Instituto Superior Minero Metalúrgico de Moa "Dr. Antonio Núñez Jiménez"
Facultad Geología y Minas
Departamento de Informática

TRABAJO DE DIPLOMA

Presentado en Opción al Título de Ingeniero Informático.

Sistema de Gestión de Información de los parámetros del viento

Autor: Adrian Pierra Fuentes
Tutor: Roiky Rodriguez Noa.

Moa, 2013

Declaración de Autoría:

Declaro que soy el único autor de este trabajo y si el lector percibe dificultades, debilidades o insuficiencia en el desarrollo del mismo debe saber que yo soy el único responsable de tales carencias, dicho esto autorizo al Instituto Superior Minero Metalúrgico de Moa “Dr. Antonio Núñez Jiménez” y al Departamento de Informática para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los días____, del mes____ del 2013.

Autor:

Adrian Pierra Fuentes.

_____

Firma

Tutor:

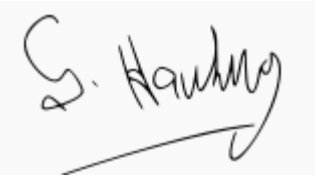
Roiky Rodríguez Noa.

_____

Firma

Pensamiento:

"Por lo tanto comencé a trabajar por primera vez en mi vida. Para mi sorpresa descubrí que me gustaba".



Agradecimientos:

Resumen:

Dentro del esfuerzo mundial que se realiza por encontrar fuentes de energía alternativas a la obtenida a partir de los combustibles fósiles, capaces de cubrir la demanda creciente de la población y que a su vez contribuyan con el desarrollo sostenible de los países, la energía eólica ha despuntado como una de las principales. En Cuba la instalación de aerogeneradores ha permitido la utilización de este tipo de energía, pero además ha potenciado los estudios en el aprovechamiento eficiente de la misma.

En este contexto es muy demandado por investigadores, los datos registrados por las estaciones meteorológicas móviles que monitorean los parámetros de viento (velocidad y dirección). Esto plantea la problemática de garantizar la disponibilidad y fiabilidad de estos datos, así como la interpretación de la información que se puede extraer de ellos.

En este trabajo se describe el proceso de desarrollo de una aplicación informática, que permite darle solución a la problemática anterior. Para guiar el desarrollo de esta aplicación se utilizó la metodología OpenUP. Por las características del entorno donde se va a desplegar el sistema se utilizó una distribución “The Broker”, esto determinó la utilización de tecnologías y herramientas para la construcción de una aplicación intermedia, ejecutada en un ambiente de escritorio, y una aplicación Web para la divulgación de los parámetros de viento y gráficos representativos de la información contenida en dichos parámetros.

Abstract:

Within the worldwide effort that comes true to find alternative sources of energy to the obtained starting from the fossil fuels, capable to cover the increasing request with the population and that in turn they contribute with the sustainable development of the countries, eolic energy has excelled like one of principal. In Cuba aerogenerar's installation has permitted the suchlike utilization of energy, but besides he has increased the power of the studies in the efficient use thereof.

In this context he is very defendant by researchers, the data registered by the movable weather stations that monitor the parameters of wind (speed and direction). This presents the problems to guarantee the availability and reliability of these data, as well as the interpretation of information that can be extracted of them.

In this work the process of development of information-technology application is described, that it permits to give him solution to the problems previous. In order to direct the development of this application the OpenUP methodology was used. For the characteristics of surroundings where the system goes to open out the “the broker” distribution was used, this determined the utilization of technologies and tools for an intermediate application's construction, executed in an environment of desk, and an application Web for the divulgation of parameters of wind and graphic representatives of information contained in said parameters.

Índice:

Introducción.

En la actualidad, en el mundo se presentan dos tendencias negativas que influyen en las características socioeconómicas de todos los países: la utilización de combustibles fósiles que provoca altos índices de contaminación medioambiental reflejados en el constante cambio climático y por otra parte, el agotamiento acelerado de estos recursos mantiene en alerta a las organizaciones energéticas mundiales. En diversos escenarios se han desarrollado cumbres, foros, y otros debates en pos de encontrar soluciones a los males que enfrenta el planeta en materia de energía y medio ambiente.

El uso de energías renovables parece ser una buena opción, varios países desarrollan una política energética con base en estos recursos renovables, actividad avalada por excelentes resultados. Todas las fuentes de energía renovable (excepto la mareomotriz y la geotérmica) provienen del Sol. La energía eólica es una fuente indirecta de energía solar, ya que son las diferencias de temperaturas y de presión inducidas en la atmósfera por la absorción de la radiación solar las que ponen en movimiento los vientos.

En Cuba el uso de energías renovables no es una novedad, aunque no está bien fortificado existen varios proyectos de estudio y algunos materializados como es el caso de los cuatro parques eólicos experimentales con que contamos hasta el presente, ellos son:

1. Turiguanó, provincia de Ciego de Ávila, 434 kilómetros al este de La Habana, de 0,45 Mega Watts, instalado en 1999.
2. Los Canarreos en el Municipio Isla de la juventud, con capacidad total de generación de 1.65 Mega Watts, permitiendo atender el 10% de la demanda del municipio pinero, instalado en febrero del 2007.
3. Gibara1, provincia de Holguín, con una capacidad de 5.1 Mega Watts, instalado en febrero 2008.
4. Gibara 2, provincia de Holguín, con una capacidad temporal de poco más de 0,40 Mega Watts, instalado en marzo de 2010.

La instalación de los aerogeneradores que conforman los parques eólicos antes mencionados parte de un estudio del potencial eólico del territorio, labor que se vio concretada con la construcción del Mapa de Potencial Eólico de Cuba en el 2005 y su última actualización se realizó en el año 2013. La conformación de este potencial, tiene como base los datos obtenidos de la medición continua de los parámetros de viento (velocidad y dirección) por varios años en varias zonas del territorio nacional.

Actualmente la forma en que los datos y la información llegan a las personas ha sido muy influenciada por el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) y particularmente su aplicación en los Sistemas de Gestión de Información (SGI). Mediante un proceso de abstracción se pueden identificar elementos fundamentales que componen el SGI del comportamiento del viento, estos elementos son: datos, información y la vía en que esa información llega a los usuarios. Partiendo de esto se pude describir como las TIC pueden favorecer este sistema:

En primer lugar tenemos los Sistemas de Gestión de Bases de Datos que han permitido manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para los usuarios.

En segundo lugar el desarrollo de las tecnologías para la construcción de aplicaciones Web que ha potenciado el uso de Internet o de una Intranet como una vía para distribuir información a miles de usuarios potenciales. Estas aplicaciones pueden contener elementos que permiten la comunicación activa entre los usuarios y la información, mediante la respuesta a acciones que los usuarios pueden realizar como tabular, listar y graficar datos.

En la actualidad para hacer la recogida de los parámetros de viento, usados para determinar el potencial eólico, se cuenta con un sistema móvil compuesto por torres, en cada una se tienen instalados a tres alturas diferentes anemómetros que se encargan de medir la velocidad del viento y en la parte superior de la torre un rumbómetro encargado de medir la dirección. Además el sistema cuenta con un equipo de cómputo al que se conectan los dispositivos mencionados, este equipo es el encargado a través de un software de registrar en una hoja de MS Excel los datos.

Hasta aquí se ha logrado un paso importante en el conocimiento del comportamiento del viento en estas zonas, pues se ha logrado almacenar un conjunto representativo de *datos*. Para

lograr la meta de convertir estos datos en información es necesario organizarlos y representarlos de forma tal que adquieran un sentido *informativo*.

Por lo todo lo antes expuesto se define el siguiente **problema de investigación:**

¿Cómo informatizar el proceso de gestión de los datos de monitoreo del viento, generados por las estaciones meteorológicas móviles que se utilizan en el estudio de las áreas idóneas para la instalación de parques eólicos en Cuba, de forma que facilite su interpretación?

Objeto de estudio: El desarrollo de sistemas informáticos para la gestión de información.

Campo de Acción: Sistemas de gestión de información de los parámetros de viento, generados por las estaciones meteorológicas móviles que se utilizan en el estudio de las áreas idóneas para la instalación de parques eólicos en Cuba.

Objetivo general:

Desarrollar un sistema informático como apoyo a la gestión de información de los parámetros de viento en los parques eólicos.

Objetivos Específicos:

- Seleccionar las metodologías y herramientas para el desarrollo de sistemas de gestión de Información.
- Documentar el proceso de desarrollo del software en correspondencia con la metodología seleccionada.
- Determinar la factibilidad de desarrollo del proyecto.
- Implementar un caso de estudio que sirva como estándar a la futura ampliación de las funcionalidades del sistema.

Para dar cumplimiento a los objetivos y resolver la situación problémica planteada, proponemos las siguientes **tareas de investigación**:

1. Estudio de las principales características para sistema de gestión de información de los parámetros de viento para los parques eólicos.
2. Analizar el formato de los datos de salida de los equipos de medición.

3. Selección de metodología de desarrollo de software, lenguaje y herramientas que se utilizarán.
4. Validación del sistema propuesto.

Idea a defender: Informatizar el proceso de gestión de los datos de monitoreo del viento, generados por las estaciones meteorológicas móviles, que se utilizan en el estudio de las áreas idóneas para la instalación de parques eólicos en Cuba, facilitará su interpretación.

Los Métodos Teóricos y Métodos Empíricos para la investigación científica que se utilizaron se describen a continuación:

Como **Métodos teóricos** se utilizaron:

Análisis y Síntesis: este método se utiliza para desglosar el problema en partes o subproblemas, para de esta forma comprobar funcionamiento de los mismos, luego integrarlo todo para corroborar las relaciones entre estas y su integración con un todo llegando así a una mejor solución, también para arribar a conclusiones y generales de la investigación.

Histórico – Lógico: para la búsqueda de antecedentes del software, las herramientas utilizadas así como la forma en que se lleva a cabo la fase de elaboración del presupuesto en el ISMMM.

Como **Métodos Empíricos** se utilizaron:

Entrevistas: para determinar los requerimientos funcionales del sistema Informático que se quiere construir. Se llevó a cabo un diálogo con personas expertas en la materia.

Análisis de documentos: para elaborar los fundamentos teóricos que se relacionan con el campo de acción.

Revisión de documentos: lo utilizamos para conocer los detalles del funcionamiento del proceso de planificación del presupuesto.

El trabajo está estructurado en tres capítulos:

Capítulo 1. Fundamentación Teórica: Contiene la fundamentación teórica del tema, donde se abordan los lenguajes de programación y las tecnologías que se utilizan en el desarrollo de la aplicación. También se exploran soluciones existentes similares al campo de acción para tener una guía de las posibles automatizaciones que se pueden realizar.

Capítulo 2. Análisis y diseño de la aplicación: Este capítulo estará enfocado a la solución del problema en cuestión, en el mismo se realizará una descripción de la propuesta del sistema. Se incluyen las plantillas que se generan según la metodología empleada, que permite una mejor comprensión del proceso de desarrollo.

Capítulo 3. Implementación y realización de pruebas: A partir de los resultados obtenidos en el capítulo anterior, se procede a validar los mismos, mediante las diferentes técnicas de validación, como son: las revisiones, los prototipos no funcionales. Los resultados obtenidos de las diferentes validaciones son analizados y corregidos.

Capítulo 4. Estudio de Factibilidad: En este se realiza un estudio para ver la factibilidad del producto por la metodología Coste-Beneficio. Además de un estudio de los esfuerzos requeridos para la realización del sistema propuesto.

Capítulo I. Fundamentación Teórica.

Introducción.

En el presente capítulo se describe de forma general los aspectos relacionados con el objeto de estudio y el campo de acción en que se trabaja. Este capítulo constituye la base teórica para la comprensión del trabajo que se desarrolla y sus principales aspectos.

1.1 Energía Eólica.

La energía eólica es una forma indirecta de energía solar, puesto que son las diferencias de temperatura y de presión inducidas en la atmósfera por la absorción de la radiación solar las que ponen en movimiento los vientos. Se calcula que un 2 % de la energía solar recibida por la Tierra se convierte en energía cinética de los vientos, la cantidad de energía correspondiente es enorme: unos 30 millones de TWh por año, o sea, 500 veces el consumo mundial de energía en 1975. Teniendo en cuenta que sólo el 10 % de esta energía se encuentra disponible cerca del suelo, el potencial sigue siendo considerable; así, es difícil concebir en la actualidad la explotación de una parte notable de este potencial.

En efecto, sería necesario cubrir las tierras emergidas y las superficies marinas con enormes motores eólicos. En estas condiciones, es más razonable estimar que por mucho tiempo las aplicaciones de la energía eólica se limitarán a utilizaciones locales, en regiones aisladas - a un nivel de potencia de algunos kW a algunas decenas de kW- o bien a un papel de fuente complementaria en la alimentación de las redes eléctricas -con niveles de potencia de hasta algunos MW-. Las zonas más favorables para la implantación de grandes motores eólicos, donde la velocidad media del viento es superior a 30 Km/h, son las regiones costeras y las grandes estepas, donde estos vientos soplan regularmente. (http://www.cubasolar.cu/, 2011)

1.1.2 Impacto medioambiental de la Energía Eólica.

Generar energía eléctrica sin que exista un proceso de combustión o una etapa de transformación térmica supone, desde el punto de vista medioambiental, un procedimiento muy favorable por ser limpio y exento de problemas de contaminación. Se suprimen radicalmente los impactos originados por los combustibles durante su extracción, transformación, transporte y combustión, lo que beneficia la atmósfera, el suelo, el agua, la fauna y la vegetación.

Este comportamiento evita la contaminación que conlleva el transporte de los combustibles; gas, petróleo, gasoil, carbón; y reduce el intenso tráfico marítimo y terrestre cerca de las centrales. Suprime los riesgos de accidentes durante la transportación: desastres con petroleros y vertimiento de crudo en los mares, además, se evita la instalación de líneas de abastecimiento: canalizaciones a las refinerías o las centrales de gas.

Cada kWh de electricidad generada por energía eólica en lugar de carbón, evita:

- 0,60 Kg de CO_2 (dióxido de carbono).
- 1,33 gr de SO_2 (dióxido de azufre).
- 1,67 gr de NO (óxido de nitrógeno).

(http://www.cubasolar.cu/, 2011)

1.1.3 Desarrollo de la Energía Eólica en Cuba

Como parte de la Revolución Energética en Cuba se dan pasos firmes para el desarrollo de la energía eólica. Durante los últimos años, gracias al trabajo pionero de varias instituciones y personalidades nacionales, se realizaron algunos proyectos piloto de desarrollo eólico que le permiten al país contar actualmente con una capacidad instalada de 480 kW.

Se avanza rápidamente en los estudios previos de prospección del viento para conocer su real potencialidad, así como en la instalación de parques eólicos para probar en una escala limitada las más importantes tecnologías de aerogeneradores que hoy se conocen.

Uno de los logros iniciales de este programa eólico es la confección del primer mapa eólico de Cuba con fines energéticos. Los resultados que se obtengan de la medición del viento, más las experiencias que se adquieran en esos primeros parques abrirán el camino hacia un desarrollo superior.

Ya se instalan con fines de prospección eólica 100 estaciones anemométricas a 50 m de altura en diferentes puntos del país, lo cual en breve tiempo permitirá conocer las posibilidades reales de instalación de parques eólicos en esos lugares.

Se trabaja también en la preparación de personal técnico y de operación, por medio de cursos desarrollados al efecto e impartidos por especialistas nacionales, así como de conferencias de especialistas extranjeros de renombre. (Figueredo, 2006)

1.2 Antecedentes.

El software “SEMeFAV” Sistema Estimador de Mediciones de Faltantes de Vientos fue creado como trabajo de diploma por el licenciado en Ciencias de la Computación Aliet Lamorú Reyes. Este trabajo permite caracterizar el potencial eólico de una región, destacar en este la algoritmia para descartar los faltantes de datos en las mediciones de los parámetros del viento. (Reyes, 2011)

Figura 1.Interfaz del SEMeFAV.

Inconvenientes del software SEMeFav:

1. No fue desarrollado con herramientas libres.

2. No fue construido teniendo en cuenta los posibles entornos de despliegue de la aplicación, más bien fue construido con fines educativos.
3. No fue desarrollado teniendo en cuenta buenas prácticas de programación lo que dificulta la escalabilidad del sistema.

Destacar que las experiencias y opiniones del autor del trabajo mencionado anteriormente son utilizadas en el desarrollo de esta investigación.

1.3 Tendencias y Tecnologías Actuales.

1.3.1 Software libre.

Se denomina software libre a todo aquel que permita a los usuarios ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. A menudo es confundido con el software gratuito, sin embargo no se trata de una cuestión de precio sino de libertad. Precisamente, las cuatro libertades que se definen son:

- La libertad de ejecutar el programa para cualquier propósito.
- La libertad de estudiar cómo trabaja el programa y adaptarlo a sus necesidades (El acceso al código fuente es una condición necesaria).
- La libertad de redistribuir copias para que pueda ayudar al vecino.
- La libertad de mejorar el programa y publicar sus mejoras y versiones modificadas en general para que se beneficie toda la comunidad (El acceso al código fuente es una condición necesaria).

