandares-web-ocho-razones-para-escuchar-al-w3c/.

Kappel, Gerti. 2006. *The Discipline of Systematic Development of Web Applications y & Sons Ltd.* s.l. : Web Engineering, 2006. pp. 1- 22. Vol. An Introduction to Web Engineering.

Lacalle, Alberto. 2010. Importancia de una buena Interfaz. [Online] 2010. [Cited: abril 5, 2012.] http://albertolacalle.com/hci/interfaz.htm

Maldonado, Daniel M. 2007. *Que son los Frameworks.* [Online] agosto 19, 2007. [Cited: 3 13, 2012.] http://www.elcodigok.com.ar/2007/08/que-son-los-frameworks/.

Murillo, Felix. 1999. Herramientas CASE. [Online] 1999. [Cited: marzo 24, 2012.] http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

Myers. 1996. Definición de Interfaz de Usuario . [Online] 1996. [Cited: Novbienbre 4, 2011.] http://www.fismat.umich.mx/~crivera/tesis/node6.html.

Oposiciones TIC © 2010. 2010. *Introducción a la Arquitectura Cliente –Servidor.* [Online] 2010. [Cited: marzo 6, 2012.] http://oposicionestic.blogspot.com/2011/06/arquitectura-cliente-servidor.html.

Pressman. 2002. *Ingeniería de Software Un enfoque práctico. Quinta edición.* [Online] 2002. [Cited: abril 22, 2012.] http://www.scribd.com/doc/6722160/Roger-S-Pressman-Ingenieria-Del-Software-Un-Enfoque-Practico.

Ramírez., Lic. Elisa Arizaca. 2009. *Artefacto: Diagrama de componentes.* La Paz : s.n., 2009.

Ramsdale, Chris. 2010. *Google Developer Relations - Large scale application development and MVP .* [Online] marzo 2010. [Cited: mayo 15, 2012.] http://code.google.com/intl/es-ES/webtoolkit/articles/mvp-architecture-2.html.

Riesco, Daniel. 2009. *UML Diagrama de Clases y de Objetos.* [Online] 2009. [Cited: mayo 24, 2012.] http://sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf.

Rouse, Margaret. 2005. Definition de Rational Rose. [Online] 2005. [Cited: marzo 10, 2012.] http://searchcio-midmarket.techtarget.com/definition/Rational-Rose.

Sánchez, María A. Mendoza. 2012. *Metodologías De Desarrollo De Software Analista y Desarrolladora – Team Soft Perú S.A.* [Online] mayo 2012. [Cited: mayo 12, 2012.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software.

Sierra, Daniel. 2007. *Visual Paradigm For Uml.* [Online] noviembre 15, 2007. [Cited: abril 7, 2012.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

Toledo, Carlos Aimacaña. 2008. Interfaz de Usuario . [Online] 2008. [Cited: Novienbre 2, 2011.] http://www.monografias.com/trabajos6/inus/inus.shtml.

Vassallo, Daniel. 2010. *Can I use ExtJs without License like JQuery?* [Online] febrero 10, 2010. [Cited: mayo 16, 2012.] http://stackoverflow.com/questions/2235449/can-i-use-extjs-without-license- like-jquery.

Glosario de Términos

- **Activos**: conjunto de bienes tangibles o intangibles de los que es titular una empresa.
- **Análisis de Confiabilidad:** ejecución de diversos tipos de exámenes a productos o sistemas para determinar cuan confiable es el producto o sistema en cuestión.
- **API**: acrónimo de “Application Programing Interface”. Interfaz de programación de aplicaciones que puede ser utilizada por otro software como una capa de abstracción.
- **AJAX**: acrónimo de “Asynchronous JavaScript And XML”. Conjunto de tecnologías que permite crear aplicaciones web interactivas.
- **Bytecode**: conjunto de instrucciones diseñadas para ser ejecutadas por un intérprete de software.
- **Debugging**: proceso de identificación y corrección de errores de programación.
- **Framework**: marco de trabajo que constituye una estructura conceptual y tecnológica compuesta por componentes de software que permiten el desarrollo rápido de aplicaciones.
- **J2EE**: acrónimo de “Java 2 Enterprise Edition”. Conjunto de tecnologías que define el estándar para el desarrollo de aplicaciones empresariales de varios niveles.
- **Plug-in**: módulo de hardware o software que añade funcionalidades a un sistema más grande.
- **RPC**: acrónimo de “Remote Procedure Call”. Mecanismo basado en los Servlets de Java para proveer acceso a los recursos del servidor.
- **Spring**: framework de código abierto para la plataforma Java.
- **Widget**: término genérico de una GUI que brinda una interfaz a los usuarios para interactuar con la aplicación y el sistema operativo.