Universidad de las Ciencias Informáticas

Facultad 6

Título: Sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas.

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

Autores: Yan Ernesto Pizarro Estévez

Marla Cuza Carballo

Tutores: Ing. Georvys González Rojas.

Ing. Loismarx Peña González.

La Habana, junio 2012

"Año 54 de la Revolución"

Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte...
Che

DECLARACIÓN DE AUTORIA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 26 días del mes de junio del año 2012.

Yan Ernesto Pizarro Estévez	**Marla Cuza Carballo**
_____	_____
Firma del Autor	Firma del Autor
Ing. Georvys González Rojas	**Ing. Loismarx Peña González.**
_____	_____
Firma del Tutor	Firma del Tutor

DATOS DE CONTACTO

Tutor: Georvys González Rojas.

Correo electrónico: grojas@uci.cu.

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas en 2011

Tutor**:** Loismarx Peña González

Correo electrónico: lpgonzalez@uci.cu.

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas en 2008

AGRADECIMIENTOS

Marla.

Agradezco a mis padres Ana Doris y Salvador porque sin ellos no habría llegado hasta aquí, por apoyarme siempre en los buenos y malos momentos de mi vida. A Marlon mi hermano querido por mostrarme el camino para optar por esta carrera y ser mi amigo y consejero. A mi familia tan grande y linda que siempre se ha preocupado por mí y ha seguido mis pasos en cada momento brindándome cada uno de ellos su apoyo incondicional. A los amigos de mi vieja escuela Yudit, Ines, Celso, Yaciel, Omar, Hector Raúl, Daniel, Marcos y Arnel por permitirme ayudarlos y necesitarlos.

Me gustaría agradecer a todas las personas que la UCI me dejó conocer y que me brindaron su ayuda sin pedir nada a cambio, por ayudarme en estos cinco años de mi carrera hasta el final. A mis amigos Ariel Felipe, Yurisbel, Yusnieski, Luis Mario, Ale, los muchachos del café y mis compañeras de cuarto.

A mis amigas del alma que me han acompañado en estos cinco años de preocupaciones, trabajo, alegrías y fiestas. Agradezco haberlas conocido a cada una de ellas, a Anita por ser mi consejera y paño de lágrimas, por salvarme en los momentos más difíciles por apoyarme siempre que lo necesitaba y por hacerme ver mis errores cuando yo no podía. A Claudia por dejarme ser su amiga y por brindarme su mano consejera y su cariño incondicional. A Juliet por preocuparse siempre por mí y acompañarme en los buenos momentos de alegrías y locuras. A Margarita, por soportarme desde el segundo año de la carrera hasta hoy, por ayudarme y quererme tanto como yo a ella. A Nelly y Angel que me han dado tanta confianza y cariño. A todos los profesores que de verdad me enseñaron e hicieron por mí para mi superación. Agradecerle también a Yan mi compañero de tesis que sin él no hubiera podido concluir mi trabajo de diploma.

Yan

Le agradezco principalmente a mi abuela Alla por ser la persona que más me ha soportado durante tanto tiempo que llevo atormentando a la gente que más me quiere.

A mi abuelo Lolo por ser la persona que más me ha querido y a la que más he maltratado en mi vida.

A María mi mamá que la quiero más de lo que le digo, a mi hermana Thais que la quiero mucho más de lo que se imagina.

A mi papá por darme todos los gustos que he querido, a mi hermano Manuel, que a pesar que no hemos vivido junto, lo quiero igual que a las personas de mi casa, a mi tía Marica que siempre nos ha tratado como si fuéramos sus hijos, a mi tía Marta que aunque ella me diga que los besos que le doy son los besos de judas también la quiero mucho.

Al amor de mi vida que esta demás mencionar quien es, que aunque ella no me lo cree, es la novia que más he querido en mi vida y la que más mala crianza me ha soportado, y que de no ser por ella no hubiera estado aquí.

A mi primo Cheo que venimos juntos, desde que nos cuidaban Irma y Nena, a mi abuela chele por darme lo poco que tiene y por soportar todas las cosas que le he dicho.

A todos los socios que tengo Janier, Ruano, Gordillo, Rodolfo, el Butin, Silvio, Yaniel, Víctor, Jota, Ramiro, Alejandro y Dayatnis, Santana, Marquimides, a tiburon, la bala, a mama, Humberto, al chata, Raúl.

DEDICATORIA

A mis padres y mi hermano por ser mi razón de vivir.

Marla.

Le dedico esta tesis primeramente a mi abuela, mi abuelo.

A mi mamá, a Ileana que ahora si me va a tener de tiempo completo.

A nenyn que ya no va a tener que gastar más ni un peso en llamadas de teléfono.

A mi tía Martica por ser como una segunda madre para mí.

Yan

RESUMEN

El agotamiento del combustible más eficiente jamás conocido: el petróleo, sumerge la humanidad en uno de los momentos más traumáticos de su historia. La presente investigación surge debido a la necesidad del ahorro de la energía eléctrica en la Universidad de las Ciencias Informáticas (UCI). El presente trabajo de diploma se enmarca en el tema de los Sistemas de Gestión de la Información y su utilización para apoyar la distribución del plan de consumo energético en las áreas docentes. En el mismo se realiza una investigación detalla para escoger la metodología, herramientas y tecnologías a utilizar para el correcto diseño e implementación del Sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas. Por último se probaron los requerimientos funcionales identificados aprobando el resultado final, donde se obtuvo un sistema que contiene todos los reportes del consumo estimado según un tipo de local en una fecha determinada y visualiza el consumo real existente. Además gestiona información, planifica y controla el consumo energético estimado de un área docente.

PALABRAS CLAVES: consumo energético, gestión de información, plan de consumo energético, sistemas de gestión.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Para el hombre moderno, es impensable la vida sin iluminación, calefacción, refrigeración y transporte. Esta dependencia energética, se ha convertido en un exceso, específicamente de combustibles fósiles, siendo estos recursos no renovables.

En los últimos años el consumo de energía eléctrica en el mundo se ha elevado a un ritmo superior al crecimiento económico, la reducción de los suministros de petróleo y la duplicación de los precios del crudo en el mundo, unido a las graves consecuencias provocadas por el calentamiento global, han llevado a muchos países a sufrir una crisis económica. Para confrontar esta crisis los países del mundo se han hecho un llamado con el objetivo de emprender planes de control de energía, medidas de ahorro efectivas y equilibradas para evitar el agotamiento de estos recursos.

Cuba no se ha quedado atrás ante esta realidad mundial y ha desarrollado grandes proyectos como la Revolución Energética, la cual tiene como objetivo primordial el uso racional y eficiente de los pocos recursos energéticos con que cuenta. Este proyecto consta de varios programas como es el caso del Programa de Ahorro de Energía de Cuba (PAEC); el mismo está integrado por proyectos de trabajo y sistemas de información que han permitido ahorrar cuantiosos volúmenes de combustible y recursos a la nación. En todas y cada una de las instituciones del país se trabaja para disminuir el gasto energético, a pesar de ello, todavía existen entidades donde el nivel tecnológico es bastante avanzado y se hace difícil el control y reducción del mismo.

La Universidad de las Ciencias Informáticas (UCI) es una de las entidades del país que más energía eléctrica consume. Este crecimiento se ha evidenciado en la cantidad de consumidores eléctricos con los que cuentan las entidades creadas a partir del aumento de la infraestructura docente-productiva en función de sus respectivos objetivos, entre las que se destacan las áreas docentes. El centro universitario cuenta con un departamento de Atención a Programas Energéticos, encargado, entre otras cosas, de analizar el consumo de la energía eléctrica. El departamento recibe periódicamente un plan de consumo energético dictado por el Ministerio de Educación, en el cual está plasmada la energía eléctrica que debe consumir la universidad. Con el objetivo de mejorar su trabajo ha divido el centro por áreas, de esta manera realiza una distribución del plan entre las diferentes áreas dependiendo de un análisis basado en los datos anteriores obtenidos en los analizadores eléctricos de las mismas. Para visualizar, comprobar y obtener los datos del consumo energético, la dirección del departamento cuenta con un Sistema de Supervisión del Consumo Energético que recolecta en tiempo real los datos de los analizadores eléctricos en cada área. En caso de que un área docente consuma más de lo establecido, el departamento no tiene manera de conocer si el consumo asignado a esta área era el correspondiente de acuerdo con las necesidades diarias que debe tener esta entidad con

respecto al cronograma de producción y docencia en las cuales intervienen los consumidores eléctricos. Por lo antes planteado la dirección del departamento no puede tomar medidas diferenciadas de distribución de consumo energético para las distintas áreas docentes de la universidad.

Debido a esto se identifica el siguiente **problema de la investigación:** ¿Cómo contribuir a la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas?

Definiendo como **objeto de estudio:** los sistemas de gestión de información, centrando su **campo de acción** en el sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes.

Para dar respuesta al problema antes planteado se determina como **objetivo general:** Desarrollar un sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas.

De acuerdo con esta propuesta se derivan los siguientes **objetivos específicos:**

1. Fundamentar los conceptos asociados a los sistemas de gestión de información para la planificación y control del consumo energético.
2. Seleccionar la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas.
3. Realizar el análisis y diseño del sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas.
4. Implementar el sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas.
5. Realizar pruebas al sistema para la gestión de la información referente a la planificación y control del consumo energético en las áreas docentes de la Universidad de las Ciencias Informáticas.

Para dar cumplimiento a los objetivos anteriores se plantean las siguientes **tareas de la investigación:**

1. Análisis de los conceptos fundamentales de los sistemas de gestión de información.
2. Caracterización de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del sistema.
3. Realización del modelo de análisis del sistema.
4. Realización del modelo de diseño del sistema.
5. Implementación de los componentes del sistema.

6. Realización de pruebas de caja negra y caja blanca para validar el correcto funcionamiento de la aplicación.

El presente trabajo de diploma se ha estructurado de la siguiente manera, introducción, cuatro capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, anexos y glosario de términos.

Capítulo 1 Fundamentos teóricos de la investigación.

En este capítulo se definen temas relacionados con el desarrollo de los sistemas para la gestión de la información, se describen los conceptos fundamentales que serán tratados a lo largo de la investigación, así como la selección de la metodología y herramientas que serán utilizadas, identificando las ventajas y las desventajas de los diferentes tipos de soluciones para el apoyo de la toma de decisiones. Además, se realiza un profundo estudio del arte para un mejor desarrollo del sistema que se prevé realizar.

Capítulo 2 Análisis y diseño del Sistema de información

En este capítulo para un mejor entendimiento del negocio, se detallan las características del sistema para la gestión de la información, se especifican las necesidades del sistema, los requisitos funcionales y no funcionales, así como los casos de uso del sistema. Además, se diseña el modelo de datos del sistema.

Capítulo 3 Implementación del Sistema de información

En este capítulo se muestran los diagramas de componentes precisados para la implementación del sistema, a su vez se muestran fragmentos de códigos de las principales clases del mismo. Se muestran los diagramas de componentes que se definieron durante la implementación, así como las principales pantallas del sistema.

Capítulo 4 Pruebas del Sistema de información

En este capítulo se exponen pruebas de caja negra y caja blanca para realizar las validaciones a nivel de desarrollador del sistema, obteniendo resultados de estas pruebas, los cuales se describen en las no conformidades del sistema. A su vez se muestran varias pantallas de la aplicación.

CAPÍTULO 1: FUNDAMENTO TEÓRICO DE LA INVESTIGACIÓN

En este capítulo se dan a conocer los principales conceptos que contribuyen a un mejor entendimiento del problema en cuestión, además se realiza una caracterización de diferentes metodologías para una mejor selección de la que finalmente utilizaremos. También de las diferentes herramientas y tecnologías actuales a utilizar en la construcción de la solución.

1.1 Sistema de gestión

Un sistema de gestión es una estructura probada para administrar y mejorar las continuas políticas, cúmulo de información, procedimientos y procesos de una organización. Ayuda a lograr los objetivos de la organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado. Además permite facilitar, simplificar y realizar automáticamente procesos que tradicionalmente se realizan de forma manual. Esto evita errores y mejora el tiempo de realización de las tareas. Estableciendo un progresivo control en las entidades financieras, obteniendo una mayor fiabilidad, seguridad y agilidad.

Los sistemas de gestión se hacen necesarios en las empresas u organizaciones que operan en el siglo XXI, las cuales se enfrentan a muchos retos significativos entre ellos:

- ✓ Competitividad
- ✓ Rentabilidad
- ✓ Globalización
- ✓ Velocidad de los cambios
- ✓ Capacidad de adaptación
- ✓ Crecimiento

Equilibrar estos y otros requisitos empresariales puede constituir un proceso difícil y desalentador. Es aquí donde entran en juego los sistemas de gestión, al permitir aprovechar y desarrollar el potencial existente en la organización.

La implementación de un sistema de gestión eficaz puede ayudar a [1]:

- ✓ Gestionar los riesgos sociales, medioambientales y financieros
- ✓ Mejorar la efectividad operativa
- ✓ Reducir costos
- ✓ Aumentar la satisfacción de clientes y partes interesadas
- ✓ Lograr mejoras continuas
- ✓ Potenciar la innovación
- ✓ Eliminar las barreras al comercio.

1.1.1 Componentes de un Sistemas de gestión

Un Sistema de Gestión está compuesto por un conjunto de etapas unidas en un proceso continuo, que permite trabajar ordenadamente una idea hasta lograr mejoras y su continuidad.

En este proceso se establecen cuatro etapas, que hacen de este sistema, un proceso circular virtuoso, pues en la medida que el ciclo se repita recurrente y recursivamente, se logrará en cada ciclo, obtener una mejora.

Las cuatro **etapas del sistema de gestión** son:

Etapa de ideas: El objetivo de esta etapa es trabajar en la idea que guiará los primeros pasos del proceso de creación que se logra con el sistema de gestión propuesto.

Etapa de planeación: La planificación constituye una etapa fundamental y el punto de partida de la acción directiva, ya que supone el establecimiento de sub-objetivos y los cursos de acción para alcanzarlos.

Etapa de implementación: En su significado más general, se entiende por gestión, la acción y efecto de administrar. Pero, en un contexto empresarial, esto se refiere a la dirección que toman las decisiones y las acciones para alcanzar los objetivos trazados.

Etapa de control: El control es una función administrativa, esencialmente reguladora, que permite verificar (o también constatar, palpar, medir o evaluar), si el elemento seleccionado (es decir, la actividad, proceso, unidad, sistema, etc.), está cumpliendo sus objetivos o alcanzando los resultados que se esperan [2].

1.1.2 Sistemas de gestión de energía existentes en el mundo

En el mundo existen muchos sistemas de gestión de la información dedicados a tareas específicas de una empresa u organización. Estos sistemas han sido clasificados y agrupados en distintos tipos de conceptos como son: sistemas de gestión energética, calidad, empresarial, integral, comercial, entre otros.

Los sistemas de gestión energética han sido desarrollados con el propósito de satisfacer necesidades existentes en diferentes organizaciones, los cuales permiten controlar, planificar y visualizar de forma efectiva el consumo de combustible energético asociado a entidades [3].

Algunos de estos son:

- ✓ El Sistema de Gestión Energética **ContaLuzWeb** es una herramienta de apoyo a los encargados públicos en el proceso de control y toma de decisiones relativas al consumo y despacho de electricidad. Aplicable a cualquier institución tanto pública como privada, de servicio o de producción. Conociendo de forma estadística los consumos energéticos tanto eléctricos como térmicos, identificados por áreas, se puede analizar la información obtenida, obtener índices, comparar los indicadores de distintas áreas y finalmente el equipo de gestión

de energía de la institución tomará medidas correctivas necesarias. Con la gestión apoyada por el sistema ContaLuzWeb se desarrolló el programa PureUSP, programa que ha permitido incentivar el uso racional de la energía eléctrica de la Universidad de San Pablo y aumentar la eficiencia del recurso energético [4].

- ✓ El Centro de Control del Régimen Especial, **CECRE**, es el primero del mundo que gestiona la producción de régimen especial. Creado por la Red Eléctrica de España (REE) para integrar la mayor cantidad de energías renovables en condiciones de seguridad y en función de las necesidades del sistema. Su función principal es supervisar y controlar las plantas de generación renovables, principalmente los parques eólicos, para garantizar de forma segura la integración de su producción en el sistema eléctrico [5].