Las ventajas especialmente económicas que aportan las soluciones libres y las aportaciones de la comunidad de desarrollo han permitido un constante crecimiento del software libre hasta superar en ocasiones al mercado propietario. Estas ventajas hacen que el país siga una política de migración hacia el software libre y como parte de este proceso se decide para el desarrollo de la aplicación la utilización de herramientas y tecnologías pertenecientes al software libre. (GNU Operating System. , 2009)

1.3.2 Lenguajes de Programación para la creación de aplicaciones de escritorio.

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. A continuación se realiza un estudio de los principales lenguajes de programación existentes que se utilizan para la creación de aplicaciones de escritorio, como son Delphi, C++ y Java.

1.3.3 Java.

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma.

La sintaxis del lenguaje heredó características de C y C++, adoptando una muy similar a la del C++. Actualmente, dentro de los lenguajes populares es uno de los mejores en cuanto a definición, debido a que goza de total independencia del implementador del lenguaje y de sus clases auxiliares. Proporciona los tipos de datos primitivos similares a los de C++, proporciona todas las estructuras contenedoras "clásicas". Tiene cuatro niveles de empaquetamiento: variables y funciones, al igual que los lenguajes anteriores y otros dos propios de él, denominados: clases y paquetes.

Este lenguaje cuenta con una interfaz orientada a objetos para acceder de un modo portable a cualquier base de datos, promoviendo la portabilidad. Es poseedor de una extensa biblioteca utilitaria y la portabilidad alcanzada es cualitativamente superior a la que se puede obtener con los lenguajes C/C++.

Desde sus inicios se concibió como un lenguaje muy fácil de comprender y utilizar, sin que esto signifique que su aprendizaje sea trivial. (Gutierrez, 2007)

1.3.4 Entornos de desarrollo integrados (IDE)

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). (Wikipedia, 2012)

1.3.5 NetBeans.

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo, es un producto libre y gratuito sin restricciones de uso. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las API's de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en esta plataforma pueden ser extendidas fácilmente por otros desarrolladores de software. Por tales características NetBeans cumple con los requisitos necesarios para ser utilizado en el desarrollo de la aplicación. (netbeans, 2013)

1.3.6 Lenguajes de Programación para la creación de aplicaciones Web.

1.3.7 RIA

Son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales, estas aplicaciones utilizan un "navegador web" estandarizado para ejecutarse y por medio de un "plugin" o independientemente con una "virtual machine", se agregan las características adicionales. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales. Buscan mejorar la experiencia del usuario. Normalmente en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces, a recargarla misma página con un mínimo cambio. En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos. (Frank W. Zammetti, 2009)

1.3.8 ExtJS.

ExtJS es una biblioteca o conjunto de librerías de JavaScript para el desarrollo de aplicaciones web interactivas, usa tecnologías AJAX, DHTML y DOM.

ExtJS permite realizar completas interfaces de usuario, fáciles de usar, muy parecidas a las conocidas aplicaciones de escritorio. Esto permite a los desarrolladores web concentrarse en la funcionalidad de las aplicaciones en vez de en las advertencias técnicas. (Frank W. Zammetti, 2009)

Antes de adoptar este nuevo framework, es importante entender los términos de la licencia. ExtJS provee varios términos de licencia los cuales son:

Open Source License: Esta licencia está regida bajo los términos de la licencia Open Source LGPL 3.0, la cual es más apropiada para en el plan de usar ExtJS en proyectos

de código abiertos en ámbitos personales o educacionales y para aplicaciones sin fines comerciales.

Comercial License: Esta licencia es la más apropiada para el uso de ExtJS con fines para la creación de aplicaciones comerciales, por lo que deberá pagar a ExtJS por el soporte de la librería.

Original Equipment Manufacturer (OEM)/reseller license: Esta licencia es la más apropiada para el plan de re-empaquetar o cambiar ExtJS en el desarrollo del software. (Frank W. Zammetti, 2009)

1.3.9 Sistema Gestor de Base de Datos PostgreSQL.

Consiste en un conjunto de programas, procedimientos y lenguajes que nos proporcionan las herramientas necesarias para trabajar con una base de datos. Incorporar una serie de funciones que nos permita definir los registros, sus campos, sus relaciones, insertar, suprimir, modificar y consultar los datos. Actualmente existen muchos sistemas gestores de bases de datos, ejemplo de esto es PostgreSQL.

PostgreSQL es un gestor de base de datos orientado a objetos, muy conocido y usado en entornos de software libre. Es considerado el sistema gestor de base de datos de código abierto más avanzado del mundo, es gratuito, se integra perfectamente con PHP. Propone un tamaño ilimitado para las bases de datos, lo que da la medida de un gestor de bases de datos robusto. Permite que mientras un proceso escribe en una tabla, otros accedan a la misma sin necesidad de bloqueos. Es estable, flexible, se puede extender su funcionalidad y tiene gran compatibilidad con diversos sistemas operativos. (Quiñones, 2010)

1.4 Sistemas Distribuidos.

Una consideración importante a la hora de seleccionar la arquitectura de un software es las restricciones que imponen los entornos en los que se ejecutarán sus componentes. Los sistemas distribuidos nos permiten por su naturaleza resolver este problema, al tratar cada componente según los requerimientos funcionales que demanden, para su funcionamiento y el del sistema en general. Entonces se le podrían añadir a las ventajas

señaladas por (Tanembaum., 2002) sobre los sistemas distribuidos: la adaptabilidad de los componentes a los requerimientos de hardware y software en los ambientes de ejecución.

Precisamente los requerimientos que impone el medio donde se instalan temporalmente las estaciones meteorológicas precisa de un hardware de poco consumo eléctrico, esto limita la potencia de los ordenadores que se utilizan. Aquí surge una disyuntiva por una parte se necesita garantizar la disponibilidad fiabilidad de los datos y facilitar la interpretación de la información que se puede extraer de los mismos; y por otra parte existen restricciones en cuanto al consumo de recursos de los software instalados.

La creación de un sistema distribuido posibilita la utilización de un software intermediario entre el componente encargado de tomar los datos de la estación meteorológica y el componente encargado de distribuir esos datos y la información.

1.4.1 Arquitectura de sistemas de gestión de información en ambientes distribuidos. Patrón The Broker.

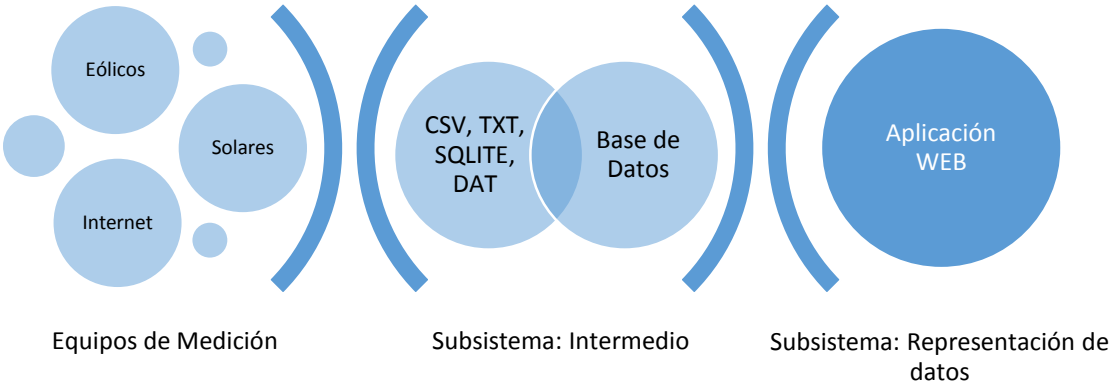


Figura 2. Arquitectura de la aplicación usando el patrón The Broker.

El patrón arquitectónico The Broker puede usarse para estructurar sistemas de software distribuidos con componentes desacoplados que interactúan por invocaciones de servicios remotos. Un componente Broker es responsable de coordinar la comunicación,

remitir las demandas, así como transmitir resultados y excepciones. El ambiente es un sistema distribuido y heterogéneo con componentes independientes que cooperan entre sí. (Tanembaum., 2002)

El patrón arquitectónico The Broker tiene algunas ventajas importantes:

1. **Transparencia**: Como el Broker es responsable de localizar un servidor usando un único identificador, los clientes no necesitan saber dónde se localizan los servidores. De igual forma, los servidores no se preocupan de la ubicación de los clientes que les realizan requerimientos ya que ellos reciben todas las solicitudes del componente broker local.
2. **Cambiabilidad y extensibilidad**: Si los servidores cambian pero sus interfaces permanecen iguales, no tiene impacto funcional en los clientes. Modificando la implementación interior del Broker pero no las API's que él provee, no tiene otro efecto en los clientes y servidores más que cambios en la performance. El uso de proxies y bridges es una razón importante para facilitar la implementación de cambios.
3. **Portabilidad**: El sistema Broker oculta detalles del sistema operativo y del sistema de red a clientes y servidores usando capas de indirección como las API's, proxies y bridges. Cuando se requiere usar puertos es suficiente en la mayoría de los casos poner el puerto en el componente broker y sus API's en una nueva plataforma y recompilar a los clientes y servidores, en estos casos, se recomienda estructurar los componentes del broker en capas.
4. **Interoperabilidad**: Diferentes sistemas Broker pueden interoperar si entienden un protocolo común para el intercambio de mensajes. Este protocolo es interpretado y manejado por bridges que son los responsables de traducir el protocolo específico del broker en el protocolo común y viceversa.
5. **Reusabilidad**: Al construir nuevas aplicaciones clientes, frecuentemente están basadas en la funcionalidad de la aplicación en servicios existentes. (Tanembaum., 2002)

1.5 Patrones de Diseño del Subsistema Intermedio.

1.5.1 Patrón Observer.

El patrón Observer define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observer se encarga de notificar este cambio a todos los otros objetos dependientes.

Se puede utilizar el uso patrón Observer cuando se presentan las siguientes situaciones:

- Una abstracción tiene dos aspectos, uno dependiente del otro. Encapsulándolos en objetos separados se permite variarlos y reusarlos de forma independiente.
- Cuando un cambio en un objeto implica cambiar otros y no se conoce de antemano cuantos objetos deben actualizarse.
- Cuando un objeto debe ser capaz de hacer notificaciones a otros sin hacer suposiciones de quiénes son, buscando un bajo acoplamiento.

Ventajas:

- Acoplamiento mínimo entre Subject y Observer: Todo lo que sabe un Subject es que tiene una lista de observers que responden a la interfaz Observer. El Subject no conoce ninguna clase observers concreta.
- Soporte para comunicación tipo broadcast: La notificación se extiende a todos los objetos de la lista. Se añaden y quitan observadores en cualquier momento.

Desventajas:

- Actualizaciones Costosas: Un observer no conoce cuántos observers más existen y por lo tanto, no conoce el costo de enviar un “cambio” al Subject. Podría producirse una actualización en cascada. (Freeman, 2010)

1.5.2 Patrón Command.

Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma. Para ello se encapsula la petición como un objeto, facilitando la parametrización de los métodos.

Al encapsular un mensaje como un objeto, permite gestionar colas o registros de mensajes, deshacer operaciones y restaurar el estado a partir de un momento dado. Ofrece una interfaz común que permite invocar las acciones de forma uniforme y extender el sistema con nuevas acciones de forma más simple.

Este patrón presenta una forma sencilla y versátil de implementar un sistema basado en comandos facilitando su uso y ampliación.

Se puede utilizar el patrón Command cuando se necesita:

1. Facilitar la parametrización de las acciones a realizar.
2. Independizar el momento de petición del de ejecución.
3. Implementar CallBacks, especificando que órdenes se necesitan ejecutar en ciertas situaciones bajo otras órdenes. Es decir, un parámetro de una orden puede ser otra orden a ejecutar.
4. Dar soporte para deshacer comandos, procesos de identificación y/o transacciones.
5. Desarrollar sistemas utilizando órdenes de alto nivel que se construyen con operaciones sencillas (primitivas).

Consecuencias:

1. Independiza el objeto que invoca una operación de un objeto que conoce como implementarla.
2. Los Command son objetos de primera clase, pueden ser manipulados y extendidos como cualquier otro objeto.
3. Se facilita la ampliación del conjunto de comandos ya que las clases existentes no cambian. (Freeman, 2010)

1.6 Patrones de Diseño del Subsistema de Representación de Datos.

1.6.1 Patrón arquitectónico Modelo Vista Controlador.

El patrón arquitectónico Modelo Vista Controlador separa la lógica de negocio (el modelo) y la presentación (la vista) logrando un mantenimiento más rápido y sencillo de las

aplicaciones. Ejemplo, para el caso de la web, si se fuera a mostrar una misma aplicación en un navegador estándar, como en un navegador de un dispositivo móvil, sólo es necesario crear una vista nueva por cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (Aplicación de escritorio, HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. (Freeman, 2010)

1.7 Patrones GRASP.

GRASP es un acrónimo de General Responsibility Assignment Software Patterns o Patrones Generales de Software para Asignar Responsabilidades. El nombre se eligió para sugerir la importancia de aprender estos principios para diseñar con éxito el software orientado a objetos. Los patrones GRASP son utilizados para describir los principios fundamentales del diseño y la asignación de responsabilidades. (Baños, 2012)

• Experto en información

El GRASP de experto en información es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada, posibilitando la disminución del acoplamiento.

• Creador

El patrón creador permite identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización.

La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia de esto resulta útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización.

• **Alta cohesión**

La cohesión es la medida de la fuerza que une a las responsabilidades de una clase. Una clase con baja cohesión hace muchas cosas no relacionadas, o hace demasiado trabajo. Tales clases no son convenientes pues son difíciles de mantener, de reutilizar y de entender. Una clase con alta cohesión mejora la claridad y la facilidad de su uso, su mantenimiento se simplifica y es fácil de reutilizar. Es decir, dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.

• **Bajo acoplamiento**

Bajo acoplamiento es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Los conceptos de cohesión y acoplamiento no están íntimamente relacionados, sin embargo se recomienda tener un mayor grado de cohesión con un menor grado de acoplamiento. De esta forma se tiene menor dependencia y se especifican los propósitos de cada objeto en el sistema.

1.8 Metodologías de desarrollo.

Las metodologías de desarrollo de software definen una serie de procedimientos, técnicas y herramientas para la realización de un producto de software. Para el desarrollo de la solución propuesta se toma OpenUP. Esta metodología de desarrollo es un proceso unificado (de aplicación general) y ágil (se centra en el desarrollo rápido de sistemas) que involucra un conjunto mínimo de prácticas que ayudan a los equipos de trabajo a ser más efectivos en el desarrollo de sistemas software (u otros sistemas de

ingeniería). OpenUP integra una filosofía pragmática y ágil que se centra en la naturaleza colaborativa del desarrollo de software. Es un proceso anti- burocrático y agnóstico en cuanto a herramientas (IDE, lenguajes, sistemas operativos, etc.) que puede ser usado o que puede ser expandido y adaptado de acuerdo a las especificaciones de cada proyecto. OpenUP está organizado en dos dimensiones diferentes pero interrelacionadas: el método y el proceso.

El contenido del método es donde los elementos del método (roles, tareas, artefactos y lineamientos) son definidos, sin tener en cuenta como son utilizados en el ciclo de vida del proyecto.

El proceso es donde los elementos del método son aplicados de forma ordenada en el tiempo. Muchos ciclos de vida para diferentes proyectos pueden ser creados a partir del mismo conjunto de elementos del método. (Fernández Y. , 2012)

1.8.1 Ciclo de vida de OpenUP.

Los integrantes del equipo contribuyen aportando micro- incrementos que puede ser el resultado del trabajo de unas pocas horas o unos pocos días. El progreso se puede visualizar diariamente, ya que la aplicación va evolucionando en función de este micro-incremento.

El objetivo de OpenUP es ayudar al equipo de desarrollo a través de todo el ciclo de vida de las iteraciones, de modo que este sea capaz de añadir valor de negocio para los clientes de una forma predecible: con la entrega de un software operativo y funcional al final de cada iteración. El ciclo de vida del proyecto provee a los clientes de una visión del proyecto, transparencia y les dota de los medios para que les permitan controlar la financiación, el riesgo, el ámbito, el valor de retorno esperado, etc.

Los elementos del OpenUP dirigen la organización del trabajo en los niveles personal, de equipo y de interesados. A nivel personal, los integrantes de un proyecto contribuyen con su trabajo con pequeños incrementos en funcionalidad denominados micro incrementos, los cuales representan los resultados obtenidos en pocas horas o pocos días de trabajo. La solución evoluciona basada en dichos micro incrementos de tal forma

que el progreso puede ser visualizado efectivamente cada día. Los integrantes del equipo de desarrollo de forma abierta comparten su progreso diario el cual incrementa la visibilidad en el trabajo, la confianza y el trabajo en equipo.

El proyecto en general se divide en iteraciones, las cuales son planificadas en un intervalo definido de tiempo que no superan las pocas semanas. El OpenUP tiene elementos que ayudan a los equipos de trabajo a enfocar los esfuerzos a través del ciclo de vida de cada iteración de tal forma que se puedan distribuir funcionalidades incrementales de una manera predecible una versión totalmente probada y funcional al final de cada iteración. (Fernández Y. , 2012)

1.10.2 Fases.

Fase Inicio: Las necesidades de cada participante del proyecto son tenidas en cuenta y son plasmadas en objetivos del proyecto. Se deben definir el ámbito del proyecto, los límites del mismo y el criterio de aceptación del proyecto. Los casos de uso críticos, aquellos que dirigen la funcionalidad del sistema, son definidos en esta fase, así como una estimación inicial del coste del proyecto y un boceto de la planificación. El propósito en esta fase es lograr concurrencia entre todos los stakeholders sobre los objetivos del ciclo de vida para el proyecto.