1.1.3 Sistemas de gestión de energía existentes en Cuba

- ✓ **XenoLight** es un sistema de control automático integral, que atendiendo a los valores de parámetros de temperatura, caudal, seguridad, ocupación, presión y consumo eléctrico, actúa positivamente sobre los espacios ocupados con énfasis en el ahorro energético y el confort climático. La automatización integral de la instalación resultante permite ahorros energéticos que están entre el 25 y el 40 por ciento del consumo energético anterior.
- ✓ El **Sistema de Supervisión y Control de Procesos Industriales** para los países integrantes del ALBA (Alternativa Bolivariana para las Américas), supervisa y controla los procesos industriales en el sector petrolero, aumentando la eficiencia y seguridad de esta industria. Esta aplicación, desarrollada por especialistas cubanos y venezolanos para Petróleos de Venezuela S.A. (PDVSA), utiliza software libre, con lo cual contribuye no solo a la disminución de los costos sino también a elevar la soberanía tecnológica de esta importante rama de la economía venezolana.
- ✓ El **Sistema de Monitoreo de Grupos Electrógenos** posibilita la supervisión de las principales variables de los equipos de generación distribuida, instalados en los emplazamientos de todo el país, en aras de una mayor eficiencia energética. Actualmente se encuentra instalado en prácticamente todas las baterías y siempre que entra en línea un nuevo emplazamiento es una condición fundamental que tenga el sistema instalado [6].

1.1.4 Sistemas de gestión de energía existentes en la UCI

En la UCI existe un departamento de Atención a Programas Energéticos encargado de supervisar, analizar el consumo de la energía eléctrica y distribuir el plan de consumo de la universidad entre las áreas, con las cuales cuenta. El departamento para la realización de estas tareas cuenta con un Sistema de Supervisión de Consumo Electro Energético que brinda la posibilidad de recoger en tiempo real los datos de consumo de energía y otros parámetros de los analizadores eléctricos de cada

entidad o área en la universidad. Este sistema abre un gran número de posibilidades desde el punto de vista de la supervisión energética como herramienta para el control y ahorro de energía, esta última con una gran importancia para el país. Con la utilización de este sistema de supervisión energético la dirección del departamento obtiene un reporte periódico según el consumo energético en las entidades y lleva un seguimiento total del mismo. Mediante este seguimiento el departamento realiza una distribución diferenciada de consumo estimado por cada área, basándose en los reportes obtenidos en fechas pasadas.

Análisis de las Soluciones Existentes

A pesar de que las soluciones mencionadas anteriormente permiten controlar, supervisar, manejar operaciones de consumo energéticos y contar con excelentes funciones, se puede apreciar que solo fueron creadas para funciones específicas. Después de haber analizado estos sistemas de gestión de energía, teniendo en cuenta los aportes que brindan y las características específicas de cada sistema como herramienta informática, se puede arribar a la conclusión de que no existe un sistema que se ajuste a las necesidades del departamento de Atención a Programas Energéticos de Universidad de las Ciencias Informáticas. Es por ello que se consideró necesaria la construcción de una solución económica y acorde a sus requisitos individuales. Para ello es imprescindible realizar un estudio de la metodología, herramientas y tecnologías existentes, con el propósito de seleccionar las más adecuadas para la solución a desarrollar.

Metodologías de desarrollo de software

Una metodología de desarrollo de *software* es una colección de documentación formal sobre los procesos, políticas y procedimientos que intervienen en el desarrollo del *software*, encaminada a garantizar la eficacia, mediante el cumplimiento de los requisitos y la eficiencia, mediante la optimización del tiempo.

Las metodologías se pueden agrupar en dos grandes grupos, las tradicionales y las ágiles. Las metodologías tradicionales se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán; mientras que metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas, en estas el cliente llega a formar parte del equipo de trabajo. Además flexibilidad ante los cambios que quisieran efectuar los clientes a lo largo del desarrollo de un sistema sin perder de vista la calidad del producto final.

Existe un número elevado de estas metodologías entre las que se destacan Open Up y XP.

Para el desarrollo del sistema propuesto es necesario que se utilice una metodología ágil debido a que este tipo de metodologías son capaces de adaptarse a los cambios que pueden ocurrir tanto por modificaciones solicitadas por los clientes, así como cambios en la arquitectura y en la infraestructura.

OpenUP

El OpenUp es un marco de proceso de desarrollo de software, proceso modelo y extensible dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. Provee un conjunto simplificado de artefactos, roles, tareas y guías de trabajo. Además evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP. Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas. Todas y cada una de las actividades y procedimientos deben ser correctamente descritas para que se pueda saber en todo momento quién está haciendo qué, cómo y cuándo lo está haciendo y cuáles son los objetivos que persigue [7].

Programación Extrema

XP (acrónimo de Extreme Programming), es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Esta metodología se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo técnico. Entre los artefactos que se utilizan en XP vale la pena mencionar las tarjetas de historias (story cards); son tarjetas comunes de papel en que se escriben breves requerimientos de un rasgo, jamás casos de uso; pueden adoptar el esquema CRC. Tienen una granularidad de diez o veinte días, se usan para estimar prioridades, alcance y tiempo de realización. Otros productos son listas de tareas en papel o en una pizarra (jamás en computadora) y gráficos visibles pegados en la pared [8].

Teniendo en cuenta los conceptos y características expuestos anteriormente y conociendo que el sistema a realizar es una aplicación web dinámica, se hace necesaria la ejecución de un diseño completo del producto con los diagramas necesarios para un mejor entendimiento por parte del desarrollador. Se concluye que para el desarrollo de este *software* se propone utilizar la metodología OpenUP. Esta metodología cuenta entre sus principales características con un desarrollo iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura con el fin de reducir los riesgos y organizar el desarrollo.

1.2 Fundamento de las herramientas y tecnologías a utilizar

Para poder realizar una aplicación web es necesario tener conocimiento de las tecnologías y herramientas que aportan mejoras, garantizando así un desarrollo y mantenimiento estándar, de este modo se establecen criterios de selección para las mismas.

1.2.1 Framework de desarrollo

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona estructura al código fuente, forzando al desarrollador a crear un código más legible y fácil de mantener facilitando la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas [9].

Symfony 2.0

Es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Cuenta con una enorme colección de bibliotecas bien pensada que hace el trabajo más fácil y un programador puede sacar utilidad de cualquiera de estas bibliotecas que están integradas y son fáciles de usar. Symfony es compatible con la mayoría de Gestores de Bases de Datos, como MySQL, PostgreSQL, Oracle, entre otros. Es multiplataforma y está basado en el patrón arquitectónico Modelo Vista Controlador (MVC), donde divide una aplicación interactiva en tres áreas: procesamiento, salida y entrada. Symfony 2.0 es independiente del sistema gestor de bases de datos, está desarrollada completamente con PHP 5 sigue la mayoría de las mejores prácticas y patrones de diseño para la web y es fácil de extender, lo que permite su integración con librerías desarrolladas por terceros [9].

En esta versión se integra el uso de bundles (que no son más que paquetes que contendrán todo lo referente a una determinada aplicación) lo que permite la integración con otras aplicaciones que se realicen posteriormente usando la misma versión del framework, tiene integrado el ORM Doctrine para la persistencia de los datos, además permite la conectividad con servicios web, de ahí la necesidad del uso de Symfony en su versión 2.0 para el desarrollo del sistema.

1.2.2 Framework de interfaz

jQuery

jQuery es un framework para el lenguaje javascript que ofrece una infraestructura con facilidad para la creación de aplicaciones complejas del lado del cliente. Implementa una serie de clases que permiten al programador trabajar sin preocuparse del navegador al que accede el usuario, ya que funcionan de forma exacta en todas las plataformas más habituales [10].

Ventajas del framework jQuery

- Ofrece ayuda en la creación de interfaces de usuario, efectos dinámicos y aplicaciones que hacen usos de Ajax.
- Se obtiene de manera gratuita, ya que el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial.
- Es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework.
- Es ligero en comparación con otros marcos de javascript [10].
- Tiene una amplia gama de plugins disponibles para las diversas necesidades específicas.

Teniendo en cuenta que sus componentes visuales permiten desarrollar en poco tiempo una aplicación agradable y que hace uso de un diseñador gráfico para mostrar información de cuantiosa importancia en el sistema, se hizo necesario el uso del *framework* jQuery en el desarrollo del mismo.

1.2.3 Lenguajes de programación

PHP 5.3

PHP es un lenguaje interpretado del lado del servidor, especialmente creado para el desarrollo de páginas web dinámicas, con gran facilidad para incrustarse dentro del código HTML. Entre las ventajas que lo distinguen se destaca el hecho de ser un lenguaje libre, multiplataforma y que permite establecer conexión con la mayoría de los gestores de base de datos, tales como MySQL, PostgreSQL, Oracle, entre otros. Tiene gran cantidad de funciones ejemplificadas y una amplia documentación. No requiere definición de tipos de variables ni manejo detallado de bajo nivel [11].

Se selecciona PHP como lenguaje de programación para realizar este sistema debido a la necesidad de prever una futura integración con productos del Centro de Tecnologías y Gestión de Datos (DATEC), estos últimos están escritos en dicho lenguaje, por lo cual se recomienda el desarrollo del sistema en este lenguaje. Por otra parte Symfony 2.0 obliga a utilizar la versión 5.3 de dicho lenguaje u otra versión superior

Javascript

Javascript es un lenguaje de programación del lado del cliente que permite a los desarrolladores crear acciones en sus páginas web. Utilizado para crear pequeños programas que luego son insertados en una página. Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios y es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros. Este es un lenguaje interpretado, no requiere compilación y está basado en objetos. No es como Java, un lenguaje de programación orientada a objetos (OOP). JavaScript no emplea clases ni herencia, típicas de la OOP [12].

1.2.4 Leguaje de etiqueta HTML

Es un procedimiento para especificar tipos de documentos estructurados y expresiones de marcas para representar esos mismos documentos. El término HTML se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de marca. Es además un lenguaje muy natural que admite representar hipertexto, es decir, texto mostrado de forma organizada e interesante, con enlaces que transfieren a otros documentos o fuentes de información relacionadas, y con inclusiones de multimedia.

Ventajas que proporciona:

1. Permite especificar la estructura lógica del contenido ya sea títulos, párrafos, enumeraciones y definiciones así como los disímiles efectos que se quieren tratar [13].

1.2.5 Sistema de Gestor de Bases de Datos

Un Sistema Gestor de Base Datos (SGBD) es un conjunto de programas de propósito general que dan paso a la creación, mantenimiento y construcción de una base de datos, asegurando la integridad, confidencialidad y seguridad de la misma [14].

PostgreSQL

PostgreSQL es un servidor de base de datos relacional orientada a objetos de software libre, es un sistema que aproxima los datos a un modelo objeto-relacional, pues incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

Características de PostgreSQL v 9.1

- ✓ Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes, cadenas de bits, así como la creación de tipos propios.
- ✓ Incorpora una estructura de datos array y soporta el uso de índices, reglas y vistas.
- ✓ Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- ✓ Alta concurrencia, permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- ✓ Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados en varios idiomas [14].

Se decide utilizar PostgreSQL en su versión 9.1 como gestor de base de datos por ser esta una herramienta de código abierto potente en el mercado. Utiliza un modelo cliente/servidor y usa multiprocesos, que permiten garantizar la estabilidad del sistema. Un fallo en algunos de sus procesos no altera la funcionabilidad de los otros procesos en ejecución y el sistema no se verá afectado.

1.2.6 Administrador de Bases de datos

Las bases de datos necesitan ser administradas, por lo que necesitan un programa con interfaz gráfica. Existen muchas herramientas que administran las bases de datos desarrolladas en PostgreSQL entre las que se encuentra PgAdmin que es una de las más utilizadas por la comunidad de desarrollo del *software* libre.

PgAdmin v 1.14: esta herramienta diseña, mantiene y administra fácilmente las bases de datos construidas en PostgreSQL, es multiplataforma y está liberada bajo la licencia Open Source. Es la herramienta más popular y completa, diseñada para responder a las necesidades de los usuarios permitiéndole escribir desde simples consultas y sentencias SQL hasta diseñar complejas bases de datos [14].

1.2.7 IDE de desarrollo

Integrated Development Environment (IDE), es un entorno de desarrollo integrado. Un editor de código que además puede servirnos para depurar y facilitarnos las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación.

NetBeans 7.0.1 es un IDE, producto de un proyecto de código abierto exitoso que contiene una gran comunidad de desarrolladores. Está codificado en el lenguaje de programación Java, y permite desarrollar aplicaciones web, de escritorio, y móviles usando las plataformas de Java. Soporta otros lenguajes de programación como son C/C++, Ruby y PHP, posibilitando el desarrollo de aplicaciones web. Es libre y gratuito, multiplataforma, disponible para diversos sistemas operativos como OpenSolaris, Windows, MacOS y GNU/Linux [15]. Es un IDE gratuito e ideal para el trabajo con Symfony 2.0 en su versión 7.0.1, por lo que resuelve las necesidades propias del proceso de desarrollo del sistema.

1.2.8 Servidores Web

Apache 2

Apache es el servidor Web más difundido y utilizado en Internet, con código abierto, multiplataforma y modular. Se desarrolla dentro del proyecto HTTP server de la fundación de software apache. Presenta alta estabilidad, seguridad y facilidad de expansión. El servidor apache es un software que está estructurado en módulos, es decir, está dividido en muchas porciones de código que hacen referencia a diferentes aspectos o funcionalidades del servidor web. Esta estructura es intencionada ya que la

configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Las características más comunes que lo distinguen son:

- ✓ Soporta CGI, Perl, PHP.
- ✓ Soporte para bases de datos.
- ✓ Soporte SSL para transacciones seguras.
- ✓ Soporta HTTP 1.1.
- ✓ Incluye soporte para *host* virtuales.
- ✓ Mensajes de error altamente configurables.
- ✓ Bases de datos de autenticación y negociado de contenido [16].

1.2.9 Lenguajes de Modelado

Los lenguajes de modelado son notaciones, en su mayoría visuales, que intentan representar un sistema de software a un nivel mucho más alto que los lenguajes de programación, representándolo en formas más intuitivas para personas sin especialización en informática.

Lenguaje Unificado de Modelado

El Lenguaje de Modelado Unificado (**UML - Unified Modeling Language**) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar conceptualmente procesos de negocio y funciones de sistema, además de clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. Tiene como objetivo entregar un material de apoyo que le permita al lector poder definir diagramas propios como también poder entender el modelado de diagramas ya existentes [17].

1.2.10 Herramienta CASE para el modelado

Las Herramientas de Software Asistidas por Ordenador (CASE, por sus siglas en inglés de Computer Aided Software Engineering), se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. La principal ventaja de su utilización, es la mejora de la calidad de desarrollos realizados y el aumento de la productividad.

Visual Paradigm 8.0

Para el modelado del sistema, se decide utilizar **Visual Paradigm for UML** en su versión 8.0, por ser una herramienta que soporta el ciclo de vida completo en el desarrollo de *software*: análisis y desarrollos orientados a objetos, construcción, prueba y despliegue. Permite dibujar todo tipo de diagrama de clases, código inverso, generación de código a partir de diagramas y generar documentación. Provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java .NET y PHP. Está diseñada para

usuarios interesados en sistemas de *software* de gran escala con el uso del acercamiento orientado a objeto [18].

1.3 Conclusiones del capítulo

En este capítulo se presentaron los conceptos más significativos para la comprensión de este trabajo. Se hizo un estudio y valoración de las principales herramientas, metodologías y lenguajes a utilizar en el desarrollo del sistema. Se definió como framework para la implementación del sistema a Symfony en la versión 2.0. Como metodología de desarrollo OpenUP, utilizando UML como lenguaje de modelado. El lenguaje de programación del lado del servidor a utilizar es PHP 5.3 y el Sistema de Gestor de Base de Datos que más se ajusta a las necesidades de esta aplicación es PostgreSQL 9.1. Se define como servidor web Apache 2 y la herramienta para modelado Visual Paradimg en la versión 8.0.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

La creación de sistemas informáticos es un proceso en el que se desarrollan artefactos que, luego de ser integrados, posibilitan responder a las necesidades del cliente. En este proceso se identifican varias etapas, que van desde la declaración del problema y los requisitos del sistema, hasta las pruebas y la liberación del mismo. En el presente capítulo se define la propuesta del sistema a desarrollar, con el propósito de satisfacer el objetivo de la investigación. Para optimizar la comprensión de la solución propuesta se realiza el análisis y diseño utilizando la metodología ágil OpenUp, donde describe el modelo conceptual y los principales procesos del sistema mediante la especificación de casos de uso. Como parte del diseño de la solución se desarrollan los diagramas de clases y de secuencia, que constituyen los artefactos principales generados para esta etapa.