Hay cuatro objetivos para la fase de Inicio que clarifican el alcance, los objetivos del proyecto y la viabilidad de la solución proyectada:

- Entender qué construir. Determinar la visión, el alcance del sistema y sus límites. Identifique quién está interesado en este sistema y por qué.
- Identificar la funcionalidad clave del sistema. Decidir qué requerimientos son los más críticos.
- Determinar al menos una posible solución. Identificar al menos una arquitectura candidata y su viabilidad.
- Entender el costo, el cronograma y los riesgos asociados al proyecto.

Fase Elaboración: En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Si se decide continuar con el proyecto se debe elaborar

un plan de proyecto en esta fase, para lo cual se deben establecer unos requisitos y arquitectura estables. Por otro lado el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo también se especifican en detalle en esta fase. Al final de la fase se debe tener una definición clara y precisa de los casos de uso, los actores, la arquitectura del sistema y un prototipo ejecutable de la misma.

Hay objetivos para la fase de Elaboración que le ayudan a direccionar los riesgos asociados con los requisitos, la arquitectura, los costos y el cronograma:

- Obtener un entendimiento más detallado de los requisitos.
- Diseñar, implementar, validar y establecer la línea base para la arquitectura.
- Mitigar los riesgos esenciales y producir un cronograma exacto y unos costos estimados.

Fase Construcción: Todos los componentes y funcionalidades del sistema que falten por implementar son realizados, testeados e integrados en esta fase. Los resultados obtenidos en forma de incrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado. El propósito de esta fase es completar el desarrollo del sistema basado en la arquitectura. Hay objetivos para la fase de Construcción que nos ayudan a tener un desarrollo con costo eficiente de un producto completo una versión operativa del sistema que pueda ser entregada a la comunidad de usuarios:

- Desarrollar iterativamente un producto completo que esté listo para hacer transición a su comunidad de usuarios.
- Minimizar el costo de desarrollo y alcance algún grado de paralelismo.

Fase Transición: El propósito de esta fase es asegurar que el producto software está listo para ser distribuido a los usuarios. Hay objetivos para la fase de Transición que le ayudan a afinar elegantemente la funcionalidad, el desempeño y la calidad total de la versión beta del producto desde el final de la fase previa.

- La prueba beta valida que las expectativas del usuario sean satisfechas. Esto típicamente requiere algunas actividades de afinamiento, tales como depuración de errores y mejora del desempeño y la usabilidad.
- Lograr la concurrencia de interesados que el despliegue se ha completado. Esto puede implicar que los distintos niveles de las pruebas de aceptación del producto, incluyendo pruebas formales e informales y las pruebas beta.
- Mejorar el desempeño en futuros proyectos a través de lecciones aprendidas. Documentar las lecciones aprendidas y mejorar el ambiente de los procesos y las herramientas para el proyecto. (Fernández Y. , 2012)

1.11 Herramienta de Modelado Visual Paradigm.

Esta herramienta que soporta el Lenguaje Unificado de Modelado (UML) y permite generar artefactos del ciclo de vida del desarrollo de software, ayuda a una rápida construcción de aplicaciones de calidad; permite elaborar todo tipo de diagramas de clases, código inverso, generar códigos desde diagramas y generar documentación.

Además presenta características como:

- Permite generar varios diagramas, dentro de los que se encuentran los diagramas de procesos de negocio.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.

1.12 Conclusiones del Capítulo.

Después del estudio realizado de los diferentes aspectos tratados en este capítulo quedan sentadas las bases para el desarrollo del Sistema de Información de Energía Eólica, la cual estará formada por dos subsistemas regidos por el patrón The Broker, el

primero de estos subsistemas o subsistema intermedio es para la captura de datos arrojados por el equipo de medición e insertarlos a la base de datos del sistema y el segundo subsistema o subsistema de representación de datos es para presentación y análisis mediante la Web de los datos almacenados en las base de datos.

- Metodología desarrollo: OpenUP.
- Herramienta Case: Visual Paradigm.
- Lenguajes de Programación: Java y JavaScript.
- Herramientas de Desarrollo: NetBeans y ExtJS.

Capítulo 2. Descripción de las Soluciones Propuestas.

2.1 Introducción.

En el presente capítulo se realizará el análisis, diseño y desarrollo de la solución propuesta para el problema planteado, haciendo uso de la metodología OpenUP antes mencionada para el desarrollo de la solución. Para ello se definirá la arquitectura de la herramienta, así como su proceso de funcionamiento en virtud de cumplir con los requisitos funcionales de la misma. Se hará la modelación de los diagramas fundamentales.

2.2 Actores del Sistema.

Nombre del Actor	Descripción
Especialista Técnico	El especialista técnico es la persona encargada de gestionar la carga automática o manual, los datos de los equipos al sistema de información.
Especialista Funcional	El especialista funcional es la persona encargada de visualizar el estudio de los datos mediante un navegador web.

2.3 Requisitos del Sistema.

Se desarrollará un sistema compuesto por dos subsistemas, el primero encargado extraer los datos de un fichero arrojado por equipos para medir las condiciones eólicas y el segundo es el responsable de gestionar la información asociada a los datos registrados por el primer subsistema.

El sistema deberá cumplir con los requisitos que a continuación se relacionan:

2.3.1 Requisitos Funcionales.

Un requisito funcional define el comportamiento interno del software como detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica, es decir, capacidades o condiciones que el sistema debe cumplir.

CU1: Establecer conexión con la base de datos.

RF1: Configurar la conexión con la base de datos.

RF2: Conectarse a la base de datos.

CU2: Gestión de archivo.

RF3: Buscar el archivo en el equipo.

CU3: Completamiento de datos.

RF4: Definir los datos faltantes.

CU4: Mostrar datos.

RF5: Muestra los datos transformados.

CU5: Insertar los datos a la base de datos.

RF6: Insertar los datos a la base de datos.

RF7: Visualizar el progreso de la operación.

RF8: Cancelar la operación.

CU6: Ejecutar aplicación web.

RF9: Configurar la url del navegador web.

RF10: Ejecutar navegador web.

CU7: Mostrar Mediciones según Fecha y Localización.

RF11: Mostrar las localidades según la fecha.

RF12: Mostrar las mediciones según localidad y fecha.

RF13: Mostrar los gráficos según localidad y fecha.

CU8: Generar gráficas.

RF14: Generar gráfica de serie según altura y velocidad del viento.

RF15: Generar gráfica de manta según dirección y altura.

RF16: Generar gráfica de barra según potencial eólico y altura.

CU9: Vista ampliada del gráfico de Manta.

RF17: Mostrar una vista ampliada del gráfico de Manta.

RF18: Mostrar y manipular leyenda del gráfico de Manta.

CU10: Vista ampliada del gráfico de Serie.

RF19: Mostrar una vista ampliada del gráfico de Serie.

RF20: Mostrar y manipular leyenda del gráfico de Serie.

CU11: Vista ampliada del gráfico de Barra.

RF21: Mostrar una vista ampliada del gráfico de Barra.

RF22: Mostrar y manipular leyenda del gráfico de Barra.

2.3.2 Requisitos no Funcionales.

Un requisito no funcional especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar. En otras palabras son las propiedades o cualidades que el producto debe tener.

Apariencia o interfaz externa.

RNF1: La aplicación deberá presentar y mantener una interfaz amigable, interactiva, intuitiva y entendible por el usuario.

Usabilidad.

RNF3: El Especialista Técnico permanecerá en el sistema el tiempo que así lo considere.

RNF4: Para el trabajo con el sistema se requerirán conocimientos mínimos.

Accesibilidad.

RNF5: La información y las funcionalidades estarán disponibles según el rol del usuario y se podrá acceder a ella en todo momento.

Disponibilidad.

RNF6: El sistema siempre estará disponible tanto para el Especialista Técnico, como para el Especialista Funcional.

Rendimiento.

RNF7: El subsistema web permitirá que múltiples usuarios estén conectados a la vez.

RNF8: El tiempo de respuesta y procesamiento de la información serán rápidos.

Legales.

RNF9: Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de software libre.

RNF10: La aplicación y toda la documentación generada pertenecen a los departamentos de Ingeniería Informática y Centro de Estudios de Energía y Tecnología Avanzada para la Minería.

Software.

RNF11: Las PC's clientes deben tener instalado el navegador web Google Chrome (10+) o Mozilla Firefox (4+) para el rol de Especialista funcional y para el rol de Especialista Técnico tener instalado además el framework de java.

RNF12: La PC servidor debe contar con servidor web Apache 2.0.

RNF13: La PC servidor debe tener instalado el gestor de base de datos Postgres o MySQL.

Hardware.

Procesamiento:

RNF14: El servidor de aplicaciones requiere de una CPU Intel Pentium o compatible *para su correcto funcionamiento.*

Memoria RAM:

RNF15: El servidor de aplicaciones requiere una RAM de 2 Gb o superior para su correcto funcionamiento.

Capacidad en disco:

RNF16: El servidor de aplicaciones requiere de 80 gigas disponible para su correcto uso

Soporte.

RNF17: Realizar pruebas y mantenimiento necesarios para lograr el mejoramiento y evolución en el tiempo.

Restricciones de diseño e implementación.

1. El servidor debe tener instalado servidor web Apache 2.0.

2. El servidor debe tener instalado el gestor de base de datos Postgres.

3. El subsistema intermedio se desarrollará utilizando como lenguaje de programación Java.

4. El subsistema intermedio se implementara haciendo uso del IDE NetBeans 7.3.

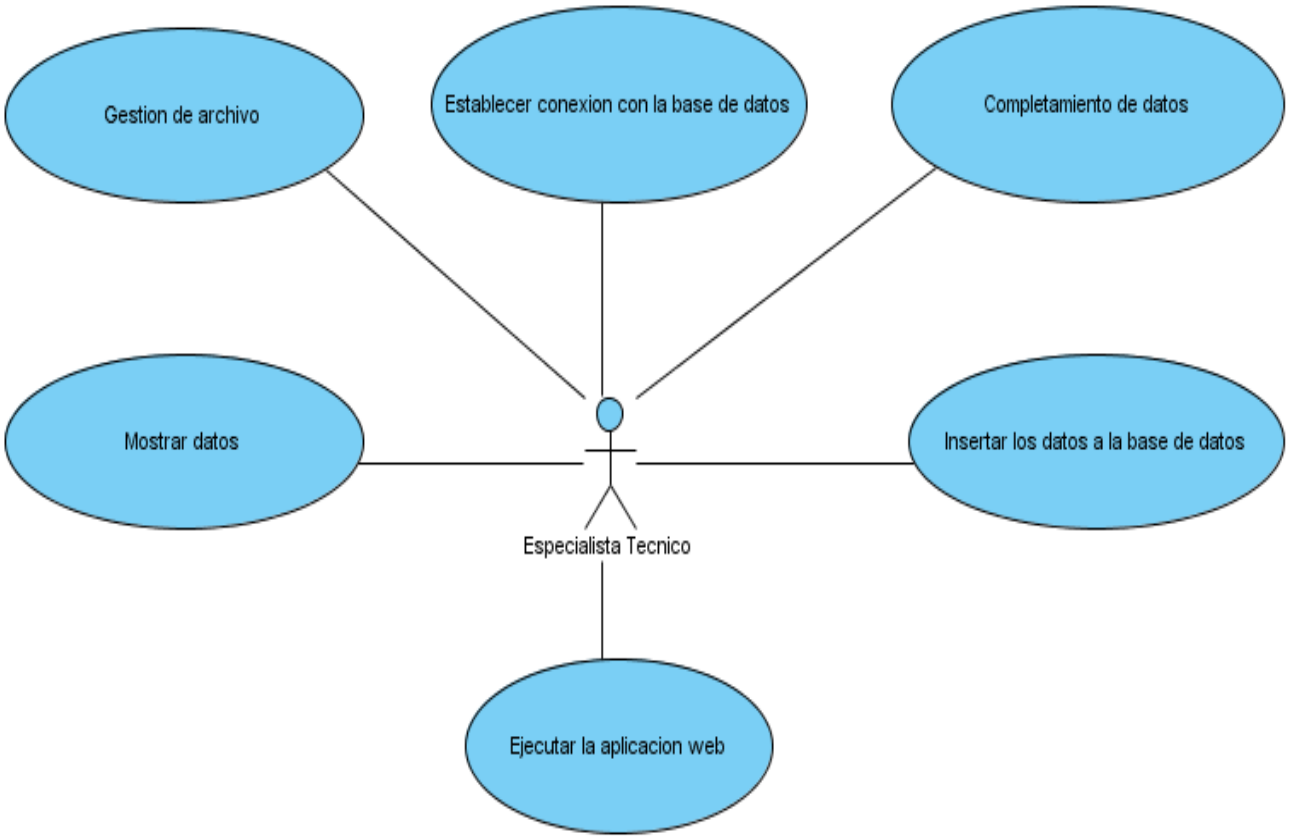
5. El subsistema web se desarrollará utilizando como lenguaje JavaScript.

6. El subsistema web se desarrollará haciendo uso del frameworck ExtJS versión 4.

7. Para la modelación del sistema se utilizará la herramienta Visual Paradigm.

8. La metodología de desarrollo de software empleada será OpenUP, haciendo uso del Lenguaje de Modelación Unificado (UML).

2.4 Casos de Uso del Sistema.



Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. Sus diagramas sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los

usuarios y otros sistemas. Los casos de uso ayudan a describir qué es lo que es sistema debe hacer.

Figura 3. Diagrama de Caso de Uso del Subsistema Intermedio.

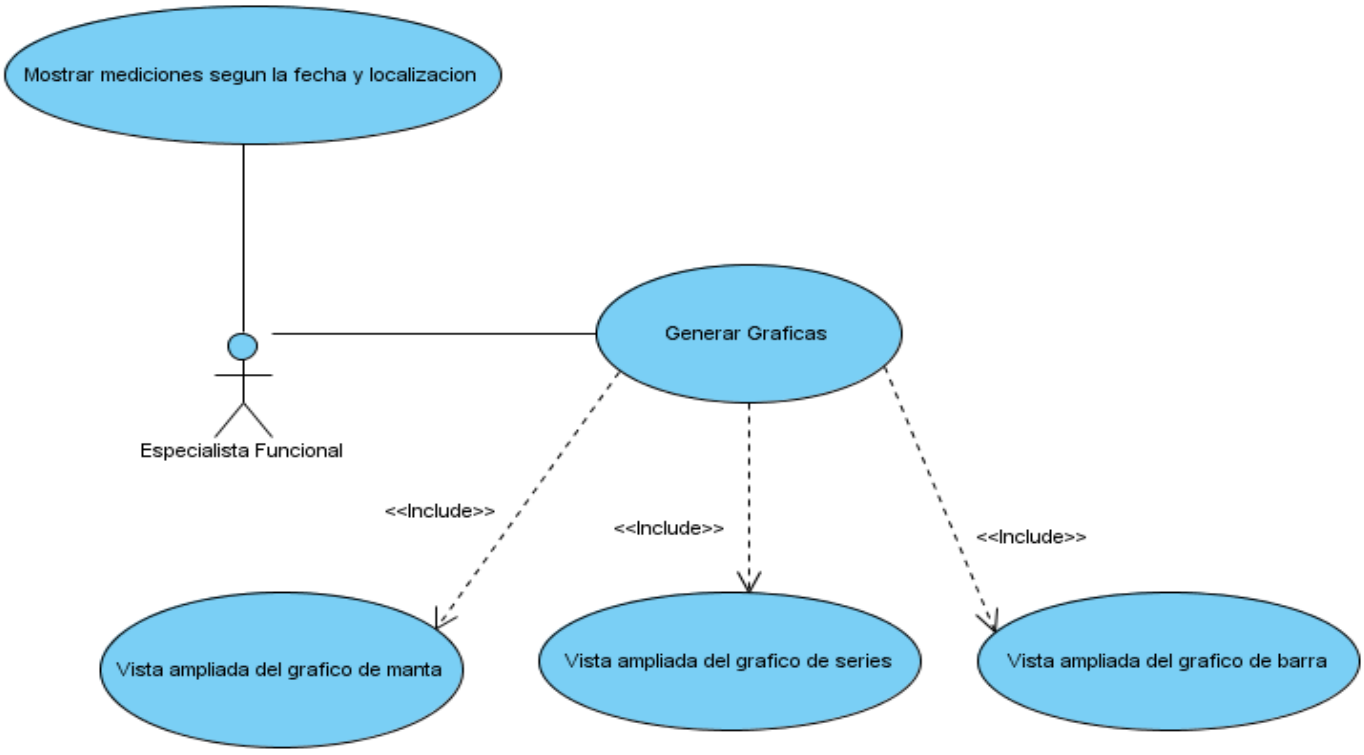


Figura 4. Diagrama de Caso de Uso del Subsistema Intermedio Representación de Datos.

2.5 Descripción de los Casos de Uso.

A continuación se describen textualmente los casos de uso del sistema que fueron modelados en el diagrama anterior, especificando su propósito y sus condiciones de existencia.

CU1: Establecer conexión con la base de datos.

Caso de Uso	Establecer conexión con la base de datos
Objetivo	Permite al Especialista Técnico conectarse a la base de datos del sistema.
Actores	Especialista Técnico.
Resumen	El caso de uso se inicia cuando el Especialista Técnico selecciona las propiedades de la conexión del sistema, en el cual el sistema muestra una ventana para el completamiento de la conexión con la base de datos.
Complejidad	Media
Precondiciones	En Especialista Técnico debe estar autenticado en la base de datos del sistema.
Postcondiciones	El sistema estará conectado a la base de datos.
Referencias	RF1, RF2.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. Selecciona la opción Configurar Conexión con la base de datos	2. Muestra una ventana las cuales tendrá los campos que la aplicación requiere para la conexión de la base de datos: 3. El sistema muestra un mensaje de verificación.

CU2: Gestión de archivo.

Caso de Uso	Gestionar Archivo
Objetivo	Permite buscar en el equipo la ubicación del archivo que contiene los datos arrojados por los equipos de medición.
Actores	Especialista Técnico.
Resumen	El caso de uso inicia cuando el Especialista Técnico selecciona la opción cargar archivo en el que el sistema mostrara un selector de archivo.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Conectarse a la base de datos.
Postcondiciones	Llenar los datos faltantes del archivo.
Referencia	RF3

Flujos Normal de Eventos	
Acción del actor	Respuesta del Sistema
1. Selecciona la opción de Cargar Archivo. 3. El usuario buscará el archivo en el equipo y cuando este lo encuentre le dará al botón aceptar.	2. El sistema mostrará un selector de archivo.

CU3: Completamiento de datos.

Caso de Uso	Completamiento de datos
Objetivo	Completar los datos faltantes de las mediciones.
Actores	Especialista Técnico.
Resumen	El caso de uso inicia cuando el Especialista Técnico acepta el archivo que desea cargar al sistema el cual si presenta datos faltantes en la medición arrojará una ventana con la cantidad de datos faltantes y los campos para llenar estos.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Cargar datos al sistema.
Postcondiciones	Visualizar datos.
Referencia	RF4
Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
	1. Es sistema visualizará una ventana con los datos faltantes en el archivo

	así como los campos para introducir los valores
2. El usuario llenará los campos con los datos faltantes.	3. El sistema validará los datos faltantes.

CU4: Mostrar datos.

Caso de Uso	Mostrar datos
Objetivo	El sistema mostrará los datos.
Actores	Especialista Técnico.
Resumen	El caso de uso inicia cuando el Especialista Técnico hace las operaciones con los datos del archivo cargados
Complejidad	Media
Prioridad	Crítico
Precondiciones	Completamiento de datos
Postcondiciones	Subir los datos a la base de datos
Referencia	RF5
Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
	1. El sistema hará las operaciones con los datos que se arrojaban en el archivo.
2. El usuario visualizará los datos transformados.	

CU5: Insertar los datos a la base de datos.

Caso de Uso	Insertar los datos a la base de datos.
Objetivo	El sistema insertará
Actores	Especialista Técnico.
Resumen	El caso de uso inicia cuando el Especialista Técnico selecciona la opción de Subir los datos a la base de datos
Complejidad	Media
Prioridad	Crítico
Precondiciones	Visualizar datos
Postcondiciones	Ver los datos en el web
Referencia	RF6, RF7, RF8.
Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. El Especialista Técnico selecciona la opción Insertar en la base de datos.	2. El sistema visualizará una ventana con el progreso de la operación.
Opción: "Cancelar Insertar datos en la base de datos"	
3. El Especialista Técnico dará la opción de cancelar la inserción en la base de datos	4. El sistema abortará la operación.

CU6: Ejecutar aplicación web.

Caso de Uso	Ejecutar aplicación web
Objetivo	El sistema se encargará de ejecutar la aplicación web
Actores	Especialista Técnico.
Resumen	El caso de uso inicia cuando el Especialista Técnico selecciona la opción de configurar url del navegador, donde se mostrará una ventana con para introducir la url donde se arroja la aplicación web aceptando, el sistema abrirá el navegador web por defecto del equipo con la url definida por el usuario.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Insertar datos en la base de datos
Postcondiciones	Abrir el navegador web
Referencia	RF9, RF10.

Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. El Especialista Técnico seleccionara la opción de configuración de la url del navegador.	2. El sistema mostrará una ventana en la cual aparecerá el campo para introducir la url que mostrará el navegador web.

	3. El sistema ejecutará el navegador web que tenga definido el equipo por defecto con la url definida.

CU7: Mostrar Mediciones según Fecha y Localización.

Caso de Uso	Mostrar Mediciones según Fecha y Localización.
Objetivo	Organizar los datos en la base de datos según su fecha y localización para su mejor navegación y búsqueda.
Actores	Especialista Funcional.
Resumen	El caso de uso inicia cuando el Especialista Funcional inicia la aplicación web la cual mostrará un árbol jerárquico ordenado cual desplegará una lista de fecha a la cual le pertenecerá una lista de mediciones.
Complejidad	Media.
Prioridad	Crítico.
Precondiciones	
Postcondiciones	Mostrar las mediciones y los gráficos.
Referencia	RF11, RF12, RF13.
Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
	1. El sistema mostrará un árbol con todas las mediciones que se registradas en la base de datos estas estarán ordenadas a su vez por su respectiva fecha.

2. El usuario seleccionará la medición que desee para su estudio	3. El sistema cargará los datos pertenecientes a la medición seleccionada y generara los gráficos de esta

CU8: Generar Gráficas.

Caso de Uso	Generar gráficas.
Objetivo	El sistema generará las gráficas dados los valores sus respectivas mediciones.
Actores	Especialista Funcional.
Resumen	El caso de uso inicia cuando el Especialista Funcional selecciona la medición a la que desea realizar su estudio donde la aplicación generará los siguientes gráficos: • Generar gráfica de serie según altura y velocidad del viento. • Generar gráfica de manta según dirección y altura. • Generar gráfica de barra según potencial eólico y altura.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Seleccionar la medición para su estudio
Postcondiciones	Una mejor visualización de las gráficas generadas por el sistema
Referencia	RF14, RF15, RF16.

Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. El Especialista Funcional seleccionará la medición para su estudio.	2. El sistema generará un conjunto de gráficas compuestas por: • Generar gráfica de serie según altura y velocidad del viento. • Generar gráfica de manta según dirección y altura. • Generar gráfica de barra según potencial eólico y altura.

CU9: Vista ampliada del gráfico de Manta.

Caso de Uso	Vista ampliada del gráfico de Manta.
Objetivo	El sistema ampliará en una nueva ventana el gráfico de Manta según dirección y altura generado por los datos seleccionados.
Actores	Especialista Funcional.
Resumen	El caso de uso inicia cuando el Especialista Funcional selecciona ver detalle ampliado de la gráfica generada visualizándolo en una nueva ventana.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Seleccionar detalle ampliado de la gráfica.

Postcondiciones	Manipular leyenda del gráfico de Manta
Referencia	RF17, RF18.
Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. El Especialista Funcional seleccionará Vista detallada del gráfico.	2. El sistema mostrará en una ventana el gráfico con el gráfico correspondiente a los datos antes seleccionados, mostrando la leyenda de este para su manipulación.

CU10: Vista ampliada del gráfico de Serie.

Caso de Uso	Vista ampliada del gráfico de Serie.
Objetivo	El sistema ampliará en una nueva ventana el gráfico de Serie según altura y velocidad generado por los datos seleccionados.
Actores	Especialista Funcional.
Resumen	El caso de uso inicia cuando el Especialista Funcional selecciona ver detalle ampliado de la gráfica generada visualizándolo en una nueva ventana.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Seleccionar detalle ampliado de la gráfica.

Postcondiciones	Manipular leyenda del gráfico de Serie
Referencia	RF19, RF20.
Flujos Normal de Eventos	
Acción del actor	Respuesta del Sistema
1. El Especialista Funcional seleccionará Vista detallada del gráfico.	2. El sistema mostrará en una ventana el gráfico con el gráfico correspondiente a los datos antes seleccionados, mostrando la leyenda de este para su manipulación.

CU11: Vista ampliada del gráfico de Barra.

Caso de Uso	Vista ampliada del gráfico de Barra.
Objetivo	El sistema ampliará en una nueva ventana el gráfico de Barra según potencial y altura generado por los datos seleccionados.
Actores	Especialista Funcional.
Resumen	El caso de uso inicia cuando el Especialista Funcional selecciona ver detalle ampliado de la gráfica generada visualizándolo en una nueva ventana.
Complejidad	Media
Prioridad	Crítico

Precondiciones	Seleccionar detalle ampliado de la gráfica.
Postcondiciones	Manipular leyenda del gráfico de Barra.
Referencia	RF21, RF22.
Flujos Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. El Especialista Funcional seleccionará Vista detallada del gráfico.	2. El sistema mostrará en una ventana el gráfico con el gráfico correspondiente a los datos antes seleccionados, mostrando la leyenda de este para su manipulación.

2.6 Diagramas de Clases del Diseño.

Representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad; y Relaciones: herencia, Composición, Agregación, Asociación y Uso.

2.6.1 Diagrama de Clases Aplicación Intermedio.

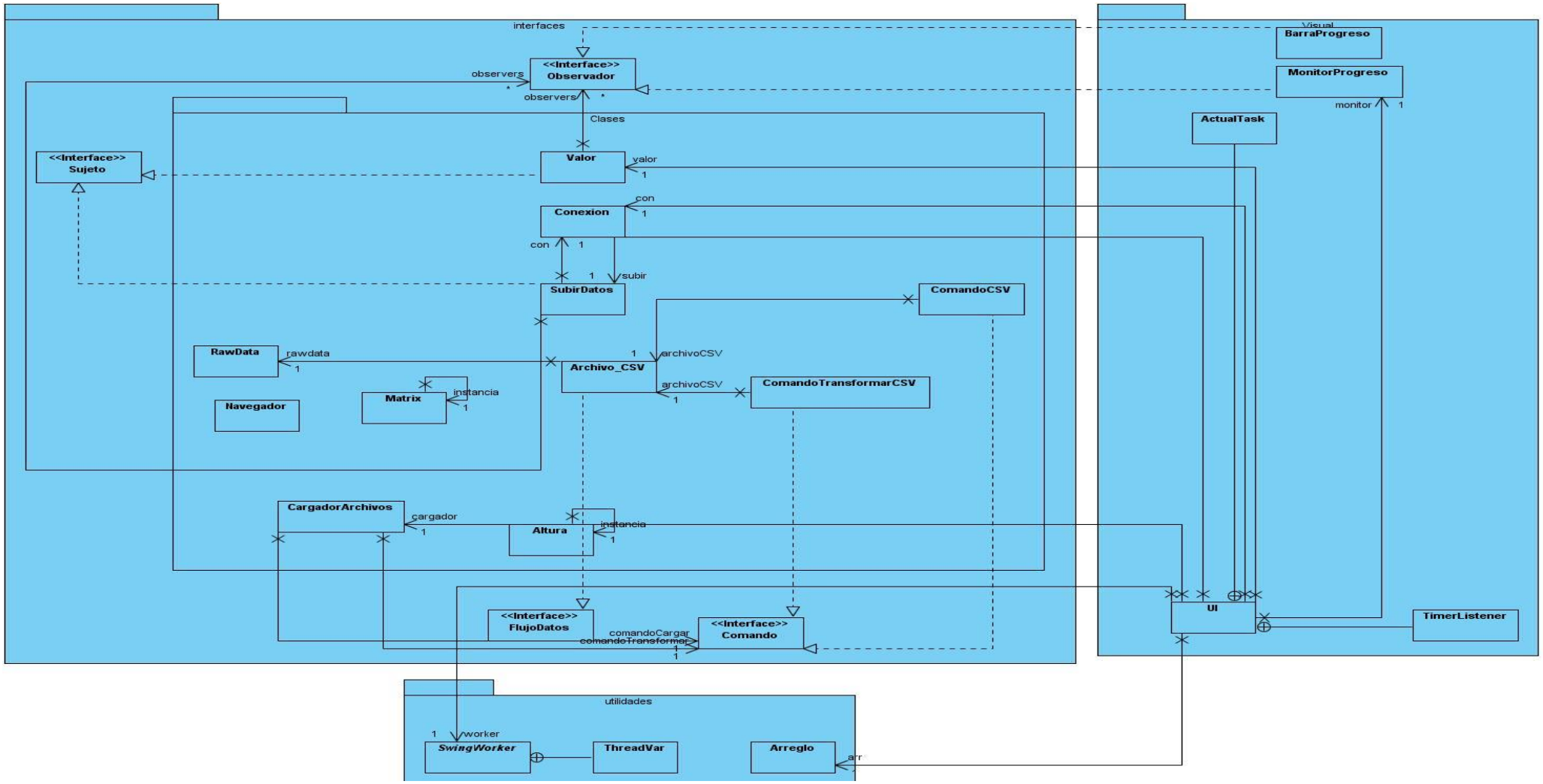


Figura 5. Diagrama de Clases del Subsistema Intermedio.

2.6.2 Diagramas de Clases Aplicación Web.

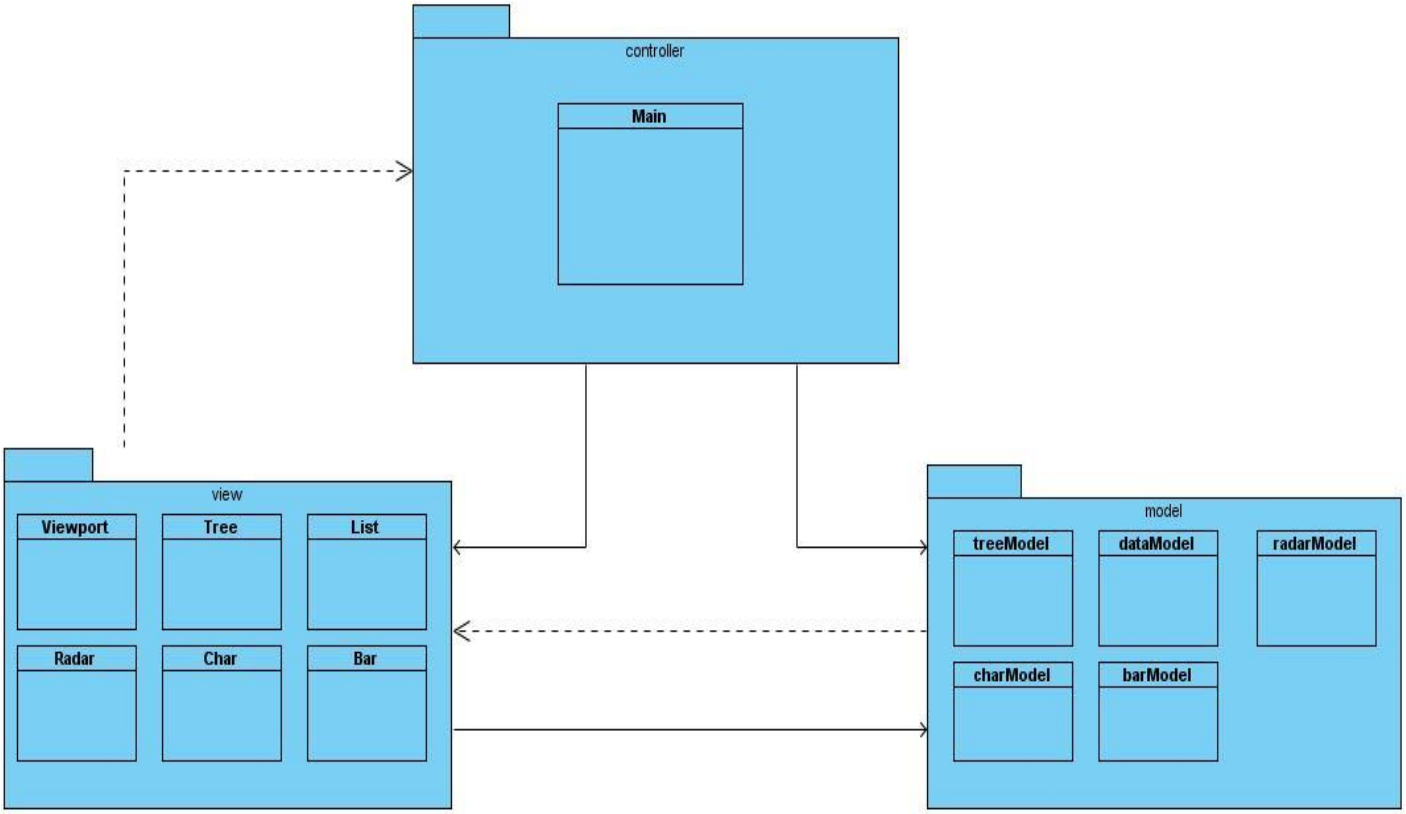


Figura 6. Diagramas de Clase de la Aplicación Web.

2.7 Modelo de Dominio.

El modelo de dominio es una representación visual estática del entorno real objeto del proyecto. Es un diagrama con los objetos que existen (reales) relacionados con el proyecto que se va a llevar a cabo y las relaciones existentes entre ellos.