2.1 Breve descripción del sistema

El sistema a desarrollar le proporcionará a la dirección del departamento de Atención a Programas Energéticos gestionar la información referente a la planificación y control del consumo energético en las áreas docentes, el mismo dará solución a las necesidades planteadas por este departamento. Esta aplicación brindará la posibilidad de conocer el consumo estimado de las áreas docentes mediante cálculos matemáticos. Podrá analizar las lecturas recogidas de los analizadores eléctricos instalados en cada área, permitiendo así comparar estas lecturas con la planificación estimada. Muestra el comportamiento diario del consumo de energía eléctrica por cada tipo de local y total del área docente mediante gráficas. El sistema está implementado de forma tal que solo le brinde al usuario las funcionalidades que le está permitido realizar. El sistema debe ser fácil de usar y seguro para el especialista que posea los conocimientos referentes al control del consumo de la electricidad**.**

2.2 Modelo de dominio

El modelo de dominio permite comprender y describir las clases más importantes dentro del contexto del sistema, es decir, este contribuye en la comprensión del problema que el sistema resuelve con relación a este. El modelo de dominio brinda una visión general de las entidades que existen en un determinado entorno, permitiendo comprender cómo está conformado el mismo, así como establecer una comunicación entre el cliente y el equipo de desarrollo. Para la comprensión del modelo de dominio se utiliza un glosario de términos, que describe todas las clases que lo componen. El glosario y el modelo de dominio ayudan a los desarrolladores, usuarios, clientes y otros interesados a utilizar un vocabulario común [19].

2.2.1 Descripción de las clases del modelo de dominio

- ✓ ***Usuario:*** Persona capacitada y autorizada para interactuar con el sistema.
- ✓ ***Reporte de consumo:*** Reportes que emite el sistema, contienen los tipos de áreas de consumo y las variables de consumo.
- ✓ ***Consumo estimado:*** Consumo del área docente que se calcula por medio de fórmulas matemáticas, estos están contenidos en los reportes de consumo.
- ✓ ***Gestión de área docente:*** Definida por el usuario, se refiere a la forma de estructurar las entidades docentes en la aplicación, mediante la inserción de los datos de las áreas que conforman esta última.
- ✓ ***Área de consumo:*** Áreas de consumo vigentes en las entidades docentes, se clasifican en áreas de laboratorios y áreas de oficina.
- ✓ ***Área laboratorio:*** Constituidas por laboratorios de docencia y producción.
- ✓ ***Área oficina:*** Constituidas por las oficinas de docencia y las oficinas no docentes.
- ✓ ***Consumidor eléctrico:*** Entidad que consume energía eléctrica, contiene diferentes variables de consumo se clasifican en activos fijos y en útiles.
- ✓ ***Variables consumo:*** Variables que permiten medir el consumo de la electricidad de los diferentes consumidores eléctricos:
    - o Demanda activa (kw) es el trabajo útil que se genera cuando comienza a funcionar un consumidor eléctrico, indica la cantidad de energía que se consume al instante.
    - o Demanda aparente (kva) es la potencia que subministran las plantas eléctricas cuando están trabajando sin ningún consumidor conectado.
    - o Demanda reactiva (kvar) es la potencia que necesitan los consumidores eléctricos para producir un campo magnético con el cual funcionan.
    - o Factor de potencia (fp) es un indicador de la eficiencia con que se está utilizando la energía eléctrica, para producir un trabajo útil.

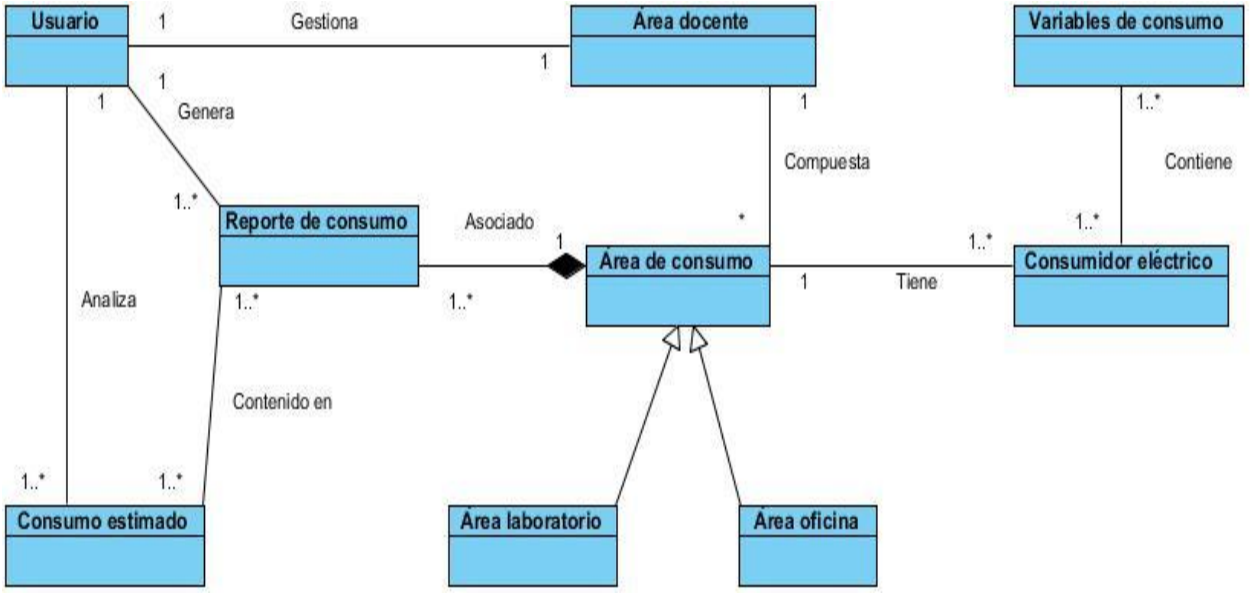

Figura 1: Modelo de Dominio del Sistema para la gestión de la información referente a la planificación y control del consumo energético.

2.3 Modelado del sistema

En cualquier proyecto de software los requisitos son las necesidades del producto que se debe desarrollar. La captura de los requisitos en el flujo de requerimientos es una de las principales actividades que se deben documentar en la fase de inicio de desarrollo del software.

Según la IEEE (acrónimo de Institute of Electrical and Electronics Engineers, en español: Instituto de Ingenieros Eléctricos y Electrónicos) el análisis de requisitos se puede definir como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, hardware o software, así como el proceso de estudio y refinamiento de los mismos. [19]

2.3.1 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir; permiten expresar específicamente las responsabilidades del sistema que se propone, determinar de una manera clara lo que el sistema debe hacer y no alteran las funcionalidades del producto. [19]

El sistema a desarrollar debe cumplir con los siguientes requisitos funcionales.

RF 1 Autenticar usuario.

RF 2 Gestionar datos del usuario.

RF 2.1 Adicionar datos del usuario.

RF 2.2 Modificar datos del usuario.

RF 2.3 Eliminar datos del usuario.

RF 2.4 Listar datos del usuario.

RF 3 Gestionar datos del local.

RF 3.1 Adicionar datos del local.

RF 3.2 Modificar datos del local.

RF 3.3 Eliminar datos del local.

RF 3.4 Listar datos del local.

RF 4 Gestionar datos del equipo.

RF 4.1 Adicionar datos del equipo.

RF 4.2 Modificar datos del equipo

RF 4.3 Eliminar datos del equipo.

RF 4.4 Listar datos del equipo.

RF 5 Gestionar datos del horario.

RF 5.1 Adicionar datos del horario.

RF 5.2 Modificar datos del horario.

RF 5.3 Eliminar datos del horario.

RF 5.4 Listar datos del horario.

RF 6 Gestionar datos del tipo de local.

RF 6.1 Adicionar datos del tipo de local.

RF 6.2 Modificar datos del tipo de local.

RF 6.3 Eliminar datos del tipo de local.

RF 6.4 Listar datos del tipo de local.

RF 7 Gestionar datos del tipo de equipo.

RF 7.1 Adicionar datos del tipo de equipo.

RF 7.2 Modificar datos del tipo de equipo.

RF 7.3 Eliminar datos del tipo de equipo.

RF 7.4 Listar datos del tipo de equipo.

RF 8 Administrar datos del reporte.

RF 8.1 Adicionar datos del reporte.

RF 8.2 Listar datos del reporte.

RF 9 Mostrar consumo estimado.

RF 9.1 Visualizar consumo estimado por tipo de local.

RF 9.2 Visualizar consumo general estimado.

RF10 Mostrar consumo real.

RF 10.1 Visualizar consumo real por fecha.

Rf 10.1 Visualizar consumo real del día.

RF 11 Mostrar gráfico de comparación de consumo.

RF 12 Mostrar gráfico del consumo por tipos de locales.

RF 13 Visualizar datos del reporte.

RF 14 Exportar datos del reporte.

2.3.2 Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener [21]. Para el desarrollo de la solución se identificaron como requisitos no funcionales:

Apariencia o interfaz externa

RNF1: El sistema debe tener una apariencia profesional, con un diseño gráfico sencillo que facilite la localización de las funciones del sistema.

Usabilidad

RNF2: El sistema brindará la posibilidad de ser usado por cualquier persona cuyos conocimientos en cuanto al trabajo con los ordenadores sean básicos, solo es necesario contar con conocimientos especializados en el consumo eléctrico para poder entender los resultados brindados por la aplicación.

RNF3: El sistema solo podrá ser utilizado por los usuarios creados por la administración. Cada usuario que se autentique en el sistema solo tendrá acceso a la información que le corresponde según su rol.

RNF4: La información deberá estar disponible en todo momento.

Hardware

RNF5: La computadora donde se desplegara la aplicación debe contar con una memoria RAM de 512mb o superior.

Software

RNF6: software requerido para desplegar la aplicación.

- ✓ Sistemas operativos Unix y Linux
- ✓ Servidor web apache2
- ✓ Lenguaje de programación php5.3
- ✓ Servidor de base de datos PostgreSQL versión 9.1
- ✓ Administrador de base de datos PGAdmin III.
- ✓ Usuario con privilegios de administración del SO.

RNF7 Software requerido para utilizar la aplicación.

- ✓ Sistemas operativos Unix y Linux
- ✓ Deberá contar con un navegador web

Soporte

RNF8: Un Sistema Gestor de base de datos con soporte para grandes volúmenes de datos y alta velocidad de procesamiento.

Disponibilidad

RNF9: El sistema deberá tener un 100% de disponibilidad por lo que podrá ser usado las 24 horas del día por todos sus clientes.

Requisitos Legales, de Derecho de Autor y otros

RNF10: Los derechos de autor serán registrados por la política que sigue la Universidad de las Ciencias Informáticas.

Seguridad

RNF11: El acceso a cada una de las funcionalidades del sistema estará restringido en dependencia del rol del usuario que acceda al mismo. El uso de un usuario y una contraseña será siempre obligatorio para acceder a los servicios de la aplicación.

RNF12: La conexión al sistema debe ser mediante el protocolo de seguridad HTTPS.

2.3.3 Definición de los actores del sistema

Los actores del sistema representan terceros fuera del sistema que colaboran con él. Estos pueden representar el rol que juega una o varias personas, un equipo o un sistema. Luego de definir qué es un actor, se definirán los actores de la aplicación en cuestión [20].

Tabla 1: Definición de los actores del sistema.

Actor(es)	Descripción
Administrador	Representa a la persona que interactúa con el sistema para realizar todas las operaciones de administración.
Usuario	Representa a los usuarios que interactúan con el sistema y realizan las operaciones de acuerdo al rol que tengan asignado en el sistema.
Especialista	Persona que interactúa con el sistema para analizar la información del consumo eléctrico que brida la aplicación.

2.3.3 Diagrama de casos de uso del sistema

El diagrama de casos de uso (CU) del sistema diseñado muestra la relación existente entre los actores y los casos de uso del sistema [20]. El actor administrador realiza las principales funcionalidades del sistema como la gestión de los locales, equipos, horarios, reportes y usuarios, las cuales permiten

controlar el consumo de la electricidad. Este consumo será visualizado por los actores que intervengan en el sistema dependiendo del rol. Por su parte el actor especialista podrá visualizar el consumo de la electricidad estimado y el consumo real que mide el analizador de la entidad.

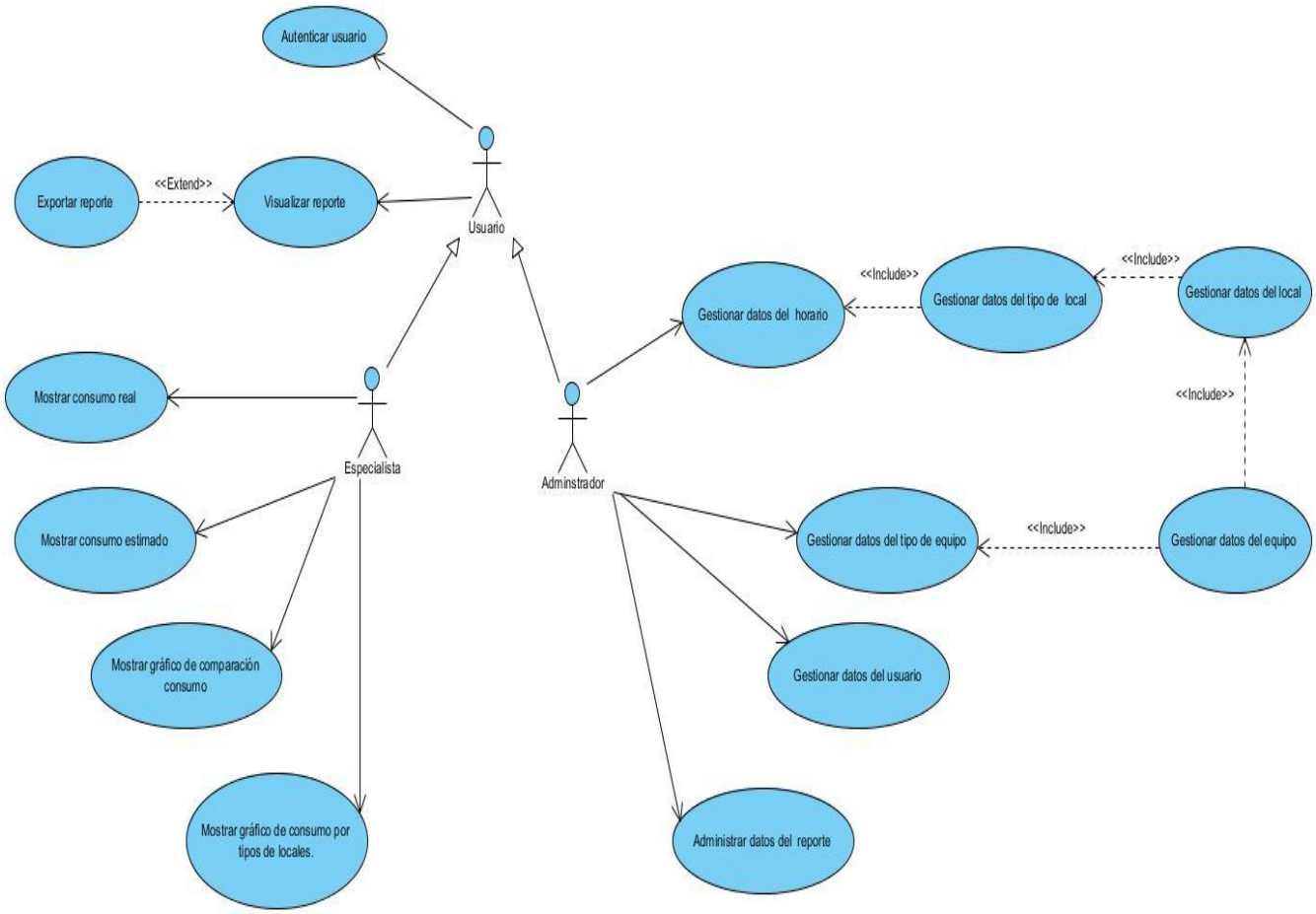

Figura 2: Diagrama de casos de uso del sistema

2.3.4 Descripción textual de casos de uso del sistema

Un caso de uso define la secuencia de transacciones desarrolladas por un sistema en respuesta a un evento que inicia un actor al interactuar con él [20].

A continuación, se muestra la especificación del caso de uso Gestionar datos del equipo, se describe el flujo de eventos que realiza el actor administrador al interactuar con el sistema. La totalidad de estas descripciones se encuentran en el artefacto Especificación de Casos de Uso dentro del Expediente de Proyecto.

Tabla 2: Descripción de caso de uso Gestionar datos del equipo.

Nombre del Caso de Uso	Gestionar datos del equipo.
Actores	Administrador.
Objetivo	Gestionar los equipos existentes en el sistema.
Resumen	El caso de uso inicia cuando el administrador desea realizar una de las siguientes opciones: ✓ Adicionar datos del equipo: se inicia cuando el administrador introduce los datos de un nuevo equipo y finaliza cuando el sistema muestra los datos del mismo. ✓ Modificar datos del equipo: se inicia cuando el administrador selecciona un equipo y modifica los datos y finaliza cuando el sistema muestra los datos modificados. ✓ Eliminar datos del equipo: se inicia cuando el administrador selecciona un equipo y lo elimina finalizando así el CU. ✓ Listar datos del equipo: se inicia cuando el administrador entra a la sección de administración de los equipos y se muestra el listado de los mismos, si el administrador desea eliminar los equipos del listado, selecciona los equipos a eliminar y el sistema elimina los equipos. Finaliza así el CU.
Complejidad	Alta
Prioridad	Alta
Precondiciones	El sistema debe estar disponible. El administrador tiene que estar autenticado.
Postcondiciones	En dependencia de la acción del administrador: ✓ Se adicionan los datos de un equipo. ✓ Se modifican los datos de un equipo. ✓ Se elimina un equipo de la base de datos. ✓ Se listan los datos de un equipo.
Flujo de eventos	
Flujo básico Gestionar equipo	

	Actor	**Sistema**
1.	El administrador selecciona la opción “Equipo”.	
2.		El sistema muestra la vista principal de la sección de administración de equipos.
3.	El administrador selecciona la opción que desea realizar: ✓ Adicionar datos del equipo.	El administrador puede: ✓ Adicionar datos del equipo. Ir a la

	✓ Modificar datos del equipo. ✓ Eliminar datos del equipo. ✓ Listar datos del equipo.	sección: Adicionar datos del equipo. ✓ Modificar datos del equipo. Ir a la sección: Modificar datos del equipo. ✓ Eliminar datos del equipo. Ir a la sección: Eliminar datos del equipo. ✓ Listar datos del equipo. Ir a la sección: Listar datos del equipo.
Sección 1: "Adicionar datos del equipo"		
Flujo básico Adicionar datos del equipo		
	Actor	**Sistema**
1.	El Administrador selecciona la opción "Nuevo" que aparece en la ventana principal.	
2.		El sistema muestra en la interfaz los datos para adicionar un equipo: nombre, código, descripción, kw, kva, kvar, fp, activo, la lista desplegable para seleccionar un local y un tipo de equipo.
3.	El Administrador introduce los datos correspondientes y da clic en el botón "Publicar".	
4.		El sistema guarda los cambios en la base de datos y muestra en la sección ¨Listar¨ el equipo insertado. Finaliza el CU.
Sección 2: "Modificar datos del quipo"		
Flujo básico Modificar datos del equipo		
	Actor	**Sistema**
1.	El Administrador selecciona la opción "Editar" del equipo a modificar que aparece en la ventana principal.	
2.		El sistema muestra en la interfaz los datos para modificar un equipo: nombre, código, descripción, kw, kva, kvar, fp, activo, la lista desplegable para seleccionar un local y un tipo de equipo.
3.	El Administrador modifica los datos correspondientes y da clic en el botón "Editar".	
4.		El sistema guarda los cambios en la base

		de datos y muestra en la sección ¨Listar¨ el equipo actualizado. Finaliza el CU.
Sección 3: "Eliminar datos del quipo"		
Flujo básico Eliminar datos del equipo		
	Actor	**Sistema**
1.	El Administrador selecciona la opción "Mostrar" que aparece en la ventana principal.	
2.		El sistema muestra en la interfaz los datos de un equipo: nombre, código, descripción, kw, kva, kvar, fp, activo, el tipo de local y el tipo de equipo.
3.	El Administrador da clic en el botón "Eliminar".	
4.		El sistema elimina el equipo de la base de datos y muestra la interfaz con el listado de los equipos restantes. Finaliza el CU.
Sección 4: "Listar datos del equipo"		
Flujo básico Listar datos del equipo		
	Actor	**Sistema**
1.	El Administrador selecciona la opción Listar equipo que aparece en la ventana principal.	
2.		Muestra el listado de los equipos existentes en la base de datos.
3.	El administrador no desea eliminar el equipo.	
4.		Finaliza el CU.
Flujos alternos Sección 1		
3. El administrador inserta los datos de forma incorrecta.		
	Actor	**Sistema**
3.1	Si el administrador deja campos en blanco.	El sistema muestra el mensaje de error " Por favor, rellene este campo". Ir al paso 3.
3.2	Si el administrador inserta datos inválidos.	El sistema muestra el mensaje de error" Por favor, ajústese al formato solicitado". Ir al paso 3.

Flujos alternos Sección 2		
3. El administrador modifica los datos de forma incorrecta.		
	Actor	**Sistema**
3.1	Si el administrador deja campos en blanco.	El sistema muestra el mensaje de error" Por favor, rellene este campo". Ir al paso 3.
3.2	Si el administrador inserta datos inválidos.	El sistema muestra el mensaje de error" Por favor, ajústese al formato solicitado". Ir al paso 3.

Flujos alternos Sección 4		
3.	El administrador desea eliminar los equipos del listado.	
	Actor	**Sistema**
3.1	El administrador selecciona los equipos del listado y elimina.	El sistema elimina de la base de datos los equipos seleccionados. Ir al paso 4.

Relaciones	**CU Incluidos**	No aplica.
	CU Extendidos	No aplica.
Referencias cruzadas	RF 5.1, RF 5.2 , RF 5.3 , RF 5.4	

2.4 Diseño del sistema

El objetivo del diseño es modelar el sistema, encontrar su forma para que soporte todos los requisitos. Por otro lado, el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. A continuación, se desarrolla el diseño de la solución; para ello se define el estilo arquitectónico a utilizar en el sistema y se modelan los artefactos de diagrama de clases del diseño y diagrama de iteración.

2.4.1 Estilo arquitectónico

El desarrollo del sistema se basó en una arquitectura orientada a objetos y el estilo arquitectónico utilizado es de llamada-retorno que encapsula la arquitectura en capas y tiene como objetivo fundamental la separación de la lógica de negocios de la lógica de diseño. Dicho estilo posee como ventaja fundamental, que el desarrollo se puede llevar a cabo en varios niveles.

El desarrollo del *software* se basó en patrón arquitectónico Modelo Vista Controlador (Model-View-Controller - MVC), el cual sigue la filosofía del estilo arquitectónico de llamada-retorno que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. La capa del modelo representa la información con la que trabaja la aplicación, es decir, la lógica de negocio, la vista transforma el modelo en una página web que permite al usuario interactuar con ella y el controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista [20].

En la aplicación el cliente envía una señal llamada petición, esta es interceptada por el controlador que realiza las validaciones necesarias, procesamientos de los datos y lógica de negocio asociadas a esa petición del cliente. El controlador envía datos a la capa del modelo, por ejemplo cuando necesitan ser guardados en la base de datos y los obtiene dependiendo de la solicitud del usuario para finalmente enviarlos a la capa de la vista a fin de ser mostrados nuevamente al cliente a través de una respuesta.

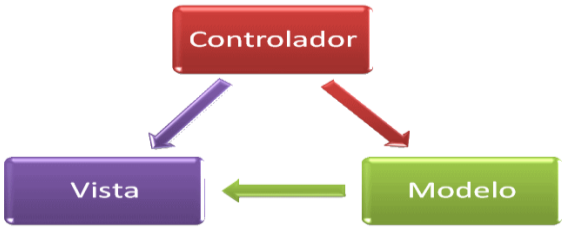

Figura 3: Representación del patrón arquitectónico MVC.

2.4.2 Patrones de diseño.

Los patrones brindan la facilidad de reutilizar el conocimiento de desarrolladores, clasificando y describiendo problemas y soluciones a problemas que surgen con frecuencia durante el desarrollo, por lo que se convierten en un historial que ayuda a no cometer errores ya descritos [20].

En la solución del sistema principalmente se utilizan los siguientes patrones de diseño GRASP y GOF

Patrones GRASP:

Bajo Acoplamiento*:* Es la idea de tener las clases lo menos relacionadas entre sí que se pueda. Por ejemplo las clases del controlador no afectan el modelo al ser modificadas ya que estas son acciones independientes, estas acciones pueden ser reutilizadas desde otros módulos y aplicaciones en el sistema. En la Figura 4 se evidencia la existencia de este patrón, las acciones de la clase EquipoController al ser modificadas no afectan a las funcionalidades de la clase Equipo, estas están separadas en diferentes paquetes que permiten este bajo acoplamiento.

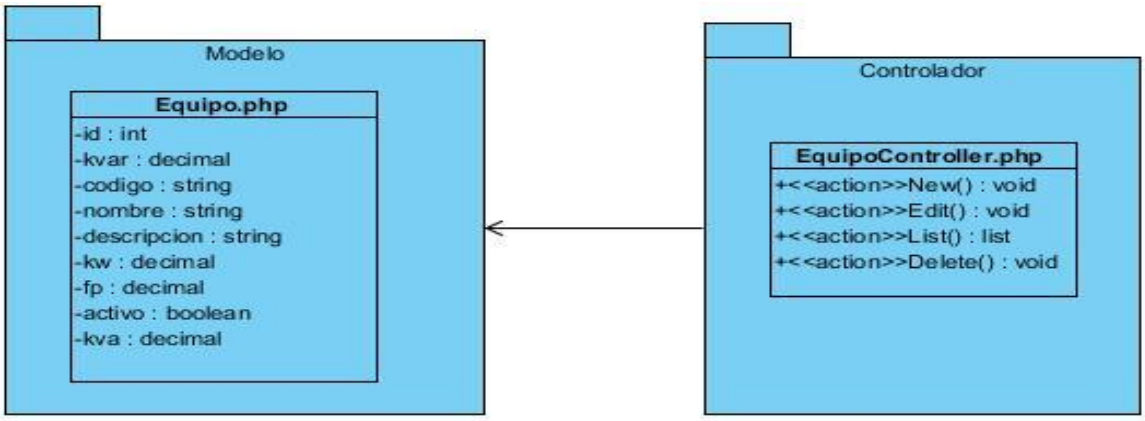

Figura 4: Uso del patrón Bajo acoplamiento en el diseño del sistema.

Controlador: Es un objeto de interfaz no destinado al usuario que se encarga de manejar un evento del sistema, definiendo además el método asignado al evento del usuario. Coordina y controla el trabajo que se necesita hacer en la aplicación. Este patrón se evidencia en las clases del paquete controlador, donde estas realizan las funcionalidades correspondientes a las peticiones que envían las clases del paquete de la vista. La clase EquipoController es la encargada de manejar todas las acciones referentes al caso de uso GestionarEquipo.

Controlador Frontal: Todas las peticiones web son manejadas por el script app.php el cual se conoce como controlador frontal, que es el único punto de entrada de toda la aplicación. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la dirección URL entrada por el usuario. En la Figura 5 se observa el uso de este patrón, donde la clase app.php recibe todas las peticiones e identifica el controlador asociado a la petición. Como es el caso de las peticiones que se envían a la clase EquipoController.

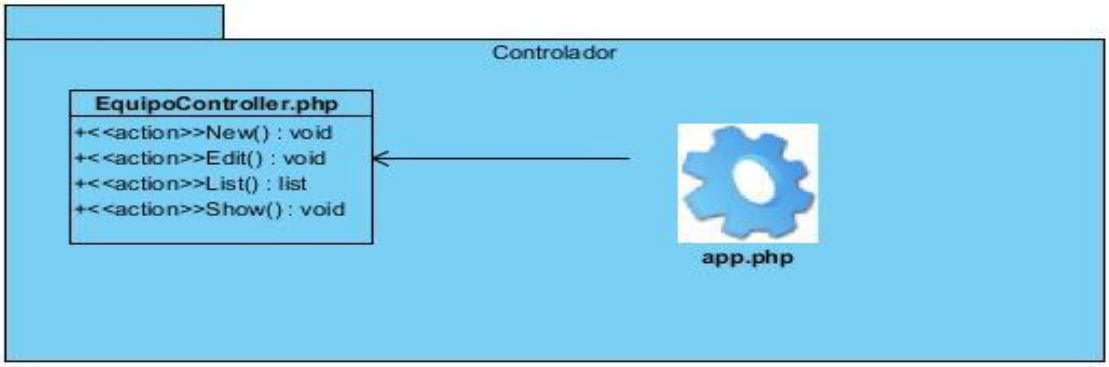

Figura 5: Uso del patrón Controlador en el diseño del sistema.

Experto: Este patrón brinda la posibilidad de asignar una responsabilidad al experto en información, es decir, asignar dicha responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Se sigue al incluir el ORM Doctrine donde se generan las clases para la gestión de las entidades con las responsabilidades debidamente asignadas.

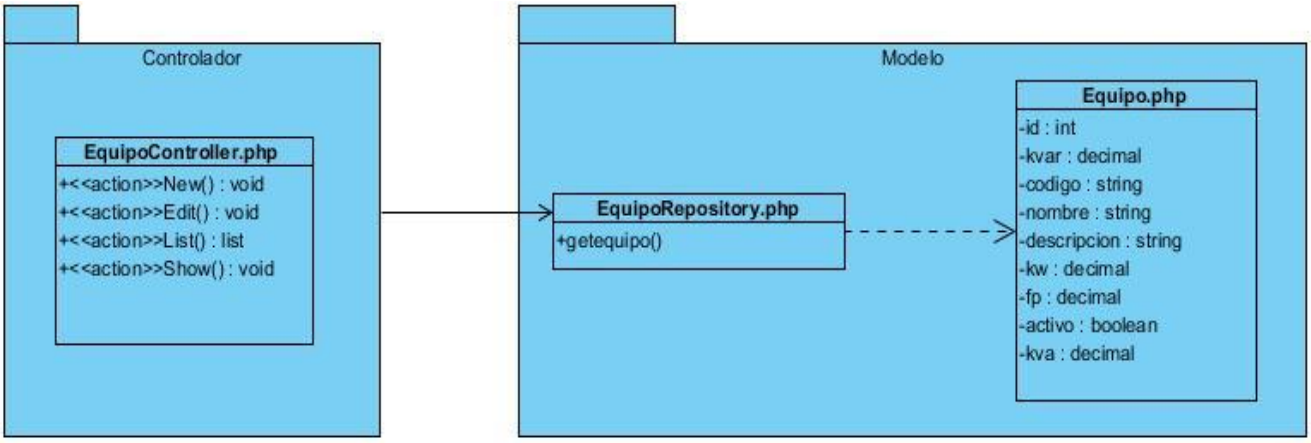

Figura 6: Uso del patrón Experto en el diseño del sistema.

En la Figura 6 se evidencia su uso cuando el usuario realiza una acción correspondiente al caso de uso GestionarEquipo, donde el controlador se auxilia de la clase EquipoRepository, esta última es la especialista en información, la misma cuenta con las funcionalidades necesarias para dar cumplimiento a dicha acción.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Este patrón se evidencia en la clase TipoLocalController, donde se crean las instancias de los reportes, que posteriormente serán adicionados a los tipos de locales que les corresponden. Se brinda un soporte al bajo acoplamiento.

Patrones GOF:

Decorator: Este patrón se refleja en la implementación del archivo o componente layout.html.twig, se caracteriza por ser la plantilla global en todas las interfaces de la aplicación que almacena el código .html.twig que es común en ellas, evitando tener que repetirlo en cada página. El uso de este patrón se observa en las clases AdminLayout.html.twig y UserLayout.html.twig que representan las plantillas globales que contienen las secciones creadas para los actores principales del sistema.