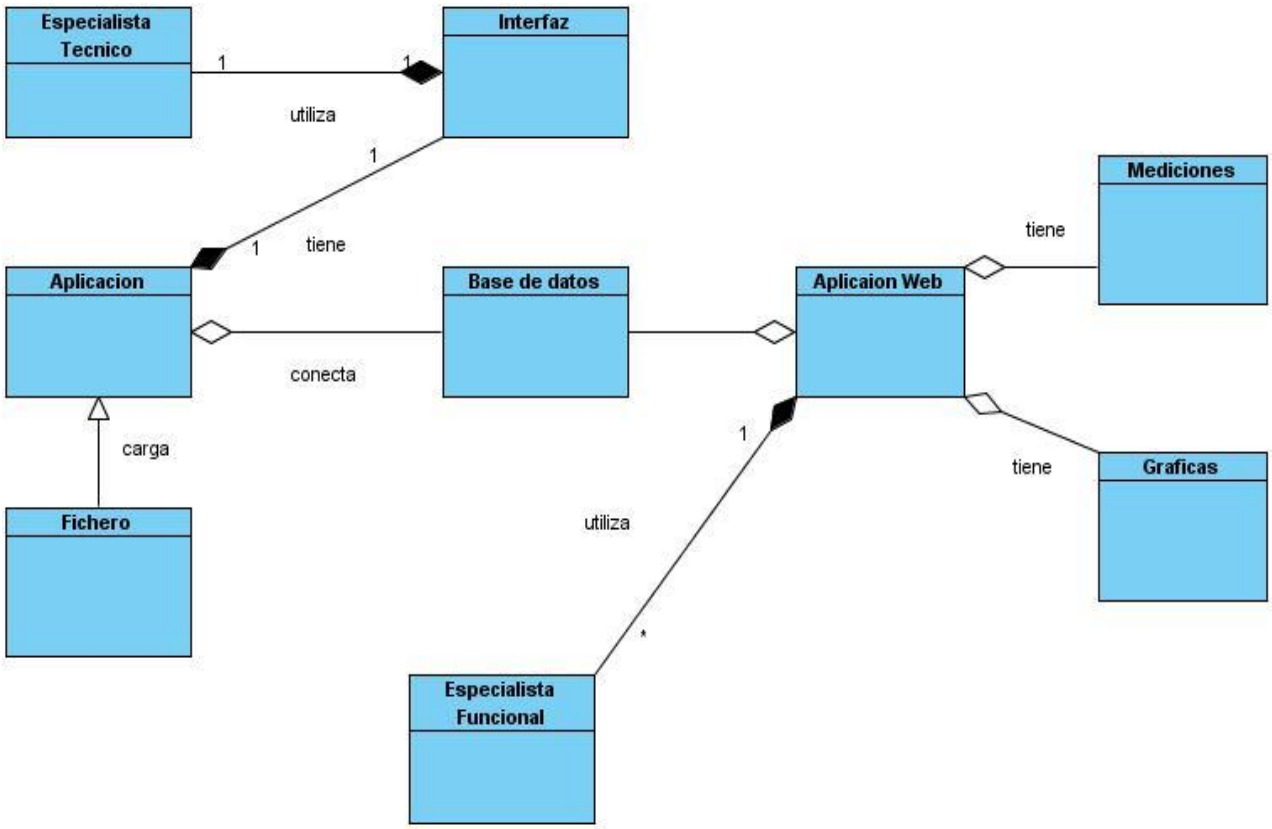


Figura 7. Modelo de Dominio del Sistema

2.8 Diagramas de Componentes.

Estos diagramas representan cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes, pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

Debido a que los diagramas de componentes son más parecidos a los diagramas de casos de usos, éstos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. En él se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

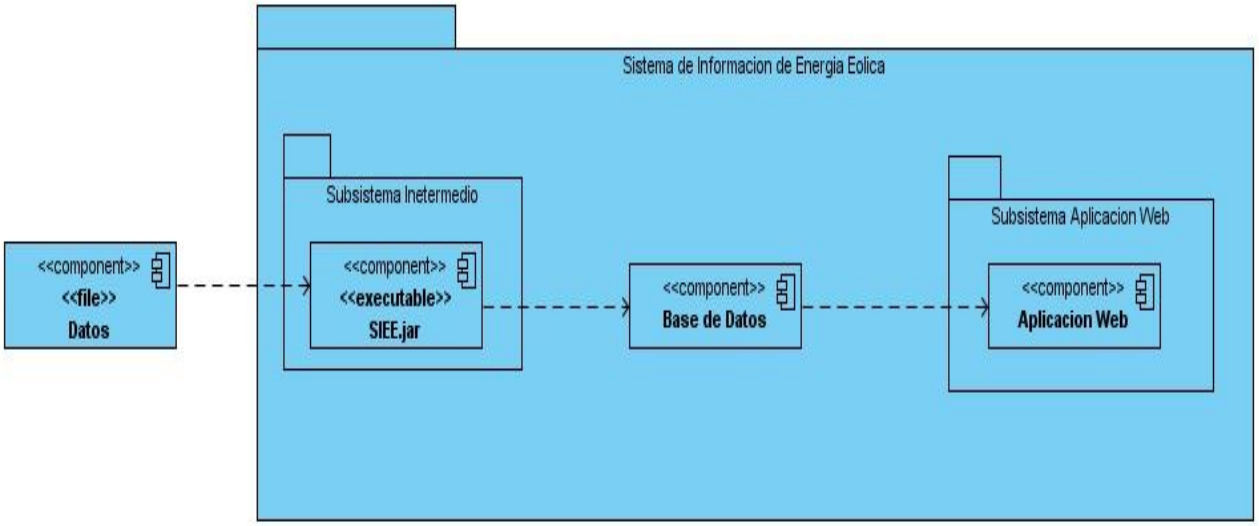


Figura 8. Diagrama de Componentes del Sistema.

2.9 Diagramas de secuencia.

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Mientras que el diagrama de casos de uso permite el modelado de una vista business del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos. Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se dispone de la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede "caminar sobre" ellos para descubrir qué objetos son necesarios para que se puedan seguir los pasos.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales. Existen dos tipos de mensajes: sincrónicos y asincrónicos. Los primeros se corresponden con llamadas a métodos del objeto que recibe el mensaje. El objeto que

envía el mensaje queda bloqueado hasta que termina la llamada. Este tipo de mensajes se representan con flechas con la cabeza llena. Los mensajes asincrónicos terminan inmediatamente, y crean un nuevo hilo de ejecución dentro de la secuencia. Se representan con flechas con la cabeza abierta. También se representa la respuesta a un mensaje con una flecha discontinua.

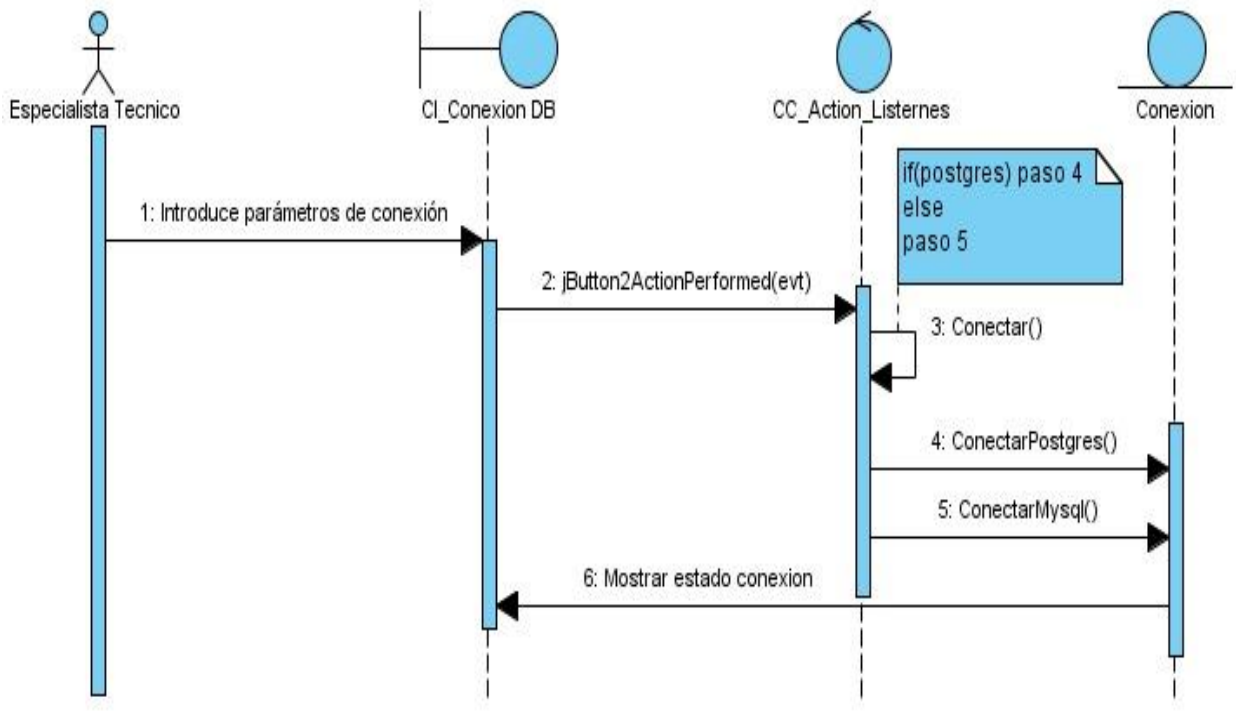


Figura 9. Diagrama de secuencia CU Establecer conexión con la base de datos.

2.10 Conclusiones del capítulo.

En este capítulo se definieron los aspectos relacionados con el análisis y el diseño de la aplicación. Se definieron, además, los requisitos funcionales de la aplicación, así como los no funcionales, en virtud de lograr el buen funcionamiento de la aplicación. Se modelaron algunos diagramas de consideración importante para favorecer una mejor comprensión de las funcionalidades con las que debe contar el sistema.

Capítulo 3. Implementación y realización de pruebas.

3.1 Introducción.

En el presente capítulo se plasman elementos como el Diagrama de Despliegue, mediante el que se representan los principales componentes y sus relaciones. También se describe la implementación de los patrones de diseño utilizados, para lo cual se muestra el código fuente de los principales métodos. Asimismo, se realizan las pruebas que validan la solución propuesta.

3.2 Diagrama de despliegue.

El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos que usa este tipo de diagrama son nodos, componentes y asociaciones.

La mayoría de las veces el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema. Aunque UML no es un lenguaje de especificación hardware de propósito general, se ha diseñado para modelar muchos de los aspectos hardware de un sistema a un nivel suficiente para que un ingeniero software pueda especificar la plataforma sobre la que se ejecuta el software del sistema.

Los elementos usados por este tipo de diagramas son:

Nodos: los elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.

Dispositivos: los nodos son estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Conectores: expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

A continuación se muestra el diagrama de despliegue de la aplicación:

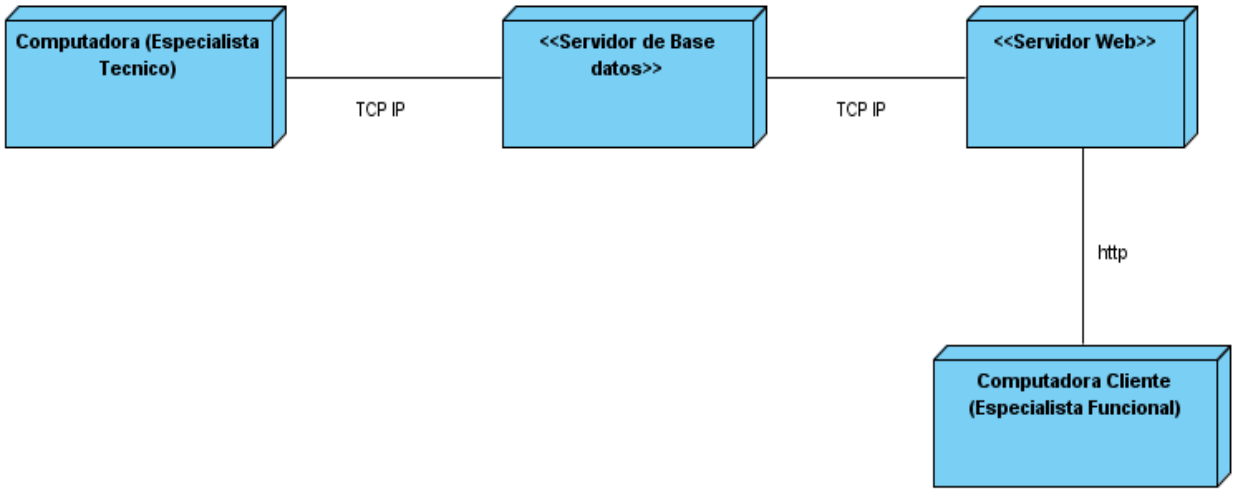


Figura 10.Diagrama de despliegue del sistema.

3.3 Implementación.

3.3.1 Implementación de los patrones de diseño del subsistema intermedio.

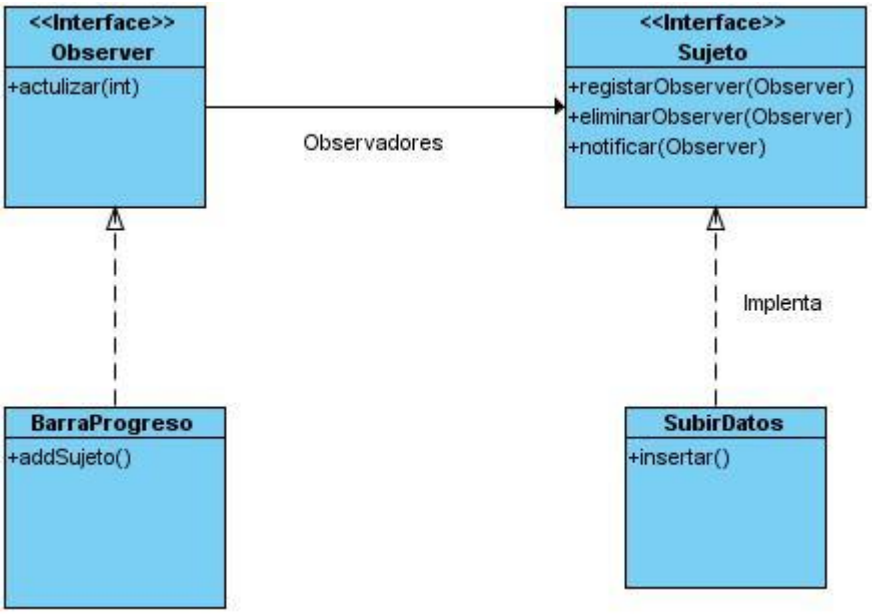


Figura 11. Diagrama de Clase del patrón de diseño Observer.

Implementación de las clases que componen el patrón Observer.

```
package interfaces;

/**
*
* @author Adrian
*/
public interface Observador {
public void actualizar(int dato);
}
```

Código fuente de la Interfaz Observer. La cual define objetos que deben ser notificados.

```
/**
 *
 * @author Adrian
 */
public class BarraProgreso extends JProgressBar implements Observador{

    public BarraProgreso(){
      setStringPainted(true);

    }
    public void addSujeto(Sujeto sujeto){
        sujeto.registrarObserver(this);

    }
    @Override
    public void actualizar(int dato) {
        setValue(dato);

    }

}
```

Código fuente de la clase BarraProgreso la cual implementa la interfaz Observer.

```
package interfaces;

/**
 *
 * @author Adrian
 */
public interface Sujeto {
    public void registrarObserver(Observador o);
    public void eliminarObserver(Observador o);
    public void notificar();
}
```

Código fuente de la Interfaz Sujeto. La cual proporciona métodos para suscribir y borrar Observers.

```
public void insertar()throws SQLException{
    PreparedStatement statement=null;
    int idTabla=0;
    String nombreTB;
    // buscar si existe una medicion en la bd..
    String selectLastId_sql="select id_tabla from selector";
    statement = (PreparedStatement)
Conexion.con.prepareStatement(selectLastId_sql);
    ResultSet result=statement.executeQuery();
    boolean noVacio=result.last();
    if(noVacio) idTabla=result.getInt("id_tabla");
    statement.close();
    idTabla++;
    //construir el nombre de la nueva tabla apartir del ultimo registro si
existe....
    nombreTB="tabla"+idTabla;
    insertarNodo(idTabla);
    //crear tabla
    String createTable_sql = "create table " + nombreTB + " (id VARCHAR(20),
fecha VARCHAR(20), hora VARCHAR(20), altura VARCHAR(20),  velocidad
VARCHAR(20), direccion VARCHAR(20));";
    statement = (PreparedStatement)
Conexion.con.prepareStatement(createTable_sql);
    statement.executeUpdate();

    //insertar la medición en la tabla creada
    String sentencia_sql ="Insert into " + nombreTB +
"(id,fecha,hora,altura,velocidad,direccion) values(?,?,?,?,?,?);";
    statement = (PreparedStatement)
Conexion.con.prepareStatement(sentencia_sql);

       for ( ; i < datos.length; i++) {
               statement.setString(1, datos[i][0]);
               statement.setString(2, datos[i][1]);
               statement.setString(3, datos[i][2]);
               statement.setString(4, datos[i][3]);
               statement.setString(5, datos[i][4]);
               statement.setString(6, datos[i][5]);
               statement.executeUpdate();
               valor++;
               notificar();//notificar a los observer...

       }
    statement.close();

}
```

Código fuente del método insertar de la clase SubirDatos. Donde la cual implementa los métodos relacionados con la interfaz Sujeto.