Comando: El patrón comando permite que solicitudes del cliente sean encapsuladas como objetos, indicar parámetros a diferentes solicitudes, encolarlas, registrarlas y hasta dar soporte para operaciones. Este se aplica cuando los controladores reciben las acciones realizadas por los usuarios mediante objetos de tipo *request*, estos últimos encapsulan las peticiones de los usuarios. Este patrón se aplica en la clase EquipoController, encargada de recibir las peticiones de los usuarios referente al caso de uso GestionarEquipo, mediante los objetos *request*.

2.4.3 Diagrama de clases del diseño

Un diagrama de clases del diseño web permite describir gráficamente un conjunto de clases, así como

las relaciones entre ellas, logrando de esta forma una muestra del sistema importante para la implementación. Para el diseño del sistema se modelaron los diagramas de clases del diseño por casos de uso como lo indica la metodología. Se han agrupado las clases de acuerdo a su funcionalidad en tres grandes paquetes, haciendo además referencia al patrón arquitectónico expuesto anteriormente: Modelo Vista Controlador (MVC). A continuación, se muestran el diagrama de clases del diseño del caso de uso Gestionar datos del equipo, en este se invidencia cómo interactúan las capas del patrón arquitectónico MVC. El resto de estos diagramas se pueden consultar en el artefacto Modelo de diseño en el Expediente de Proyecto.

2.4.3.1 Diagrama de clases del diseño del caso de uso Gestionar datos del equipo.

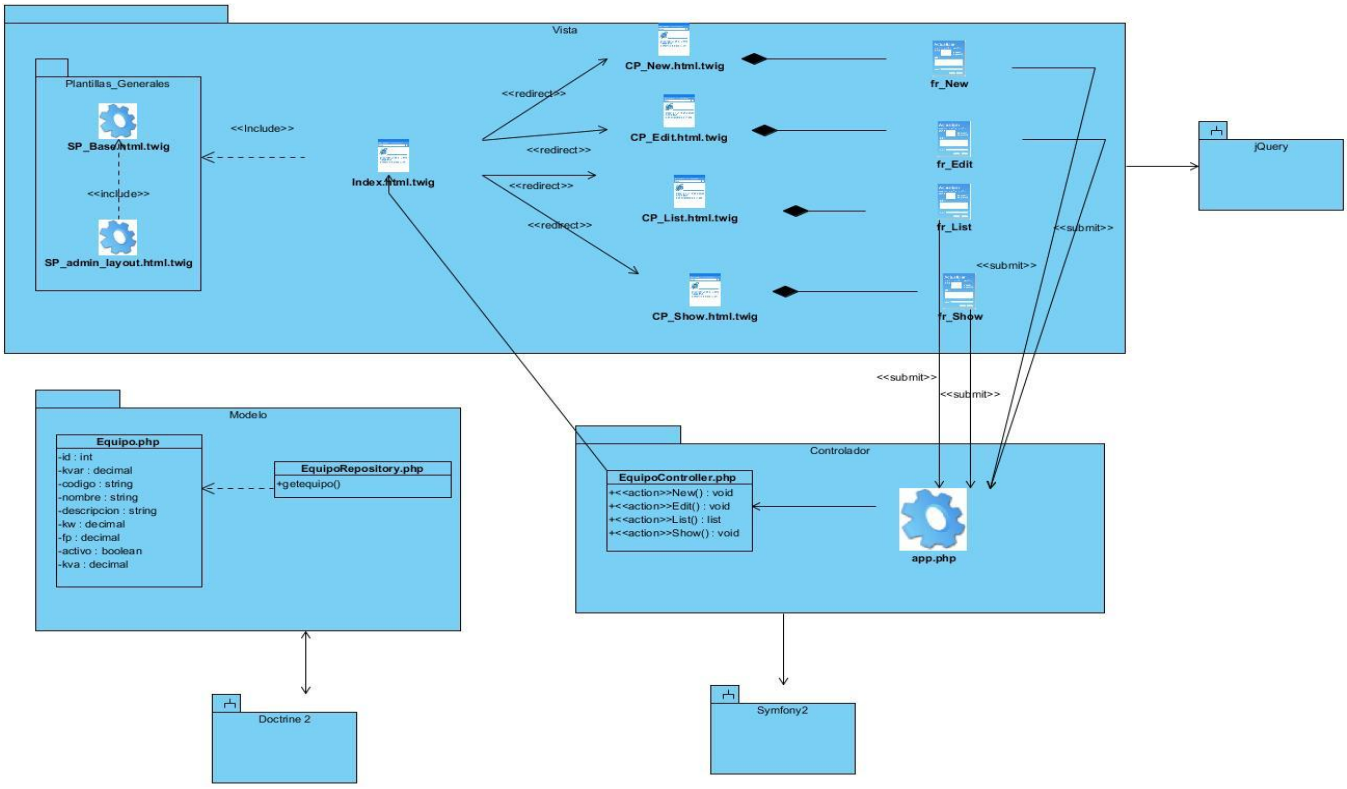

Figura 7: Diagrama de clases del diseño del caso de uso Gestionar datos del equipo.

2.4.3.2 Descripción de las clases del caso de uso Gestionar datos del equipo.

A continuación, se describen las principales clases del diagrama de clases del diseño del caso de uso Gestionar datos del equipo. Estas clases están organizadas bajo el patrón arquitectónico MVC antes explicado.

EquipoController: Es la clase principal de la capa del controlador, recibe todas las peticiones que realiza el administrador del paquete de la vista para luego procesarlas. Contiene todas las funcionalidades que permiten gestionar los datos del equipo.

App: Esta clase pertenece a la capa del controlador, es denominada como el controlador frontal y es el

único punto de entrada de la aplicación.

Index: En esta clase se muestran todas las acciones que se realizan para la gestión de un equipo. Agrupa todo el contenido HTML, CSS y jQuery del cual extienden las funcionalidades del caso de uso.

New: Esta clase muestra una interfaz que contiene los campos donde se adicionan los datos de un equipo. Es la encargada de enviar las acciones que ejecuta un usuario cuando adiciona los datos de un equipo, las cuales son recibidas por la clase controladora EquipoController.

Edit: Esta clase muestra una interfaz que permite visualizar los diferentes datos que conforman un equipo. Es la encargada de modificar los datos de un equipo seleccionado, responde a la clase controladora EquipoController donde ejecuta esta acción.

List: Esta clase permite listar todos los equipos adicionados en el sistema, contiene botones de acceso a las funcionalidades modificar, mostrar y eliminar los datos de los equipos. Responde a la clase controladora EquipoController donde se ejecuta esta petición.

Show: Esta clase muestra los datos de un equipo seleccionado, contiene la funcionalidad que permite eliminar el equipo en cuestión, la cual responde a la clase controladora EquipoController donde se ejecuta esta funcionalidad.

AdminLayout: Está diseñada como clase principal de la sección de administración, permite que se acceda a las diferentes funcionalidades antes explicadas. Agrupa todo el contenido HTML, CSS y jQuery del cual extienden las funcionalidades del caso de uso.

Base: Esta clase es la plantilla base de toda la aplicación, la genera el *framework* con el fin de agrupar todos los elementos que se repiten en todas las páginas:<html>, <head>, <body>, <title>, <footer>.

Equipo: Esta clase responde a la capa del modelo, contiene todos los atributos de un equipo que se encuentran en la base de datos.

2.4.4 Diagrama de Interacción del sistema.

Diagramas de secuencia

Los diagramas de interacción muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas, por lo que son empleados para modelar aspectos dinámicos del sistema. Los diagramas de colaboración y de secuencia son dos tipos de diagramas de interacción que son semánticamente equivalentes pero sin embargo los diagramas de colaboración destacan el orden estructural de los objetos que interactúan y los de secuencia destacan el orden temporal de los mensajes. Para la realización de los casos de uso del diseño es más factible el empleo de los diagramas de secuencia ya que representan con más claridad el flujo de las acciones que debe realizar el sistema. Para el desarrollo de la aplicación se diseñó un diagrama de secuencia por cada escenario de los casos uso, a continuación se muestran los diagramas de secuencia correspondientes al caso de uso Gestionar datos del local. El resto de los diagramas se encuentran en el artefacto

Modelo de diseño en el Expediente de Proyecto.

2.4.4.1 Diagrama de secuencia del caso de uso Gestionar datos del equipo. Escenario: Adicionar datos del equipo.

El diagrama que se muestra en la Figura 8 describe el flujo de eventos del escenario Adicionar datos del equipo. El administrador accede a la página principal CP_Index y selecciona la opción Adicionar datos del equipo. Luego, el sistema muestra la interfaz correspondiente a la opción seleccionada. El administrador inserta los datos de un equipo: nombre, código, descripción, kw, kva, kvar, fp y activo. El sistema verifica en la clase controladora EquipoController que los datos del equipo entrados estén correctos y cumpla con las restricciones especificadas. Termina el escenario cuando muestra equipo seleccionado.

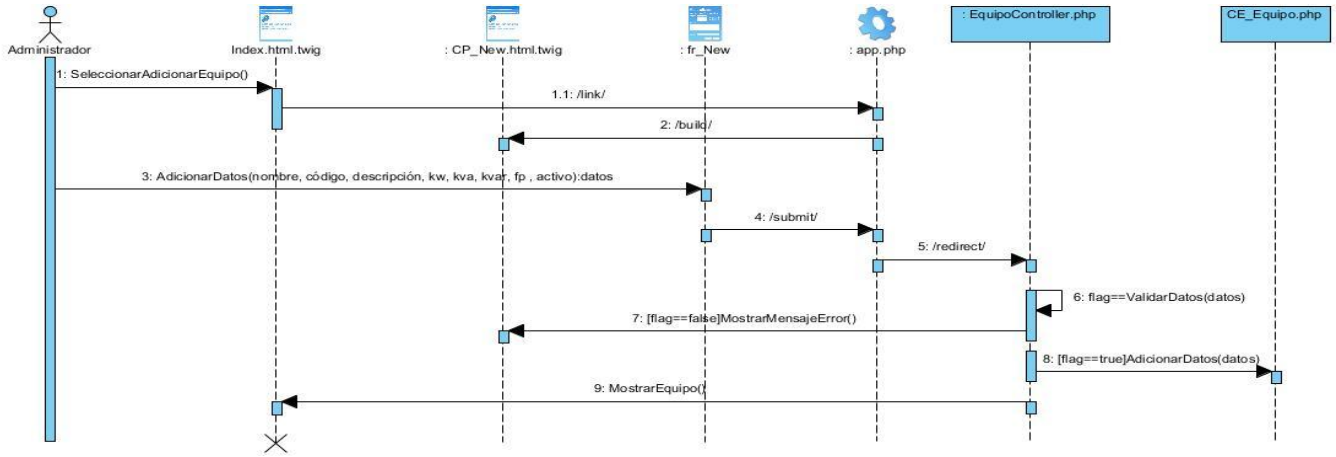

Figura 8: Diagrama de secuencia del escenario Adicionar datos del equipo del caso de uso Gestionar datos del equipo.

2.4.4.2 Diagrama de secuencia del caso de uso Gestionar datos del equipo. Escenario Modificar datos del equipo.

El diagrama que se muestra en la Figura 9 describe el flujo de eventos del escenario Modificar datos del equipo. El administrador accede a la página principal CP_Index y selecciona la opción modificar equipo. Luego, el sistema muestra la interfaz correspondiente a la opción seleccionada. El administrador selecciona el equipo e introduce los datos a modificar: nombre, código, descripción, kw, kva, kvar, fp y activo. El sistema verifica en la clase controladora EquipoController que los datos entrados estén correctos y cumpla con las restricciones especificadas. Termina el escenario cuando muestra el equipo con los datos modificados.

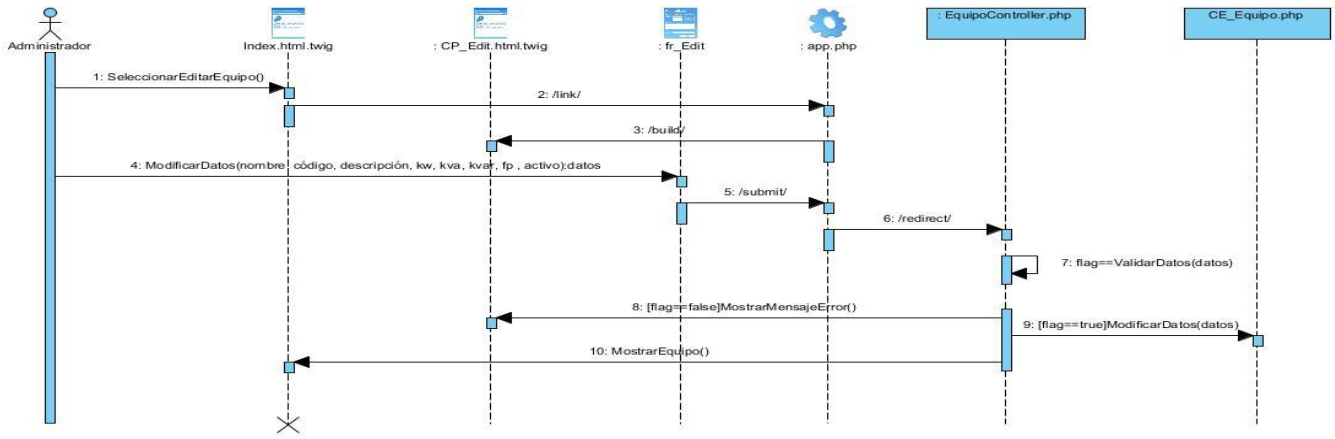

Figura 9: Diagrama de secuencia del escenario Modificar datos del equipo del caso de uso Gestionar datos del equipo.

2.4.4.3 Diagrama de secuencia del caso de uso Gestionar datos del equipo. Escenario Eliminar datos del equipo.

El diagrama que se muestra en la Figura 10 describe el flujo de eventos del escenario Eliminar datos del equipo. El administrador accede a la página principal CP_Index y selecciona la opción Mostrar equipo. Luego, el sistema muestra la interfaz correspondiente a la opción seleccionada. El administrador da clic en el botón eliminar. El sistema envía la petición a la clase controladora EquipoController y se elimina el equipo. Termina el escenario cuando se muestra una interfaz con el resto de los equipos listados.

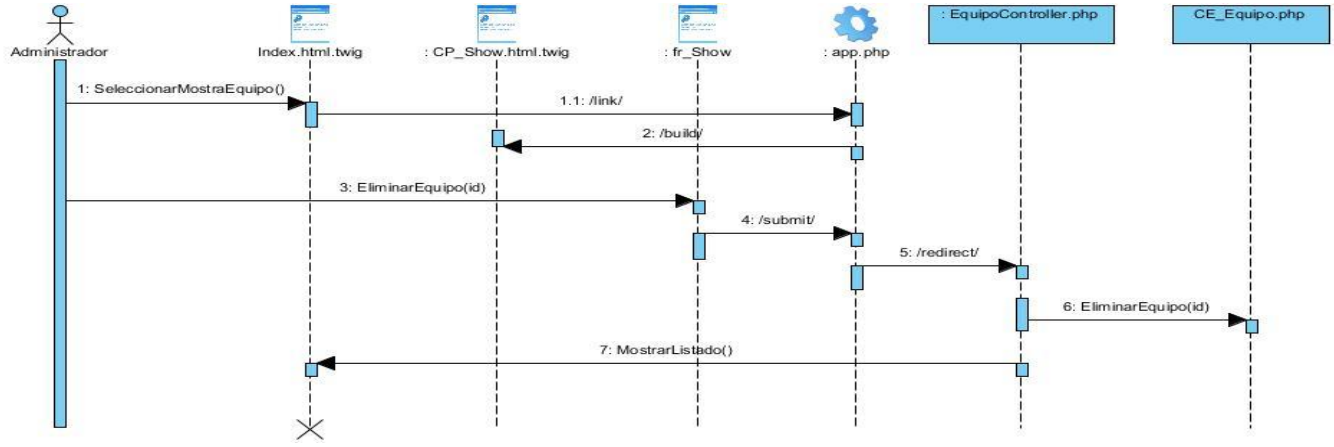

Figura 10: Diagrama de secuencia del escenario Eliminar datos del equipo del caso de uso Gestionar datos del equipo.

2.4.4.4 Diagrama de secuencia del caso de uso Gestionar datos del equipo. Escenario Listar datos del equipo.

El diagrama que se muestra a continuación, describe el flujo de eventos del escenario Listar datos del equipo. El administrador accede a la página principal CP_Index y selecciona la opción equipo. Luego,

el sistema muestra en la interfaz correspondiente a la opción seleccionada el listado de los equipos adicionados en el sistema como se muestra en la Figura 11. En esta sección si el administrador desea eliminar un equipo, selecciona la opción deseada y el sistema verifica en la clase controladora EquipoController que el equipo se elimine correctamente. Termina el escenario cuando se muestra la página principal con el resto de los equipos listados, este flujo se visualiza en la Figura 12.

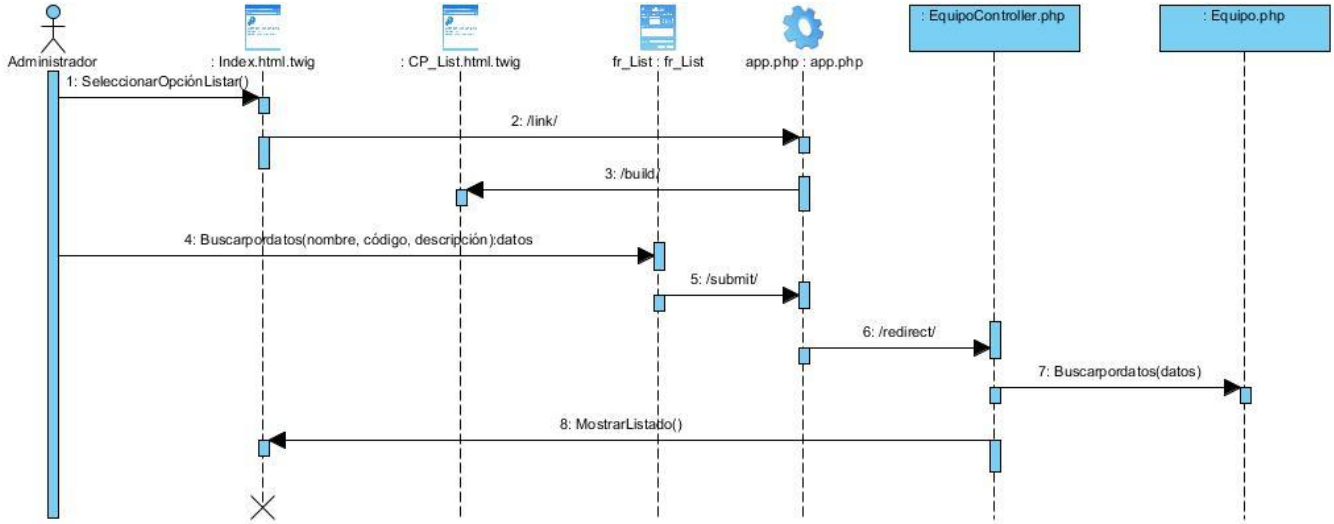

Figura 11*:* Diagrama de secuencia del caso de uso Gestionar datos del equipo. Escenario Listar datos del equipo**.**

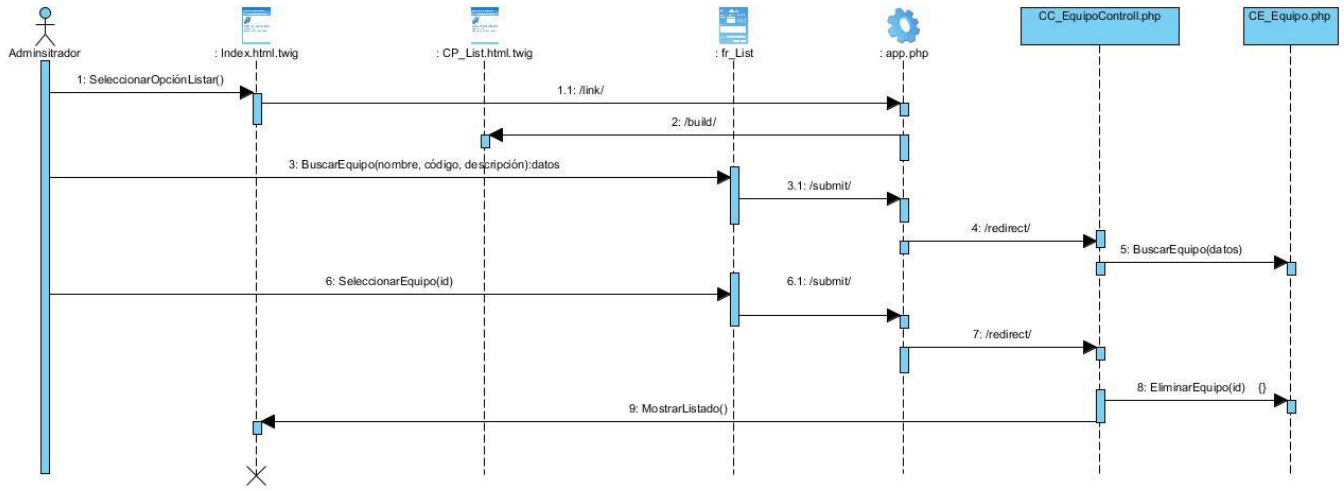

Figura 12: Diagrama de secuencia del caso de uso Gestionar datos del equipo. Escenario Listar datos del equipo. Flujo alterno de la sección.

2.5 Modelo de Datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas que gestionan y muestran información.

Un modelo de datos se define como un lenguaje orientado a describir una base de datos. Típicamente un modelo de datos permite describir:

- ✓ Las estructuras de datos de la base: el tipo de los datos que hay en la base y la forma en que se relacionan.
- ✓ Las restricciones de integridad: un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- ✓ Operaciones de manipulación de los datos: operaciones de agregado, borrado, modificación y recuperación de los datos de la base [20].

Para el diseño de la base de datos de la aplicación, se desarrolló el modelo de datos basándose en la filosofía del modelo entidad relación, el cual es una técnica de análisis que se apoyada en la identificación de las entidades y las relaciones que se generan entre ellas. Este modelo permite representar de forma abstracta los datos que se pretenden almacenar en la base de datos. A continuación, se muestra cómo quedó estructurado el diseño de la base de datos de la solución.

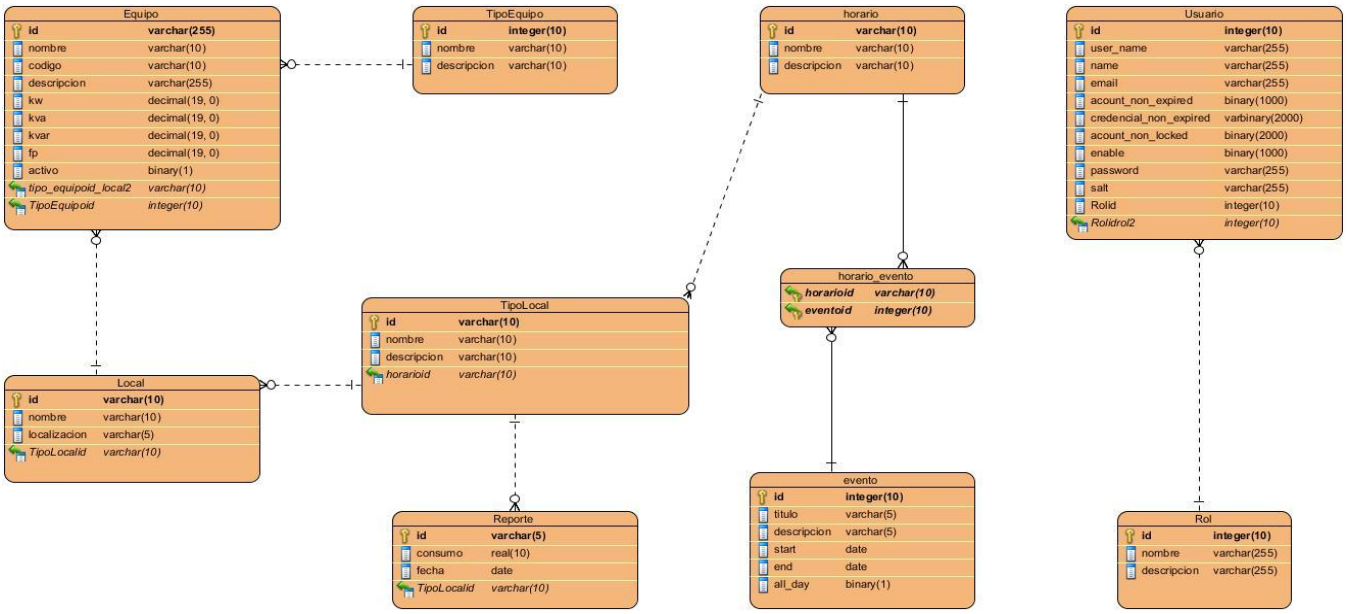

Figura 13: Diagrama entidad relación del Sistema para la gestión de la información referente a la planificación y control del consumo energético.

Para un mejor entendimiento de las clases que conforman el modelo entidad relación, a continuación, se muestra la descripción de sus atributos.

Descripción de las tablas de la base de datos.

Tabla 3: Descripción de la tabla Equipo.

Nombre	**Equipo**	
Descripción	Almacena los datos de los equipos.	
Atributo	**Tipo**	**Comentario**
id	Varchar	Llave primaria.
nombre	Varchar	Nombre del equipo.
código	Varchar	Código del equipo.
descripcion	Varchar	Se describe el equipo
kw	Decimal	Consumo de un equipo en kilovatio (kw).
kva	Integer	Consumo de un equipo por la potencia activa (KVA).
kvar	Integer	Consumo de un equipo por el factor de potencia reactiva (KVAR)
fp	Integer	Consumo de un equipo por el factor de potencia (FP).
activo	Binary	Se especifica si el equipo está activo.

Tabla 4: Descripción de la tabla TipoEquipo**.**

Nombre	**TipoEquipo**	
Descripción	Representa el tipo de un equipo.	
Atributo	**Tipo**	**Comentario**
id	Integer	Llave primaria.
nombre	Varchar	Nombre del tipo de quipo.
descripcion	Varchar	Se describe el tipo de quipo

Tabla 5: Descripción de la tabla Local.

Nombre	**Local**	
Descripción	Almacena los locales del docente.	
Atributo	**Tipo**	**Comentario**
id	Varchar	Llave primaria.
nombre	Varchar	Nombre del local.
localización	Varchar	Se especifica la ubicación del local en el área docente.

Tabla 6: Descripción de la tabla TipoLocal.

Nombre	**TipoLocal**	
Descripción	Representa el tipo de un local.	
Atributo	**Tipo**	**Comentario**
id	Varchar	Llave primaria.
nombre	Varchar	Nombre del tipo de local.
descripcion	Varchar	Se describe el tipo de local.

Tabla 7: Descripción de la tabla Reporte**.**

Nombre	**Reporte**	
Descripción	Almacena los reportes del consumo de la electricidad.	
Atributo	**Tipo**	**Comentario**
id	Varchar	Llave primaria.
consumo	Array	Se especifica el consumo por reporte.
fecha	Date	Se especifica la fecha del reporte.

Tabla 8: Descripción de la tabla Horario**.**

Nombre	**Horario**	
Descripción	Especifica el horario del consumo de los locales.	
Atributo	**Tipo**	**Comentario**
id	Varchar	Llave primaria.
nombre	Varchar	Nombre del horario.
descripcion	Varchar	Se describe información relevante de un horario en específico.

Tabla 9: Descripción de la tabla Evento.

Nombre	**Evento**	
Descripción	Especifica las horas del día por el horario creado.	
Atributo	**Tipo**	**Comentario**
id	Integer	Llave primaria.
titulo	Varchar	Se refiere al nombre del evento.
descripcion	Varchar	Se describe información relevante del evento.
start	Date	Se especifica el inicio del evento.
end	Date	Se especifica el fin del evento.

all_day	Binary	Se especifica si el evento estará todo el día o no.

Tabla 10: Descripción de la tabla Usuario

Nombre	**Usuario**	
Descripción	Tabla que contiene los datos de un usuario del sistema.	
Atributo	**Tipo**	**Comentario**
id	Integer	Llave primaria.
user_name	Varchar	Nombre del actor.
name	Varchar	Nombre de la persona.
email	Varchar	Correo del usuario.
acount_non_expired	Binary	Variable que tratan el tiempo de expiración de la contraseña
credencial_non_expired	Binary	Variable que tratan el tiempo de expiración de la cuenta
acount_non_locked	Binary	Variable que verifica que la cuenta tenga un bloqueo.
enable	Binary	Especifica si está habilitado.
password	Varchar	Se especifica la contraseña de autenticación del usuario.
salt	Varchar	Se especifica que la contraseña de autenticación del usuario sea comprobada.

Tabla 11: Descripción de la tabla Rol.

Nombre	**Rol**	
Descripción	Tabla que contiene los datos de un rol del sistema.	
Atributo	**Tipo**	**Descripción del atributo**
id	Integer	Llave primaria.
nombre	Varchar	Nombre del rol.
descripcion	Varchar	Se describe la información relevante del rol.

2.5 Conclusiones del capítulo

En el presente capítulo se realizaron los principales artefactos del análisis y el diseño: especificación requisitos y casos de uso, diagrama de clases del diseño, diagrama de secuencia y diagrama entidad relación. Se definió y describió el patrón arquitectónico MVC, que sigue la filosofía del estilo arquitectónico llamada y retorno antes explicado. En el desarrollo de la aplicación se utilizaron los patrones de diseño, entre los GRASP prevalecen bajo acoplamiento, controlador, controlador frontal, experto y creador y entre los GOF decorator y comando. En general se ganó claridad en cuanto a la adjudicación del sistema implementado y se sentaron las bases para las fases de implementación y prueba.

CAPÍTULO 3: IMPLEMENTACIÓN DEL SISTEMA

Los artefactos generados y presentados en el capítulo anterior, ayudan al programador a interpretar las funcionalidades que propone el cliente, y por tanto ser capaz de implementar el sistema. En este capítulo se aborda el flujo de trabajo de implementación, donde se describen los principales artefactos generados por el rol de implementador, destacando el modelo de implementación que incluye componentes, subsistemas de implementación y diagramas de componentes. En este flujo de trabajo se comienza a implementar el sistema en términos de componentes a partir de las clases del diseño. Además, se definen los estándares de codificación necesarios para asegurar una mejor calidad y compresión del código.

3.1. Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre estos.

3.1.1. Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en función de subsistemas y mostrar las relaciones entre los elementos. Se utiliza para modelar la vista estática de un sistema. Muestra la organización y dependencias lógicas entre un conjunto de componentes de *software*, sean estos componentes de código fuente, librerías, binarios o ejecutables [20].

Para el desarrollo de la aplicación se diseñan diagramas de componentes asociados a los casos de uso como lo indica Open UP. Donde se evidencia la distribución de las clases según el patrón arquitectónico MVC que utiliza Symfony2 como paradigma de su organización interna.

A continuación, se muestra el diagrama de componentes del caso de uso Gestionar datos del equipo, donde el subsistema Vista recoge todas las clases del CU encargadas de mostrar y manejar todo lo referente a las interfaces de los usuarios, así como el tratamiento de las peticiones realizadas por estos últimos. Estos componentes se apoyan del subsistema jQuery para la visualización de los datos de los equipos insertados. El subsistema de implementación Controlador agrupa los componentes encargados de realizar todos los métodos y operaciones que serán mostrados en las interfaces de la vista. El paquete Modelo contiene las clases que representan las entidades de la base de datos, estas clases son convertidas por el subsistema Doctrine en las tablas de la base de datos, mediante el mapeo de objetos.