Implementación de las clases que componen el patrón Command.

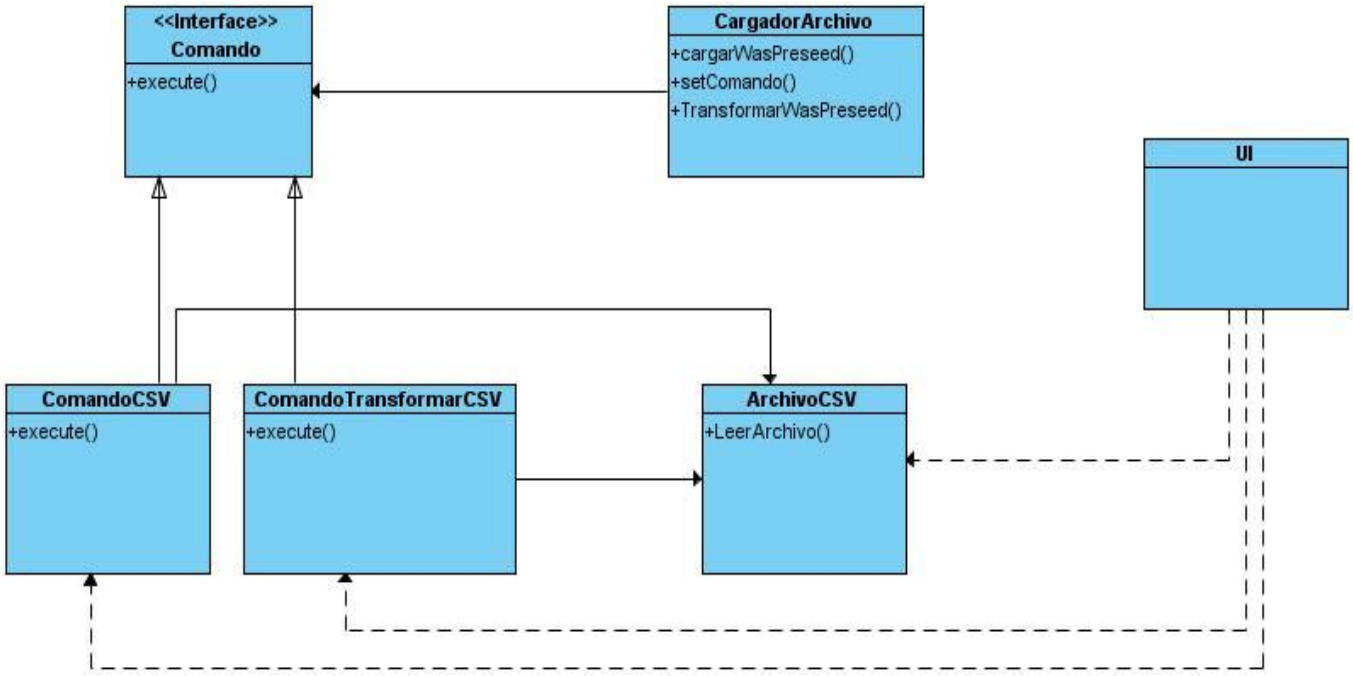


Figura 12. Diagrama de Clases del patrón de diseño Command

```
/**
 *
 * @author Adrian
 */
public class ComandoCSV implements Comando{

    Archivo_CSV archivoCSV;

    public ComandoCSV(Archivo_CSV archivoCSV) {
        this.archivoCSV = archivoCSV;
    }
    @Override
    public void execute() throws IOException{
        archivoCSV.leerArchivo();

        archivoCSV.transformarMatrix();
        //return archivoCSV.getDatos();
    }
}
```

Código fuente de la Clase Comando CSV. Donde se implementa el método execute el cual ejecuta la operación del objeto archivoCSV.

```
 * @author Adrian
 */
public class CargadorArchivos {
    Comando comandoCargar;
    Comando comandoTransformar;

    public void setComando(Comando comandoCargar,Comando comandoTransformar
){
        this.comandoCargar=comandoCargar;
        this.comandoTransformar=comandoTransformar;
    }

    public void cargarWasPreseed() throws IOException{
          comandoCargar.execute();
    }
    public void transformarWasPreseed()throws IOException{
        comandoTransformar.execute();
    }

}
```

Código fuente de la clase CargadorArcivos. La contiene la clase Comando que es la interfaz para la ejecución de una operación en este caso la carga y trasforma el archivo.

```
String direccion=selectedFile.getParent()+"/"+selectedFile.getName();
                Archivo_CSV archivo_CSV=new Archivo_CSV(direccion);
                ComandoCSV comandoCSV=new ComandoCSV(archivo_CSV);
                ComandoTransformarCSV comandoTransformarCSV=new
ComandoTransformarCSV(archivo_CSV);
                cargador=new CargadorArchivos();
                cargador.setComando(comandoCSV,comandoTransformarCSV);
                cargador.cargarWasPreseed();
                jlcant_alturas.setText( "Debe definir el valor de "+

Integer.toString(Altura.getInstancia().getCantidad())+
                        " alturas");
```

Fragmento de código de la clase UI en el cual se evidencia el uso del patrón Command a la hora los datos del archivo.

3.3.2 Implementación del patrón de diseño MVC del subsistema representación de datos.

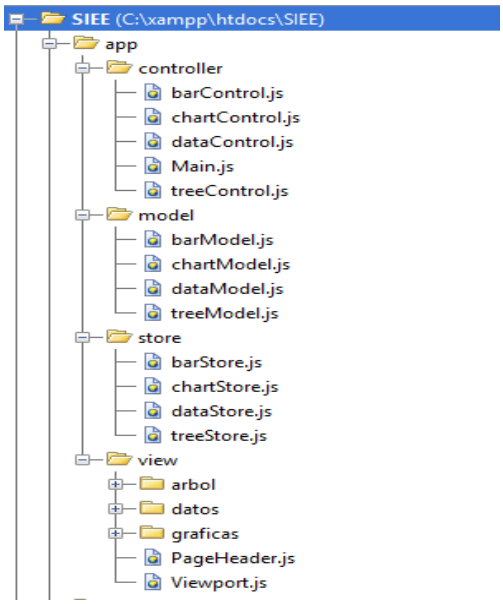


Figura 13. Representación de cómo deben ir distribuidas las clases en ExtJS para la implementación del patrón MVC.

Implementación del caso de uso (Mostrar Mediciones según Fecha y Localización).

```
Ext.define('SIEE.view.arbol.Arbol' ,{
    extend: 'Ext.tree.Panel',
    renderTo: Ext.getBody(),

    alias: 'widget.treeView',

    store : 'treeStore',
    autoScroll:true,
    title: 'Lista de Mediciones',
    titleAlign:"center",
    width: 250
});
```

Implementación de la clase Árbol la cual hace la vista del árbol para visualizar las mediciones guardadas en la base de datos.

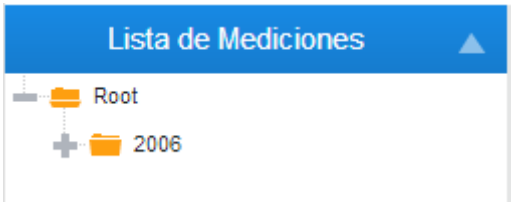


Figura 14. Vista del árbol implementado.

```
/**
 * Created with JetBrains WebStorm.
 * User: Adrian
 * Date: 6/03/13
 * Time: 11:19
 * To change this template use File | Settings | File Templates.
 */

Ext.define('SIEE.view.datos.Lista' ,{
    extend: 'Ext.grid.Panel',

    alias: 'widget.dataList',

    title: 'MEDICIONES',
    titleAlign:"center",
    store: 'dataStore',
    features:[{ftype:'grouping'}],

    initComponent: function() {
            this.columns = [
                {header: 'Nombre',  dataIndex: 'nombre',  flex: 1},
                {header: 'FECHA', dataIndex: 'fecha', flex: 1},
                {header: 'HORA', dataIndex: 'hora', flex: 1},
                {header: 'ALTURA', dataIndex: 'altura', flex: 1},
                {header: 'VELOCIDAD', dataIndex: 'velocidad', flex: 1},
                {header: 'DIRECCION', dataIndex: 'direccion', flex: 1},

            ];
            this.callParent(arguments);
        }
});
```

Implementación de la clase lista la cual construye un grid para la visualización de los datos.

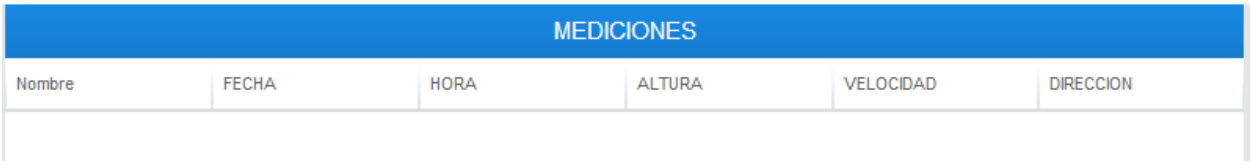


Figura 15. Vista del grid.

```
Ext.define('SIEE.model.treeModel', {
    extend: 'Ext.data.Model',

    fields: [
        { name: 'id', type: 'int' },
        { name: 'anho', type: 'string' },
        { name: 'idParent', type: 'int' },
        { name: 'text', type:'string' },
        { name: 'idTabla', type: 'int'}
    ],

    proxy: {
        type: 'ajax',
        url: 'data/tree.php'
    }
});
```

Implementación de la clase Modelo treeModel. La cual construye el modelo del árbol.

```
/**
 * Created with JetBrains WebStorm.
 * User: Adrian
 * Date: 6/03/13
 * Time: 12:13
 * To change this template use File | Settings | File Templates.
 */
Ext.define('SIEE.model.dataModel', {
    extend: 'Ext.data.Model',

    fields: ['nombre', 'fecha','hora','altura','velocidad','direccion']
});
```

Implementación de la clase Modelo dataModel. La cual construye el modelo del grid.

```
Ext.define('SIEE.controller.Main', {
    extend: 'Ext.app.Controller',

    views:
['Viewport','PageHeader','graficas.Bar','graficas.Chart','graficas.Radar','da
tos.Lista','arbol.Arbol'],
    models: ['barModel','chartModel','dataModel','treeModel','radarModel'],
    stores: ['chartStore','barStore','dataStore','treeStore','radarStore'],


    init: function() {
        this.control({
            'treeView': {
                select: this.cambiarData
            }
        });
    },
```

```
    cambiarData: function(store, record) {
        var d = new Number(record.get('idTabla'))
        console.log('Double clicked on ' + d);
        Ext.getStore('dataStore').load(({params:{d:d}}));
        Ext.getStore('radarStore').load(({params:{d:d}}));
    }
});
```

Implementación de la clase controladora de nuestra aplicación web. En la cual se implementa el método que realiza la acción de cargar los datos en dependencia del nodo señalado en el árbol.

3.4 Validación funcional.

Durante todo el ciclo de elaboración del software es preciso velar, controlar y garantizar su correcta calidad, haciendo posible el cumplimiento de los requerimientos que precisamente satisfacen las necesidades del cliente. Este aspecto debe estar presente de forma paralela desde la concepción del producto hasta la fase de producción del mismo. Para verificar lo antes mencionado se recurre a la realización de las pruebas de software.

3.5 Pruebas de software.

Las pruebas de software es un concepto que a menudo es conocido como verificación y validación. Integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. Entre algunas de las técnicas que se llevan a cabo para el proceso de prueba se encuentran las técnicas de caja negra y de caja blanca. (Fernández E. , 2011)

3.5.1 Pruebas de caja negra.

Prueba de caja negra es aquel elemento que se estudia desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. En otras palabras, de una caja negra solamente interesará su forma de interactuar con el medio que le rodea (en ocasiones, otros elementos que también podrían ser cajas negras) entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace. Por tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento.

Cuando se habla de un software, la prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del mismo. Los métodos de prueba de la caja negra se centran en los requisitos funcionales del mismo e intentan encontrar errores de las siguientes categorías: (Fernández E. , 2011).

1. Funciones incorrectas o ausentes.

2. Errores de interfaz.

3. Errores en estructuras de datos o en acceso a bases de datos externas

4. Errores de rendimiento.

5. Errores de inicialización y terminación.

3.5.2 Pruebas del subsistema Intermedio.

Figura 16. Ejecución de la aplicación Intermedia. En este caso con la ventana emergente la cual pide los datos necesarios para la conexión con la base de datos.

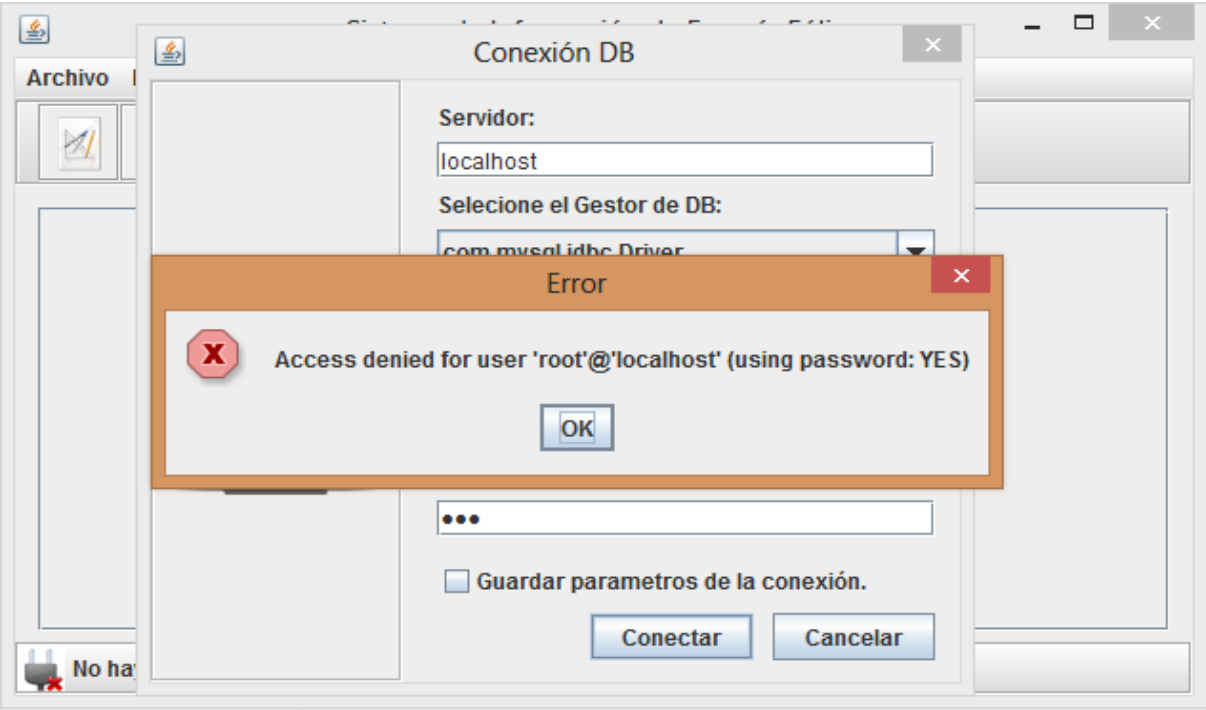


Figura 17.Error a la hora de conectarse a la base de datos. Mensaje de error si los datos a la base de datos son erróneos.

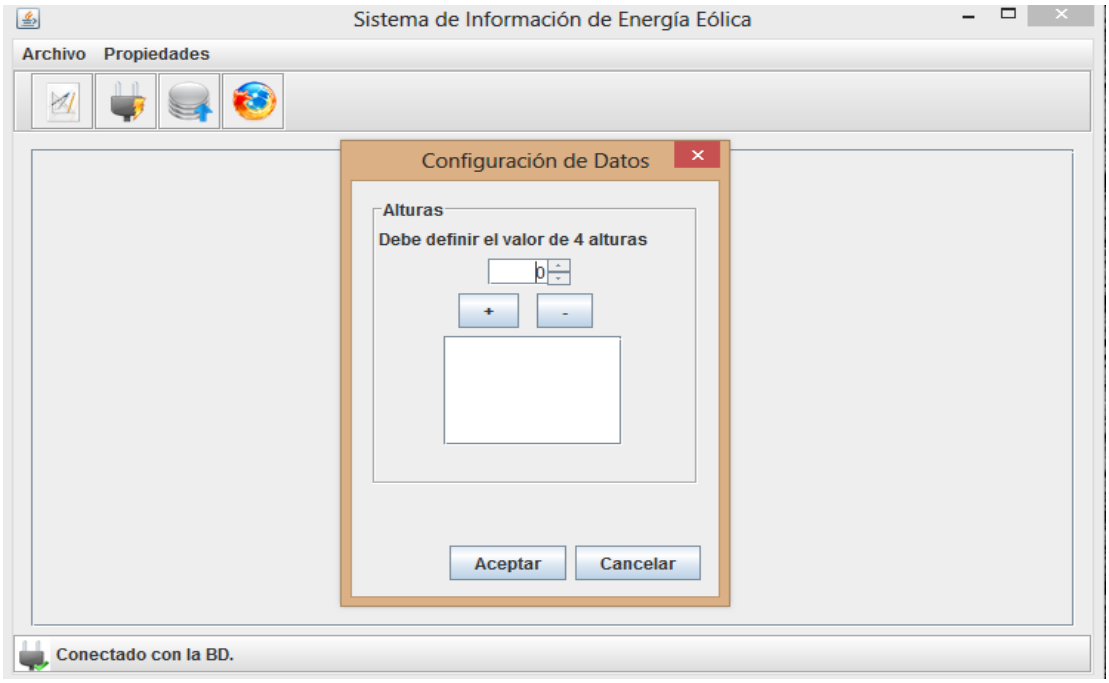
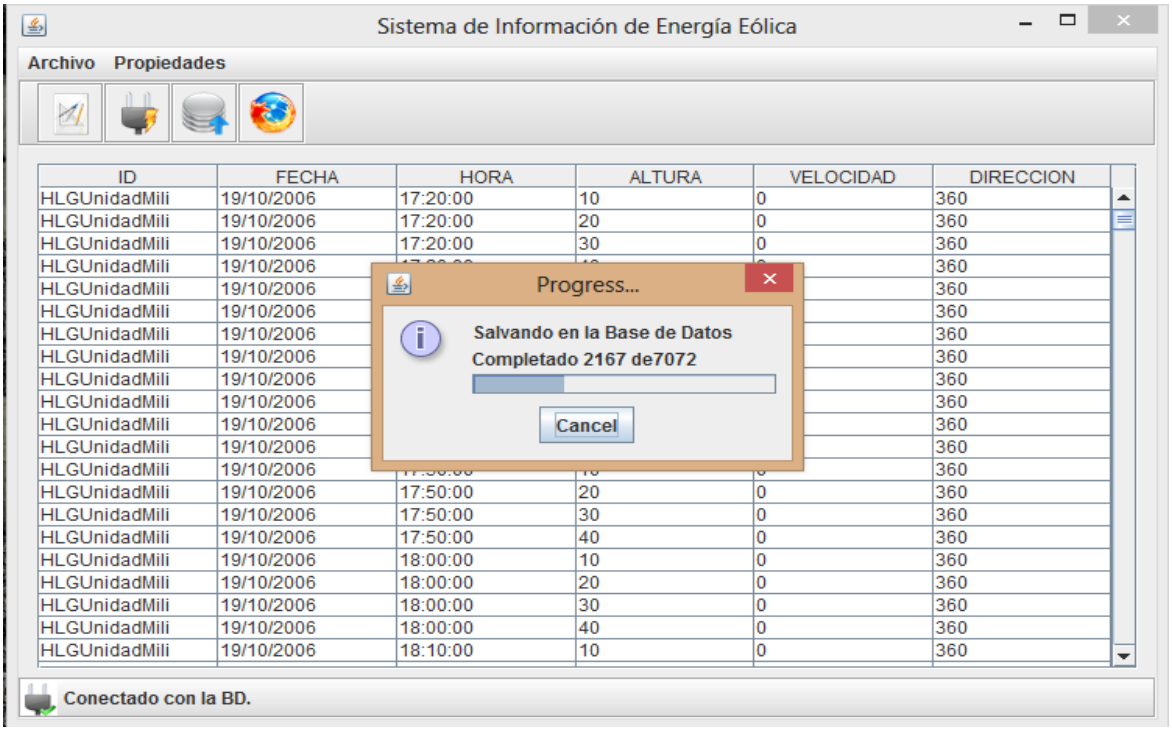


Figura 18. Muestra la ventana emergente la cual pide los datos faltantes en el archivo que será cargado.



ID	FECHA	HORA	ALTURA	VELOCIDAD	DIRECCION
HLGUnidadMili	19/10/2006	17:20:00	10	0	360
HLGUnidadMili	19/10/2006	17:20:00	20	0	360
HLGUnidadMili	19/10/2006	17:20:00	30	0	360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006				360
HLGUnidadMili	19/10/2006	17:50:00	20	0	360
HLGUnidadMili	19/10/2006	17:50:00	30	0	360
HLGUnidadMili	19/10/2006	17:50:00	40	0	360
HLGUnidadMili	19/10/2006	18:00:00	10	0	360
HLGUnidadMili	19/10/2006	18:00:00	20	0	360
HLGUnidadMili	19/10/2006	18:00:00	30	0	360
HLGUnidadMili	19/10/2006	18:00:00	40	0	360
HLGUnidadMili	19/10/2006	18:10:00	10	0	360

Figura 19. Aplicación Intermedia subiendo los datos en la base de datos.

3.5.2 Pruebas del subsistema aplicación web.

Figura 20. Aplicación web ejecutándose

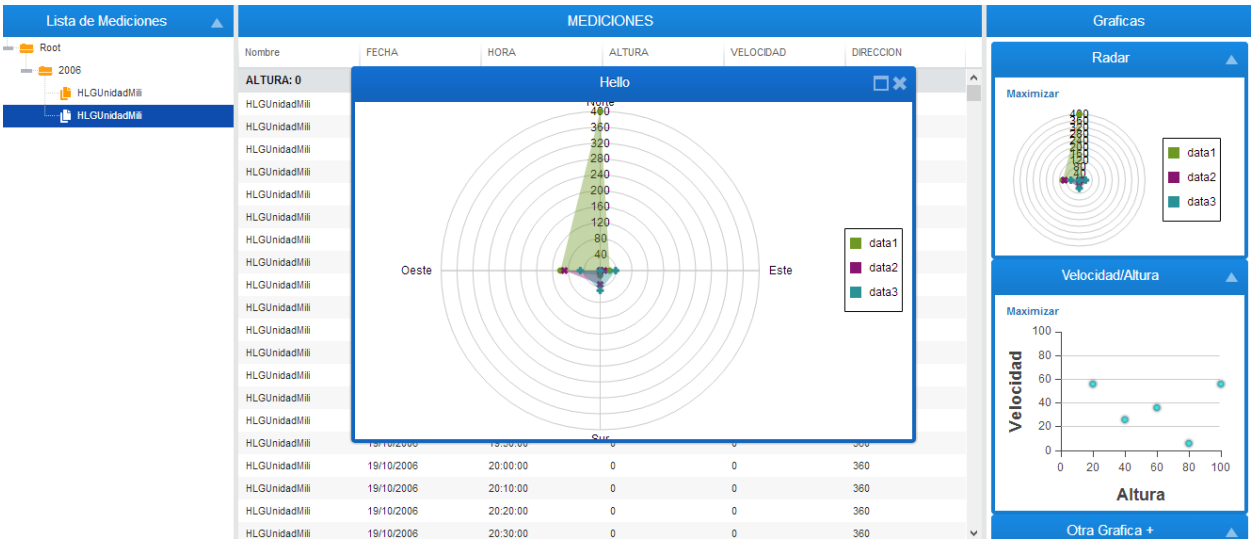


Figura 21. Mostrando la vista ampliada de la gráfica de manta generado por los datos señalados por el usuario.

3.6 Conclusiones del capítulo.

En el presente capítulo se realizó la implementación del sistema, para ello se llevó a cabo el diseño de los principales componentes y sus relaciones, mediante la modelación del Diagrama de Despliegue. Posteriormente se mostró el código de algunos métodos de la aplicación. Para realizar la validación del sistema se definió usar la técnica de caja negra la cual fue aplicada en el diseño de pruebas, mostrando así las entradas y salidas válidas de la aplicación según los requerimientos.

Capítulo 4. Estudio de Factibilidad.

Introducción.

Después de definir la problemática presente e identificar las causas que ameritan la informatización de estos procesos, es pertinente realizar un estudio de factibilidad para determinar la infraestructura tecnológica y la capacidad técnica que implica la implantación del sistema en cuestión, así como los costos, beneficios y el grado de aceptación que la propuesta genera en la Institución.

4.1 Factibilidad Técnica.

La Factibilidad Técnica consiste en realizar una evaluación de la tecnología existente en la organización. Este estudio estuvo destinado a recolectar información sobre los componentes técnicos que la organización posee y la posibilidad de hacer uso de los mismos en el desarrollo e implementación del sistema propuesto y de ser necesario, los requerimientos tecnológicos que deben ser adquiridos para el desarrollo y puesta en marcha del sistema en cuestión. De acuerdo a la tecnología necesaria para la implantación del Sistema de Gestión de Información de los Parámetros de Vientos en los Parques Eólicos (SIEE), se evaluó bajo dos enfoques: Hardware y Software.

4.1.1 Hardware

El servidor donde debe estar instalado el sistema propuesto, debe cubrir con los siguientes requerimientos mínimos:

- ✓ Procesador Pentium 1.5 Ghz.
- ✓ Tarjeta Madre: P3
- ✓ 512 MB de Memoria RAM
- ✓ Disco Duro de 5 GB.
- ✓ Unidad de Protección UPS.

Evaluando el hardware existente y tomando en cuenta la configuración mínima necesaria, no se requirió realizar inversión inicial para la adquisición de nuevos equipos, ni tampoco para mejorar o actualizar los equipos existentes. A continuación se muestran

las características de red interna con que cuenta actualmente el Centro de Estudio de la Energía y Tecnologías de Avanzada para la Minería (CEETAM):

- ✓ Servidor: Profesional HP Proliant ML 350, 2.8 Ghz de velocidad y 2GB RAM.
- ✓ Las estaciones de Trabajo: Procesador Pentium 4+, 1+GB en Memoria RAM, Disco Duro 160+GB.
- ✓ Concentradores de Puertos RJ-45.

Todas las estaciones de trabajo están conectadas al servidor a través de una red utilizando cable par trenzado. Esta configuración permite que los equipos instalados puedan interactuar con el SIEE.

4.1.2 Software.

El CEETAM cuenta con todas las aplicaciones que se emplearon para el desarrollo del proyecto y funcionamiento del sistema, lo cual no amerita inversión alguna para la adquisición de los mismos. Las estaciones de trabajo, operarán en ambientes MS Windows y GNU/Linux, el servidor se encuentra instalado sobre una plataforma GNU/Linux. Para el uso general de las estaciones en actividades diversas se debe poseer las herramientas y los navegadores que existen en el mercado actualmente.

Propiedades	Nombre
Sistemas Operativos	GNU/Linux, Microsoft Windows.
Navegador Web	Mozilla Firefox o Google Chrome.
Herramientas de escritorio	Libre Office, Microsoft Excel.

Como resultado de este estudio técnico se determinó que la Institución posee la infraestructura tecnológica (Hardware y Software) necesaria para el desarrollo y puesta en funcionamiento del sistema propuesto.

4.2 Factibilidad Económica.

4.2.1 Evaluación de Costo-Beneficio.

Para estudiar la factibilidad de este proyecto se utilizará la Metodología Costo Efectividad (Beneficio), la cual plantea que la conveniencia de la ejecución de un proyecto se determina por la observación conjunta de dos factores:

1. El costo, que involucra la implementación de la solución informática, adquisición y puesta en marcha del sistema hardware/software y los costos de operación asociados.
2. La efectividad, que se entiende como la capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo para el cual se ideó, es decir, un proyecto será más o menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad de cumplimiento del objetivo). Este puede estar justificado por los beneficios tanto tangibles como intangibles que origina el mismo. En este proceso, se necesita de una selección adecuada de los elementos más convenientes para su evaluación.

4.2.2 Efectos Económicos.

Pueden clasificarse como:

- Efectos directos.
- Efectos indirectos.
- Efectos externos.
- Intangibles.

4.2.2.1 Efectos directos.

Positivos:

- Se facilitará a través del sistema una mejor comprensión de los datos arrojados por los equipos de medición de los parámetros de viento en los parques eólicos.
- Los datos estarán almacenados por un orden según la fecha en una base de datos, lo que facilitará un mejor manejo de los mismos.

- Por el tipo distribución que presenta el sistema permite al Especialista Técnico tomar distintas mediciones en diferentes lugares para su análisis. Por su parte el Especialista Funcional del sistema puede acceder desde cualquier lugar mediante un navegador web al subsistema que permite el análisis de los datos.

Negativos:

Para usar la aplicación es vital el uso de un ordenador conectado a la red, aparejado a los gastos de consumo de energía eléctrica.

4.2.2.2 Efecto Indirecto.

Los efectos económicos observados que pudiera repercutir sobre otros mercados no son perceptibles, aunque este proyecto no está construido con la finalidad de comercializarse.

4.2.2.3 Externalidades.

Se contará con una herramienta disponible que facilitará el estudio de los parámetros de viento en los parques eólicos, optimizando el tiempo y recursos.

4.2.2.4 Intangibles.

En la valoración económica siempre hay elementos perceptibles por una comunidad como perjuicio o beneficio, pero al momento de ponderar en unidades monetarias esto resulta difícil o prácticamente imposible. A fin de medir con precisión los efectos, deberán considerarse dos situaciones:

Situación sin Proyecto:

Se describe esta situación desde la perspectiva de los dos actores fundamentales:

Especialista técnico:

Colecta los datos arrojados por los equipos de medición en hojas de MS Excel, sobre estas hojas desarrollan los cálculos necesarios para transformar y obtener información sobre el comportamiento del viento. El intercambio de estos datos, con personas interesadas (otros especialistas, directivos, personal científico), se realiza de forma

manual mediante la comunicación persona-persona utilizando dispositivos extraíbles (Memorias Flash, CD, etc.).

Especialista funcional:

Tiene un limitado acceso a los datos de los parámetros de viento, fundamentalmente porque no cuenta con un "punto de acceso" en el que se distribuyan. Para solicitar estos datos, o alguna información relacionada, debe comunicarse con los especialistas técnicos utilizando algún canal de comunicación (email, teléfono, etc.) con todas las dificultades que desde el punto de vista tanto de disponibilidad como seguridad esto conlleva.

Situación con Proyecto:

Especialista técnico:

Es el encargado de almacenar los datos, arrojados por los equipos en hojas de MS Excel, en una base de datos que puede ser accedida por la red. Las personas interesadas en estos datos, con los permisos adecuados, pueden acceder en cualquier momento. El especialista dispone de estos datos con garantías de seguridad.

Especialista funcional:

Dispone de un "punto de entrada" en el que puede encontrar información sobre el comportamiento del viento, en forma de tablas y gráficos, con garantías de seguridad.

4.2.3 Beneficios y Costos Intangibles en el proyecto.

Costos:

- ✓ Resistencia al cambio.

Beneficios:

- ✓ Mayor comodidad para los usuarios.
- ✓ Mayor información visual sobre los datos arrojados por los equipos de medición de los parques eólicos.
- ✓ Mejora la calidad del estudio de los parámetros de viento en los parques eólicos.

- ✓ Posibilita reducir el tiempo de atención y realización de los estudios de los parámetros de viento en los parques eólicos.
- ✓ Reduce el gasto de materiales de oficina utilizados en estos procesos.

4.2.4 Ficha de Costo.

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar Una Ficha De Costo de un Producto Informático [Dra. Ana María Gracia Pérez, UCLV]. Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional.

Costo en Moneda Libremente Convertible:

Costos Directos:

1. Compra de equipos de cómputo: No procede.
2. Alquiler de equipos de cómputo: No procede.
3. Compra de licencia de Software: No procede.
4. Materiales directos: $7.00.
5. Gasto por consumo de energía eléctrica: $ 15.00 (por cinco meses Nota: Este valor es un número aproximado, debido a que es imposible proporcionar un valor exacto por medirse el consumo en el Instituto Minero Metalúrgico de Moa de forma general).

Subtotal: $ 22.00

Costos Indirectos.

1. Formación del personal que elabora el proyecto: No procede.
2. Gastos en llamadas telefónicas: No procede.
3. Gastos para el mantenimiento del centro: No procede.
4. Know How: No procede.
5. Gastos en representación: No procede.

Subtotal: $0.00.

Gastos de distribución y venta.

1. Participación en ferias o exposiciones: No procede.
2. Gastos en transportación: No procede.
3. Compra de materiales de propagandas: No procede.

Subtotal: $0.00.

Total de Costo en Moneda Libremente Convertible: $22.00.

Costo en Moneda Nacional.

Costos Directos.

1. Salario del personal que laborará en el proyecto: $100.00 ($500.00 por 5 meses de trabajo).
2. El 5% del total de gastos por salarios se dedica a la seguridad social: No procede.
3. El 0.09% de salario total, por concepto de vacaciones a acumular: No procede.
4. Gasto por consumo de energía eléctrica: No procede.
5. Gastos en llamadas telefónicas: No procede.
6. Gastos administrativos: No procede.

Costos Indirectos.