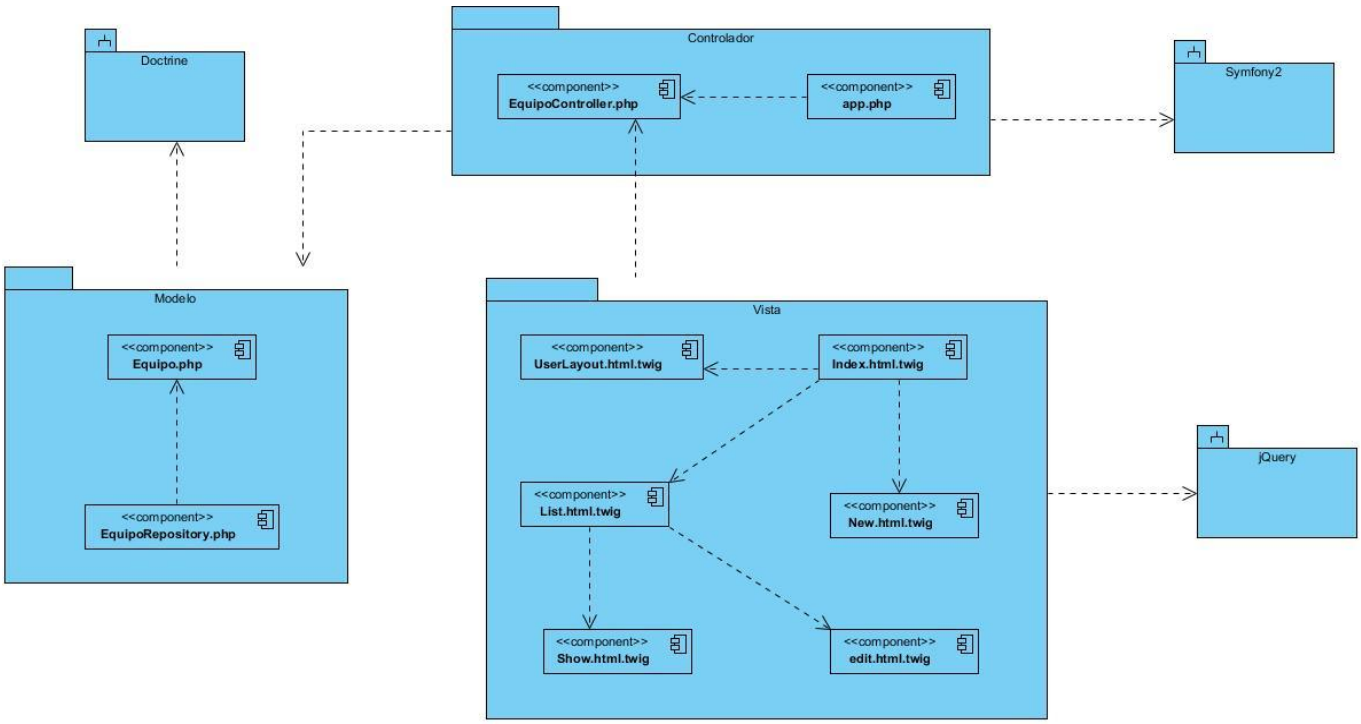

Figura 14: Diagrama de componentes del caso de uso Gestionar datos del equipo**.**

3.2. Diagrama del despliegue

El Modelo de Despliegue muestra la disposición física de los distintos nodos que componen un sistema. Este representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación [20].

Para el despliegue de la aplicación se dispone de un servidor web (nodo Servidor Web) encargado de responder las peticiones que emite la computadora cliente (nodo PC Cliente) por medio del protocolo seguro HTTPS. A través de este último los usuarios pueden interactuar y visualizar las opciones que brinda el sistema. El servidor web se conecta por el protocolo TCP/IP al servidor de base de datos (nodo Servidor BD) encargado de administrar la información de las áreas docentes almacenada en la base de datos.

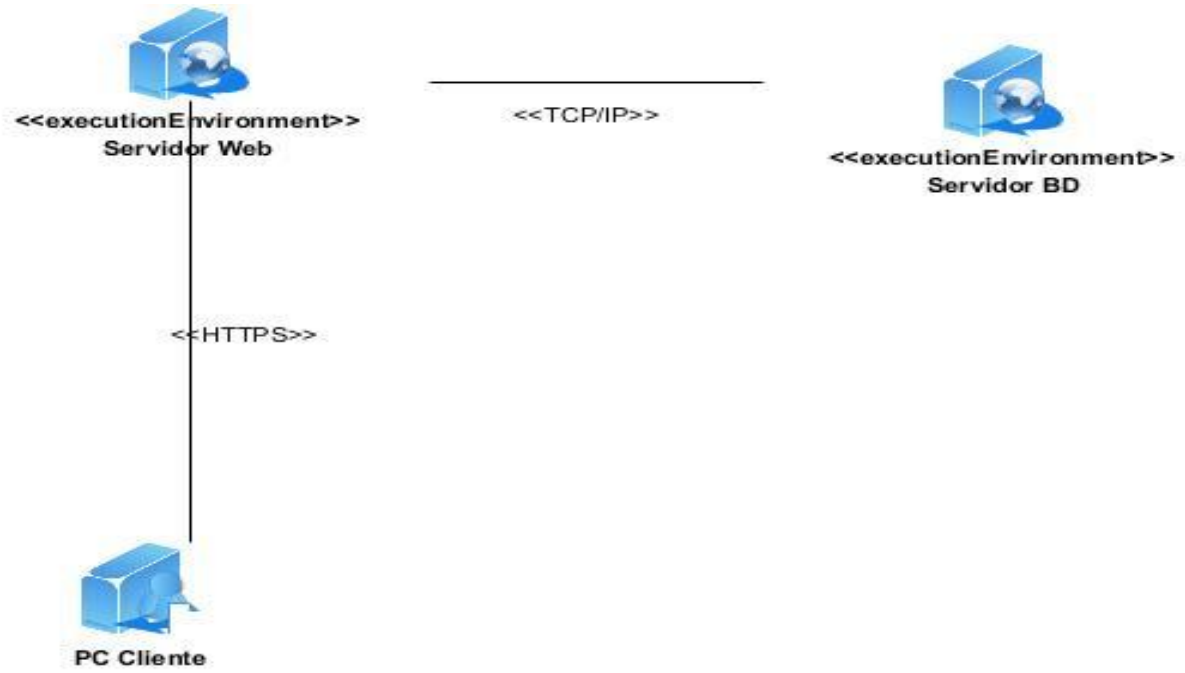

Figura 15: Diagrama de despliegue**.**

3.3. Código fuente

Para obtener una versión funcional de la aplicación se deben implementar los componentes que se han definido, como resultado se obtienen archivos que contienen el código fuente de la aplicación. El código fuente de un *softwar*e es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento. Estas instrucciones son escritas en un lenguaje de programación que consiste en un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones [21].

3.3.1. Estándar de codificación

Los estándares de codificación constituyen el elemento principal por el cual se rigen los desarrolladores para asegurar que el código sea de alta calidad y que no contenga gran cantidad de errores. Permitiendo así que el trabajo en equipo sea más efectivo y que los programadores puedan modificar el código gracias a la facilidad de programación y a la estandarización de la misma.

Symfony2 como *framework* propone unas series de normas para construcción del sistema del cual se utilizaron en mayoría las que se muestran a continuación:

Estructura:

- ✓ No utilizar las etiquetas cortas (<?).
- ✓ Agregar un único espacio antes de los paréntesis de apertura de una palabra clave de control

(*if*, *else*, *for*, *while*,...).

- ✓ Agregar una línea en blanco antes de la sentencia *return*.
- ✓ No agregar espacios al final de las líneas.
- ✓ Utilizar llaves para indicar el cuerpo de las estructuras de control sin importar el número de sentencias que estas contengan.

Convención de nombres

- ✓ Utilizar camelCase y no guiones bajos, para variables, funciones y nombres de métodos.
- ✓ Utilizar los *namespace* para todas las clases.
- ✓ Utilizar *Symfony* como el *namespace* de primer nivel.
- ✓ Utilizar los *namespace* para todas las clases.
- ✓ Añadir como sufijo *Interface* a las interfaces.
- ✓ Las anotaciones *@package* y *@subpackage* no son utilizadas.

3.3.2. Ejemplo de código fuente

3.3.2.1. Código fuente de la funcionalidad ConsumorealAction() referente al CU Mostar consumo real.

```
public function ConsumorealAction() {

    $Importar = $this->get('importar.xml.import');
    $Request = $this->getRequest();
    $Fechainicio = date("Y-m-d", strtotime($fecha_i));
    $Ruta = $Request->get('Ruta', null);
    $ArchivoTemporal = $_FILES['Ruta']['tmp_name'];
    $Tipo = $_FILES['Ruta']['type'];

    $Importar->validarreportes($ArchivoTemporal);
    $Importar->validarreporte($ArchivoTemporal);

    $Consumo = 0;
    $Suma = $importar->logit($ArchivoTemporal);
    for ($i = 0; $i < $Suma; $i++) {
        $Fecha = $Importar->fecha($i);
        if ($fecha_inicio <= $Fecha) {
            $Consumo = $Consumo + $Importar->Consumo($i);
        }
    }

    $Result = array();
    $Result["entities"] = $Consumo;

    return $Result;
}
```

Figura 16: Ejemplo de código fuente referente al caso de uso Mostrar consumo real**.**

3.4. Pantallas principales de la aplicación

El diseño de la interfaz de la aplicación se encamina a lograr interfaces agradables, de fácil navegación y activas para el cliente. Para ello se utiliza lenguaje CSS con el cual se lograron pantallas que permitirán al especialista visualizar los datos del consumo estimado y las lecturas del consumo real según un rango de fechas. A continuación, se muestran imágenes de las principales pantallas de la aplicación.

En la siguiente interfaz visual donde se muestra una comparación entre el consumo estimado que se obtiene de los reportes y el consumo real de energía eléctrica cargados de archivos XML.

Figura 17: Interfaz de la aplicación que muestra la sección de Graficar consumo estimado y real.

En la siguiente interfaz visual se muestran los reportes por fechas, tipos de local y consumo de dicho local.

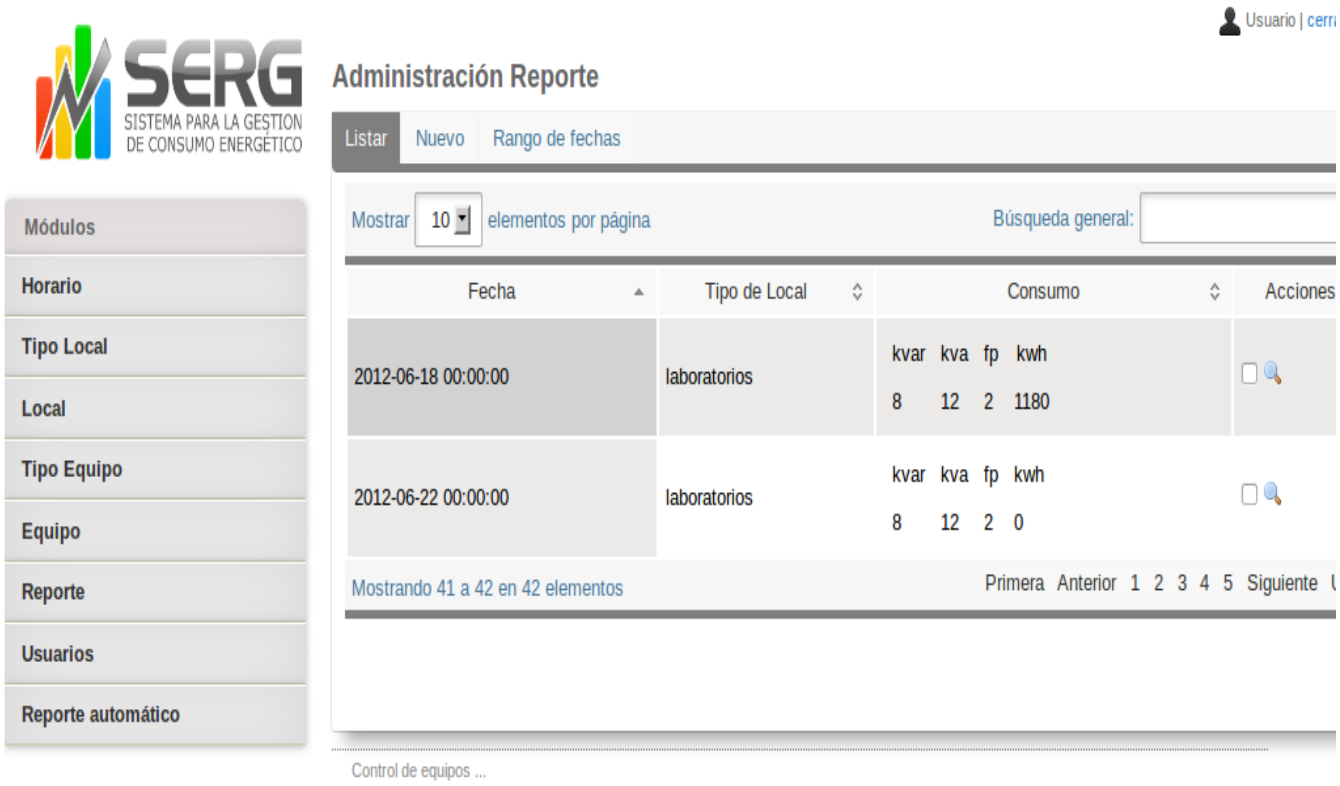

Figura 18: Interfaz de la aplicación que muestra la sección de administración de los reportes.

3.5. Conclusiones del capítulo.

En este capítulo se estructuró el modelo de implementación a partir de los resultados obtenidos en el diseño. Se realizó el diagrama de despliegue que muestra la configuración del funcionamiento del sistema incluyendo el *software* y *hardware*. Se describieron las particularidades de los subsistemas de implementación contenidos en los diagramas de componentes. Se implementó el *software* a partir de los artefactos generados en los flujos de trabajo anteriores y se caracterizaron los estilos de codificación utilizados.

CAPÍTULO 4: PRUEBAS DEL SISTEMA

Desde el inicio de la construcción del sistema pueden evidenciarse errores al definirse los objetivos, así como en las siguientes fases del diseño e implementación y posteriormente a esta. Es por ello que el proceso de desarrollo va acompañando de la actividad de realizar pruebas para garantizar su calidad. Durante el desarrollo del capítulo, se valida mediante pruebas si el sistema implementado ha cumplido con el objetivo inicial y se presenta la validación de la solución propuesta en el capítulo anterior, de manera que se pueda comprobar las funcionalidades utilizadas para dar respuesta a los distintos requisitos definidos.

4.1. Modelo de Prueba

La fase de pruebas añade valor al producto que se maneja, ya que todos los programas pueden tener errores, y en esta fase deben ser detectados, contribuyendo así a la calidad del *software*. La prueba de software es un conjunto de herramientas, técnicas y métodos que hacen la excelencia del desempeño de un programa. En esta etapa se realizan los casos de prueba, que intentan ocasionar errores en el *software*, realizando todo tipo de operaciones que pudieran ocasionar un mal funcionamiento del mismo, para detectar problemas que el sistema pudiese presentar. Las pruebas no pueden confirmar la ausencia de errores del *software*, pero sí demostrar que tiene defectos.

Al sistema desarrollado se le aplicaron las pruebas a nivel de desarrollador de tipo funcional las cuales fijan su atención a validar los casos de uso. Para esto se utilizaron los métodos de Caja Blanca y Caja Negra. Para la realización de la prueba de Caja Blanca se utilizó la técnica de camino básico y para las pruebas de Caja Negra se utilizó la técnica de partición de equivalencia. A continuación, se describe de forma breve las pruebas utilizadas y los resultados arrojados.

4.1.1. Pruebas de Caja Blanca

La prueba de Caja Blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de Caja Blanca, el ingeniero del *software* puede obtener casos de prueba que garanticen que:

- ✓ Se ejerciten por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- ✓ Se ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez [22].

Para la realización de estas pruebas de Caja Blanca se utilizará la técnica de camino básico o cobertura de caminos. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la

definición de un conjunto básico de caminos de ejecución. En este tipo de pruebas se elaboran grafos de flujo, se determina su complejidad ciclomática para obtener el conjunto básico de caminos linealmente independientes y de esta forma los casos de prueba del conjunto básico con el objetivo de ejecutar al menos una vez cada sentencia del programa [20].

Para validar el software desarrollado, fue sometido a pruebas de caja blanca por el grupo de desarrollo, haciendo uso de un caso de prueba asociado a cada requisito funcional implementado, los cuales permitieron que se ejecutara un examen minucioso de la lógica del código implementado.

4.1.1.1. Aplicación del método de Caja Blanca

A continuación, se muestra la técnica de camino básico aplicada a la funcionalidad createAction() del caso de uso Gestionar datos del equipo donde se detallan cada uno de los pasos que sigue la misma.