1. Know How: No procede.

Subtotal: $ 0.00

Gasto en Distribución y Ventas Subtotal: $ 0.00

Total de Costo en Moneda Nacional: $ 500.00

La evaluación económica se efectúa conjuntamente con evaluación técnica del proyecto, que consiste en cerciorarse de la factibilidad técnica del mismo. En el análisis de la Factibilidad Técnica del proyecto, se pudo apreciar que se cuenta con la disponibilidad de hardware/software por lo que se puede inferir que el proyecto es factible técnicamente y no necesita de inversión alguna para su realización, por tanto la decisión de inversión recae en la evaluación económica. Como se hizo referencia anteriormente, la técnica

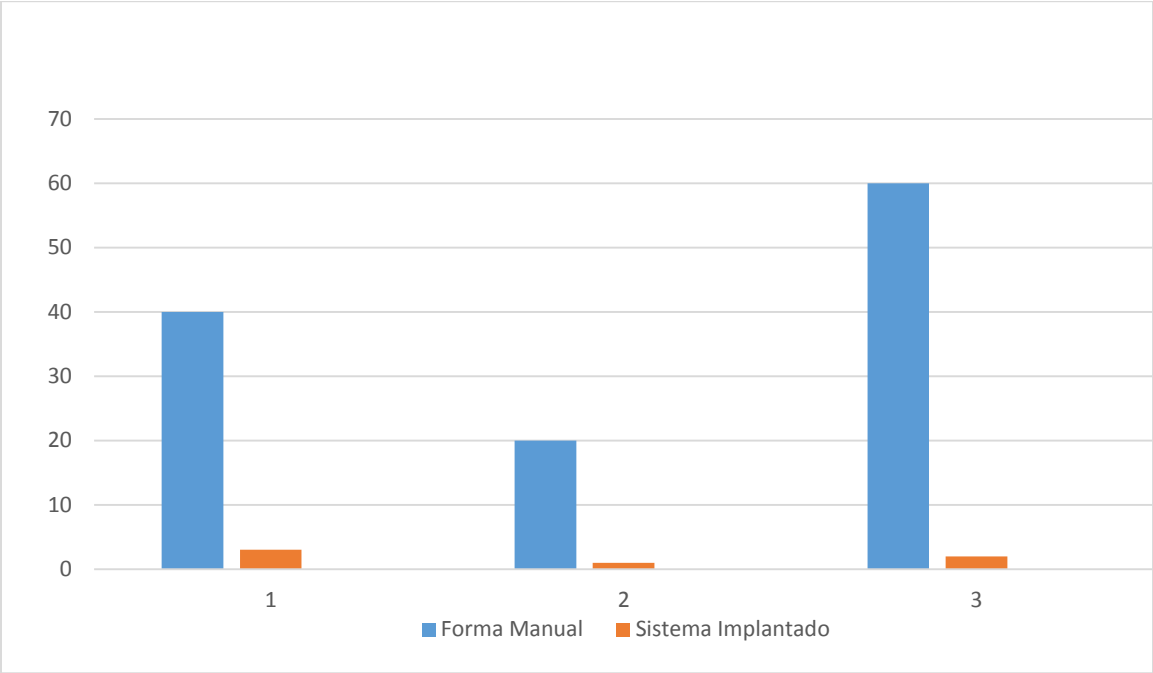
seleccionada para evaluar la factibilidad del proyecto es la Metodología Costo-Efectividad. Dentro de esta metodología la técnica de punto de equilibrio aplicable a proyectos donde los beneficios tangibles no son evidentes el análisis se basa exclusivamente en los costos. Para esta técnica es imprescindible definir una variable discreta que haga variar los costos. Teniendo en cuenta que el costo para este proyecto es despreciable, tomaremos como costo el tiempo en minutos empleado para realizar la gestión de la información de los parámetros de viento en los parques eólicos. Este se divide en 3 pasos:

Valores de las Variables (Solución Manual).

1. El Especialista Técnico deberá transformar y archivar los datos para su posterior análisis. (40 min).
2. El especialista funcional tendrá que hacer una búsqueda en los directorios de archivos para encontrar las mediciones que desea. (20 min)
3. Los Especialistas Funcionales deberán realizar un estudio de los datos realizando así gráficos en software no convencionales. (60 min).

Valores de la variable (Solución con el programa):

1. El subsistema intermedio le permitirá la Especialista Técnico almacenar los datos arrojados por los equipos en hojas de MS Excel hacia una base de datos ya transformados. (3 min).
2. El subsistema web permitirá visualizar las mediciones almacenadas en la base de datos según la fecha al Especialista Funcional para el mejor manejo de los datos. (1 min).
3. El subsistema web permitirá generar un conjunto de datos al Especialista Funcional para su mejor comprensión y fiabilidad. (2 min).



Teniendo en cuenta los resultados reflejados en las gráficas para cada uno de los casos queda demostrada la factibilidad del sistema evidenciado por la relación entre la complejidad del problema (cantidad de fuentes y receptores) y el tiempo que demora la introducción de los datos de forma manual y automatizada

4.3 Conclusiones de Capítulo.

En este capítulo se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad (Beneficio), se analizaron los efectos económicos y técnicos necesarios para la realización del software, los beneficios y costos intangibles, además se calculó el costo de ejecución del proyecto mediante la ficha de costo arrojando como resultado $ 22.00 CUC y $ 500.00 MN demostrándose la factibilidad del proyecto.

Conclusiones Generales.

Con el desarrollo del proyecto se realizó el cumplimiento de los objetivos propuestos en esta investigación, arribándose a las siguientes conclusiones:

1. Se desarrolló el software multiplataforma para la gestión de la información de los parámetros de viento en los parques eólicos que permite la automatización de los procesos de medición en las estaciones meteorológicas instaladas en los parques.
2. Durante el análisis de las metodologías y herramientas para el desarrollo de sistemas de gestión de información, se determinó la utilización de:
   - ✓ Una arquitectura de distribución “The broker”.
   - ✓ Metodología de desarrollo de software OpenUp.
   - ✓ Y los lenguajes de programación Java, Php y JavaScript.
3. La documentación de sistema permitió dejar un registro que puede ser utilizado por otros desarrolladores que deseen mantener o extender el sistema. Esto se logró con la elaboración de los artefactos:
   - ✓ Requisitos funcionales y no funcionales.
   - ✓ Diagramas de clases.
   - ✓ Diagramas de componentes.
4. El estudio de factibilidad realizado siguiendo la metodología Costo Efectividad arrojó como resultado los efectos económicos y beneficios, así como el costo de ejecución del proyecto, siendo este $22.00 CUC. y $ 500.00 MN demostrándose que es factible el proyecto.
5. La utilización de un juego de datos reales para la implementación de un caso de estudio nos permitió valorar la ampliación de algunas funcionalidades del sistema relacionadas con la entrada de datos:
   - ✓ Permitir otros formatos de documentos.
   - ✓ Permitir otra organización de los registros de datos.

Recomendaciones:

- Incorporar a la aplicación algún método para la estimación de datos faltantes o erróneos, obtenidos de las mediciones, que permita su completamiento y así evitar interpretaciones incorrectas.
- Incluir en próximas versiones la entrada de los datos de las mediciones en otros formatos de documentos.

Bibliografía

Baños, R. A. (2012). *Módulo para la administración de Savunix en Nova para servidores.* Habana.

Fernández, E. (2011). *Sistema para el Análisis de Sensibilidad en la plataforma BioSyS.*

Fernández, Y. (2012). *Metodología para el desarrollo de distribucuines GNU/Linux.* La Habana.

Figueredo, C. M. (2006). Diez Preguntas y Diez Respuestas Sobre Energía Eólica. *Diez Preguntas y Diez Respuestas Sobre Energía Eólica*, 21-22.

Frank W. Zammetti. (2009). *Practical ExtJs projects with Gears.* .

Freeman, E. (2010). *O´Reilly Desing Patterns.* O´Reilly .

GNU Operating System. . (Marzo de 2009). *http://www.gnu.org/philosophy*. Obtenido de http://www.gnu.org/philosophy: http://www.gnu.org/philosophy/free-sw.es.htm

Gutierrez, J. A. (2007). *El Mundo Informático. Software Libre.* Obtenido de El Mundo Informático. Software Libre.: http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/

http://www.cubasolar.cu/. (7 de Mayo de 2011). Obtenido de http://www.cubasolar.cu/: http://www.cubasolar.cu/biblioteca/Ecosolar/Ecosolar30/HTML/articulo01.htm;

netbeans. (2013). *www.netbeans.org.* Obtenido de www.netbeans.org.: htttp://www.netbeans.org.

Quiñones, A. (2010). *Introducción a PostgresSQL.*

Reyes, A. L. (2011). *Sistema Estimador de Mediciones de Faltantes de Vientos.* Santiago de Cuba.

Tanembaum., A. S. (2002). *Distributed System – Principles and Paradigms . .* Editorial Prentice Hall.

Wikipedia. (2012). *http://es.wikipedia.org/*. Obtenido de http://es.wikipedia.org/

Learning EXT JS. (2008). Build dynamic, desktop- style user interfaces for your Data-driven web applications.

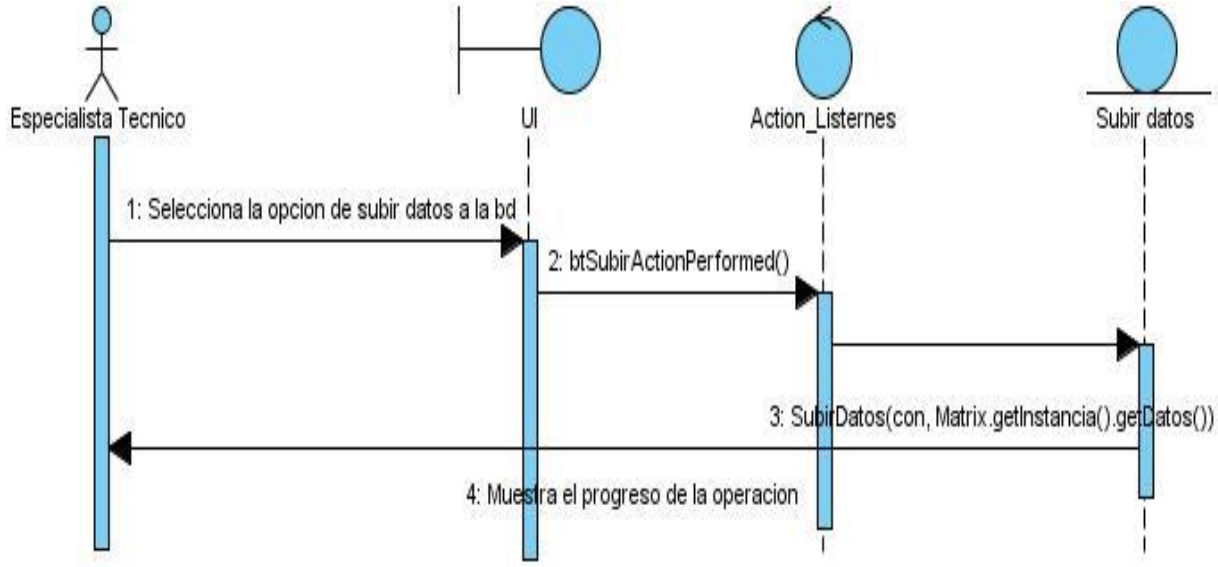
AMERICATI. Ventajas y Desventajas: Comparación de los Lenguajes C, C++ y Java. [En línea] 2006. [Citado el: 26 de Febrero de 2012.] http://www.americati.com.

Buschmann. Pattern-Oriented Software Architecture. s.l. : John Wiley & Sons, 2007.

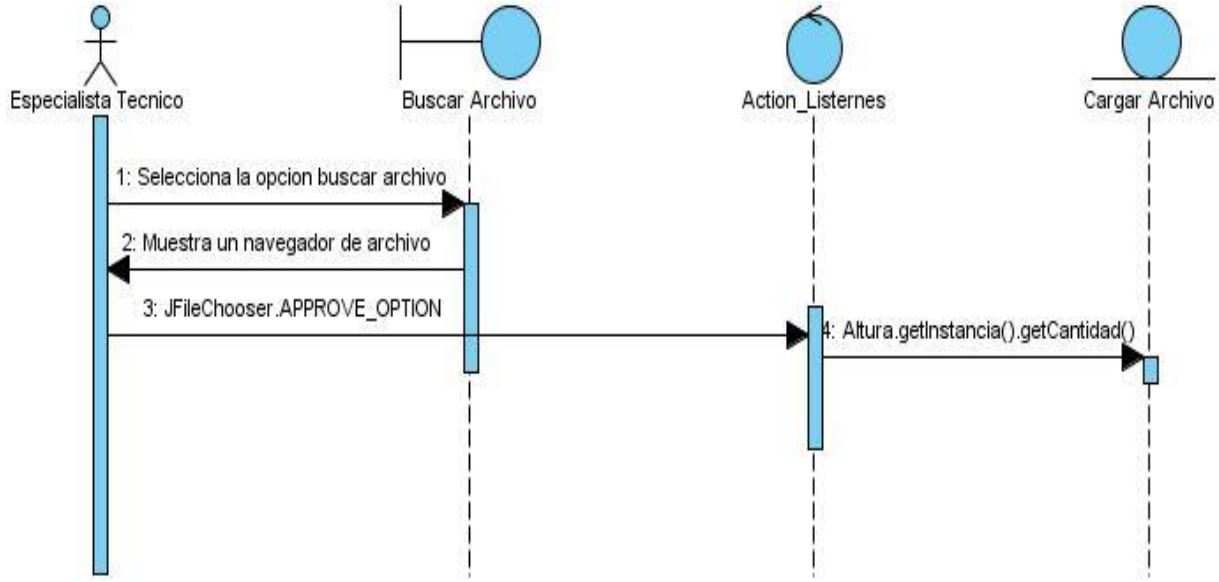
Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El proceso Unificado de Desarrollo de Software. Madrid : Addison Wesley : s.n., 2000. 84-7829-036-2.

Anexos:

Anexo 1: Diagrama de secuencia Subir los datos a la base de Datos.



Anexo 2: Diagrama de secuencia Cargar archivo.



Anexo 3: Tabla con los datos arrojados de los equipos de medición.

loggername	date_____	time___	s1a[m/s]	s1x[m/s]	s1i[m/s]	s1s[m/s]	s2a[m/s]	s2x[m/s]	s2i[m/s]	s2s[m/s]	s3a[m/s]	s3x[m/s]	s3i[m/s]	s3s[m/s]	d1a[°]	d1s[°]	[-1=error]	[ser.no.]
HLG_-_1135_	1/6/2007	0:00:00	7,4	10,6	4,3	1,5	6,1	9,9	3,4	1,5	4,6	8	2,1	1,2	147	9		C060074
HLG_-_1135_	1/6/2007	0:10:00	8,6	10,9	5,7	1,1	7,5	10,3	4,7	1,1	5,9	8,3	3,1	1,1	142	6		C060074
HLG_-_1135_	1/6/2007	0:20:00	7,2	9,5	4,6	1	6,5	9,6	4	1,1	5	7,9	2,3	1,2	147	6		C060074
HLG_-_1135_	1/6/2007	0:30:00	6,6	8,6	4,6	0,8	5,6	7,5	3,6	0,7	4,1	6	2,4	0,7	153	7		C060074
HLG_-_1135_	1/6/2007	0:40:00	6	7,4	4,5	0,6	4,8	6,3	3,4	0,6	3,6	5,4	2,3	0,6	157	6		C060074
HLG_-_1135_	1/6/2007	0:50:00	5,7	7,1	4,3	0,5	4,5	6,4	3,2	0,5	3,3	4,7	1,9	0,5	155	5		C060074
HLG_-_1135_	1/6/2007	1:00:00	5,8	7,1	4,3	0,5	4,7	6,1	3,2	0,6	3,3	4,7	2,1	0,5	151	5		C060074
HLG_-_1135_	1/6/2007	1:10:00	6,3	8,1	4,6	0,7	5,1	6,8	3,5	0,7	3,9	5,5	2,3	0,6	150	7		C060074
HLG_-_1135_	1/6/2007	1:20:00	6,5	8,4	4,1	0,8	5,4	7,3	3,1	0,8	3,9	5,9	2,3	0,7	152	7		C060074
HLG_-_1135_	1/6/2007	1:30:00	5,4	6,6	4,2	0,4	4,5	6,2	3,2	0,5	3,5	5,1	2,1	0,5	155	5		C060074
HLG_-_1135_	1/6/2007	1:40:00	5,4	7	3,8	0,5	4,3	5,9	2,7	0,6	3,2	5	1,9	0,5	157	7		C060074
HLG_-_1135_	1/6/2007	1:50:00	6,1	7,8	4,4	0,5	4,9	7	3,3	0,7	3,5	5,6	2,3	0,5	153	6		C060074
HLG_-_1135_	1/6/2007	2:00:00	5,9	7,6	4,2	0,6	4,8	6,4	2,8	0,6	3,5	5	2,2	0,6	153	7		C060074
HLG_-_1135_	1/6/2007	2:10:00	5,5	7	4,1	0,5	4,6	6,3	3,2	0,6	3,3	5,5	2,2	0,6	160	8		C060074
HLG_-_1135_	1/6/2007	2:20:00	5,4	7	3,5	0,7	4,3	5,9	2,7	0,7	3	4,1	1,9	0,4	155	6		C060074
HLG_-_1135_	1/6/2007	2:30:00	5,8	6,9	4,3	0,5	4,7	6,4	3	0,6	3,4	5,1	2,2	0,6	150	5		C060074

Anexo 4. Potencial Eólico de Holguín (según Mapa de Potencial Eólico en Cuba)