1. Función a probar.

```
public function createAction()
{
    $entity  = new Equipo();
1   $request = $this->getRequest();
    $form    = $this->createForm(new EquipoType(), $entity);
    $form->bindRequest($request);

  2 if ($form->isValid()) {
        $em = $this->getDoctrine()->getEntityManager();

        $valor2kva=pow($entity->getKva(), 2);
        $valor2kvar=pow($entity->getKvar(), 2);
        $valor3resta=$valor2kva-$valor2kvar;
        $raiz=sqrt($valor3resta);
      $entity->setKw($raiz);
3
        $valor4=$raiz/$entity->getKva();
        $arcocos=acos($valor4);
    $entity->setFp($arcocos);

        $em->persist($entity);
        $em->flush();

        return $this->redirect($this->generateUrl('admin_equipo'));
    }
        return array(
4           'entity' => $entity,
            'form'   => $form->createView()
        );
  5 }
```

Figura 19: Funcionalidad createAction() referente al caso de uso Gestionar datos del equipo.

2. Grafo de complejidad.

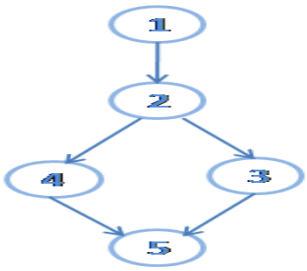

Figura 20: Grafo de complejidad referente al caso de uso Gestionar datos del equipo

3. Complejidad ciclomática

Existen varias formas de calcular la complejidad ciclomática del sistema a partir del grafo de complejidad antes expuesto:

- ✓ El número de regiones del grafo coincide con la complejidad ciclomática, V(G).
  V(G)= número de regiones= 2.
- ✓ La complejidad ciclomática, (VG), de un grafo de flujo está dada por la cantidad de aristas menos la cantidad de nodos más dos.
  V(G)= (5-5) + 2= 2.
- ✓ La complejidad ciclomática, V(G), de un grafo flujo G se define como la cantidad de nodos predicados más uno.
  V(G)= 1+ 1=2.

4. Caminos linealmente independientes.
   Camino 1: 1—2—3—5.
   Camino 2: 1—2—4—5.
5. Caso de prueba

Tabla 12: Caso de prueba referente al caso de uso Gestionar datos del equipo.

Número de camino	Caso de prueba	Objetivo	Resultado esperado
1	Que los datos entrados en los formularios sean válidos.	Permitir que se adicione un equipo	Mostrar los datos del equipo.
2	Que los datos entrados en los formularios sean inválidos.	Permitir que se adicione un equipo	Mostrar mensaje indicando el erro.

4.1.2. Pruebas de Caja Negra

Pruebas funcionales que aplican el método de Caja Negra se llevan a cabo sobre la interfaz de usuario y se centran principalmente en los requisitos funcionales del *software*. Estas pruebas mediante los casos de pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Examina aspectos del modelo, principalmente del sistema, sin tener en cuenta la estructura interna del *software*.

La prueba de Caja Negra intenta encontrar errores de las siguientes categorías:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas.
- ✓ Errores de rendimiento.

- ✓ Errores de inicialización y de terminación.

Para confeccionar los casos de prueba de Caja Negra existen distintos criterios. Algunos de ellos son:

Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Técnica del Análisis de Valores Límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones [20].

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software.

4.1.2.1. Técnica de partición de equivalencia.

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

El diseño de casos de prueba según esta técnica consta de dos pasos:

- ✓ Identificar las clases de equivalencia.
- ✓ Identificar los casos de prueba.

Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de clases de equivalencia, las clases de equivalencia válidas, que representan entradas válidas al programa, y las clases de equivalencia no válidas, que representan valores de entrada erróneos [20].

4.1.2.2. Diseño de casos de prueba.

Se entiende por caso de prueba, según la ingeniería del *software*, al conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

Los casos de pruebas diseñados pretenden demostrar que las funciones del sistema son operativas. Esto ocurre cuando la entrada se acepta de forma adecuada y se produce un resultado correcto, así como la integridad de la información externa se mantiene. A continuación, se muestra el diseño de caso de prueba del caso de uso Gestionar datos del equipo donde se recoge la descripción de las variables y los juegos de datos a probar.

Caso de Uso Gestionar datos del equipo.

Sección Adicionar datos del equipo.

Tabla 13: Caso de prueba para el caso de uso Gestionar datos del equipo, escenario Adicionar datos del equipo.

Nombre de la sección	Escenario	Descripción	1	2	3	4	5	6	7	8	Respuesta del sistema	Flujo central
SC 1: Adicionar datos del equipo.	EC 1.1 Adicionar datos del equipo. Correctamente	Se adiciona correctamente el equipo	V	V	V	V	V	V	V	V	Se adiciona correctamente el equipo en la base de datos. Se muestra en la interfaz ¨Listar¨ el los datos del equipo adicionado	1-Se selecciona la opción ¨Nuevo¨. 2- Se llenan los datos. 3-Se ejecuta la acción.
	EC 1.2 Adicionar datos del equipo. Con campos en blanco	Se adiciona el elemento dejando campos en blancos. Se muestra un mensaje indicando el error.	NA	NA	NA	NA	NA	NA	NA	NA	Se muestra un mensaje indicando que existen campos requeridos en blanco.	1- Se selecciona la opción ¨Nuevo¨. 2- Se llenan los datos. 3- Se ejecuta la acción. 4- Se muestra un mensaje de error.
	EC 1.3 Adicionar datos del equipo. No cumple con el formato adecuado.	Se adiciona el elemento con sin el formato adecuado. Se muestra un mensaje indicando el error.	NA	NA	NA	NA	NA	NA	NA	NA	Se muestra un mensaje indicando que el formato no es el correcto	1- Se selecciona la opción ¨ Nuevo¨. 2- Se llenan los datos. 3- Se ejecuta la acción. 4- Se muestra un mensaje de error.

Sección Modificar datos del equipo.

Tabla 14: Caso de prueba para el caso de uso Gestionar datos del equipo, escenario Modificar datos del equipo.

Nombre de la sección	Escenario	Descripción	1	2	3	4	5	6	7	8	Respuesta del sistema	Flujo central
SC2: Modificar datos del equipo.	EC 2.1 Modificar datos del equipo. Correctamente.	Se adiciona correctamente el equipo.	V	V	V	V	V	V	V	V	Se adiciona correctamente el equipo en la base de datos. Se muestra en la interfaz ¨ Listar¨ el los datos del equipo modificado.	1-Se selecciona la opción ¨ Editar¨. 2- Se llenan los datos. 3- Se ejecuta la acción.
	EC 2.2 Modificar datos del equipo. Con campos en blanco.	Se modifica el equipo dejando campos en blancos. Se muestra un mensaje indicando el error.	NA	NA	NA	NA	NA	NA	NA	NA	Se muestra un mensaje indicando que existen campos requeridos en blanco.	1-Se selecciona la opción ¨Editar¨. 2- Se llenan los datos. 3- Se ejecuta la acción. 4- Se muestra un mensaje de error.
	Modificar datos del equipo. No cumple con el formato adecuado	Se modifica el elemento con sin el formato adecuado. Se muestra un mensaje indicando el error.	NA	NA	NA	NA	NA	NA	NA	NA	Se muestra un mensaje indicando que el formato no es el correcto	1- Se selecciona la opción ¨ Nuevo¨. 2- Se llenan los datos. 3- Se ejecuta la acción. 4- Se muestra un mensaje de error.

Sección Eliminar datos del equipo.

Tabla 15: Caso de prueba para el caso de uso Gestionar datos del equipo, escenario Eliminar datos del equipo.

Nombre de la sección	Escenario	Descripción	1	2	3	4	5	6	7	8	Respuesta del sistema	Flujo central
SC3: Eliminar datos del equipo.	EC 3.1 Eliminar datos del equipo. Correctamente	Se elimina correctamente el equipo	V	V	V	V	V	V	V	V	Se elimina correctamente el equipo en la base de datos.	1-Se selecciona un equipo del listado. 2-Se muestran los datos del equipo. 3- Se escoge la opción eliminar.
	EC 3.2 Eliminar datos del equipo. Incorrecto.	Se elimina el equipo con problemas.	NA	NA	NA	NA	NA	NA	NA	NA	El sistema no elimina el equipo de forma correcta.	1-Se selecciona un equipo del listado. 2- Se escoge la opción eliminar.

Tabla 16: Descripción de las variables de caso de uso Gestionar datos del equipo**.**

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	campo de texto	No	Alfanumérico de 1 a 255 caracteres. Admite a-z A-Z - áéíóú ñÑ ÓÁÍÉÚ üÜ
2	Código	campo de texto	No	Alfanumérico de 1 a 255 caracteres. Admite 0-9 a-z A-Z - áéíóú ñÑ ÓÁÍÉÚ üÜ
3	Descripción	campo de texto	No	Alfanumérico de 1 a 255 caracteres. Admite 0-9 a-z A-Z - áéíóú ñÑ ÓÁÍÉÚ üÜ
4	Kva	campo de texto	No	Alfanumérico de 1 a 255 caracteres. Admite 0-9
5	Kvar	campo de texto	No	Alfanumérico de 1 a 255 caracteres. Admite 0-9
6	Activo	campo de selección	No	Alfanumérico de 0 a 1 caracteres.
7	Local	lista desplegable	No	Alfanumérico de 1 a 255 caracteres. Admite 0-9 a-z A-Z - áéíóú ñÑ ÓÁÍÉÚ üÜ
8	Tipo	lista desplegable	No	Alfanumérico de 1 a 255 caracteres. Admite 0-9 a-z A-Z - áéíóú ñÑ ÓÁÍÉÚ üÜ

Tabla 17: No conformidades detectadas**.**

No	Elemento	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativo
1	Interfaz	No se valida el formato que deben tener los datos al insertarse.	En la sección ¨ Nuevo¨, en el formulario ¨Código¨ no se muestra el mensaje de error cuando se entra el código sin el formato adecuado.	Etapa de aplicación de pruebas al sistema.	X
2	Interfaz	No se muestra el mensaje de error cuando existe algún campo en blanco.	En la sección ¨ Editar ¨, en el formulario ¨Nombre¨ no se muestra el mensaje de error cuando se entra el nombre sin el formato adecuado.	Etapa de aplicación de pruebas a la aplicación.	X
3	Interfaz	Faltas de ortografía en los campos formularios, cuando se modifica un equipo.	En la sección ¨ Editar equipo¨, los nombres de los formularios.	Etapa de aplicación de pruebas a la aplicación.	X

4.2. Conclusiones del capitulo

En el presente capítulo se describió la etapa de prueba del *software*. Se desarrollaron las pruebas de desarrollador de tipo funcionales; para ello se realizaron los casos de pruebas mediante los métodos de caja blanca y caja negra. Se definieron las técnicas de camino básico y partición de equivalencia que permitieron guiar el diseño de los casos de prueba de caja blanca y negra para determinar que se ha cumplido con los requisitos funcionales. De modo que se obtuvieron tres no conformidades las cuales fueron registradas en el artefacto Casos de Prueba y posteriormente resueltas, mejorando así el funcionamiento del software.

CONCLUSIONES

Una vez culminado el trabajo se arriba a las siguientes conclusiones:

- ✓ Para la elaboración de este trabajo de diploma, se realizaron estudios sobre los problemas que presenta el departamento de Atención a Programas Energéticos referentes a la gestión de la información para la planificación y control del consumo de la energía eléctrica en las diferentes áreas docentes.
- ✓ Se analizaron los distintos sistemas existentes a nivel internacional y nacional para la gestión de la información, estos estudios arrojaron que los sistemas no cumplen con las funcionalidades que necesita la dirección del departamento por lo que se decide comenzar desde cero.
- ✓ Se definieron las metodologías, herramientas y lenguajes de programación permitiendo que se definieran cuáles utilizar para el desarrollo de la solución.
- ✓ Se analizaron y diseñaron las funcionalidades como resultado del proceso de captura de requisitos.
- ✓ Se diseñaron los diagramas de componentes por cada caso de uso, como indica la metodología utilizada, dando paso a la implementación del sistema.
- ✓ Se probó el correcto funcionamiento de la herramienta a través de las pruebas de caja blanca y caja negra. Los resultados obtenidos, probaron que la solución cumplen con el objetivo general y las tareas propuestas.

RECOMENDACIONES

Los autores recomiendan:

- ✓ Integrar al sistema la información referente a las áreas no docentes consumidoras del portador energético electricidad en la universidad.
- ✓ Aplicar el presente trabajo en instituciones fuera de la universidad, que requieran del uso de una aplicación similar para la planificación del consumo de la energía eléctrica .

REFERENCIAS BIBLIOGRÁFICAS

1**. Sistemas-de-Gestion** [En línea] [Consultado el: 21 de 02 de 2012.] http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

2. **Mejoratugestion**. [En línea] [Consultado el: 13 de 05 de 2012.] http://mejoratugestion.com/mejora-tu-gestion/que-es-un-sistema-de-gestion/ .

3. **Scribd**. *scribd* [En línea] [Consultado el: 2 de 01 de 2012.] Disponible en: http://es.scribd.com/doc/29868446/Los-sistemas-de-gestion-de-Informacion-piedra-angular-de-la-estrategia-integral-de-gerencia .

4. **Scielo**.*ContaLuzWE*. [En línea] [Consultado el: 3 de 02 de 2012.] Disponible en: http://www.scielo.cl/scielo.php?pid=s0718-07642006000400013&script=sci_arttext.

5. **CECRE**.*CECRE*. [En línea] [Consultado: 02 14, 2012.] Disponible en: http://www.ree.es/sala_prensa/web/infografias_detalle.aspx?id_infografia=45.

6. **Juventudrebelde**. *juventudrebelde*. [En línea] [Consultado el: 02 14, 2012.] Disponible en: http://www.juventudrebelde.cu/cuba/2009-03-08/cuba-lo-mas-novedoso-en-informatizacion/ .

6. **Juventudrebelde**. *juventudrebelde*. [En línea] [Consultado el: 02 14, 2012.] Disponible en: http://www.juventudrebelde.cu/cuba/2009-03-08/cuba-lo-mas-novedoso-en-informatizacion/ .

7**. Eclipse.** *Eclipse-openUp*. [En línea] [Consultado el: 11 13, 2011.] Disponible en: http://www.eclipse.org/epf/.

8. **Extreme-Programming.** *Extreme-Programming***.** [En línea] [Consultado el: 02 18, 2012.] Disponible en: http://es.scribd.com/doc/72837716/5/eXtreme-Programming-XP.

9. **Fabien Potencier, François Zaninotto** *Symfony 1.1, con pocas palabras. Framework.*

10. **Chaffer, Jonathan; Swedberg, Karl.** *Learning jQuery 1.3*. 2009.

11. **PHP Manual**. *The PHP Group. PHP Manual.* [En línea] [Consultado el: 03 01, 2012.] Disponible en: http://www.php.net/manual/en/preface.php..

12. **Extjs**. *Extjs.* [En línea] [Consultado el: 03 de 01 de 2012] Disponible en: http://extjs.com/blog/2009/03/16/ext-js-books.

13. **World Wide Web Consortium.** *HTML*. [En línea] [Consultado el: 20 de 03 de 2012.] Disponible en: http://www.w3.org/TR/xhtml1/#xhtml.

14. **pgAdmin PostgreSQL Tools.** *pgAdmin PostgreSQL Tools.* pgAdmin [En línea] [Consultado el: 03 de 13 de 2012.] Disponible en: http://www.pgadmin.org/..

15**. ORG, NETBEANS**. *Welcome to NetBeans.* [En línea] 2008. [Consultado el: 18 de 11 de 2011] Disponible en: http://www.netbeans.org.

16. **Mateus, Carlos. 2004.** *Desarrollo de Aplicaciones Web.* Barcelona, España

17.**Scribd.** *UML*. [En línea] [Consultado el: 25 de 03 de 2012] Disponible en: http://es.scribd.com/doc/51515328/El-Proceso-Unificado-de-Desarrollo-de-Software-Jacobson-Booch-%E2%80%93-Rumbaugh-capitulo-6.

18. **Freedownloadmanager.** *Paradigma visual para UML (Plataforma Java)*. [En línea] [Consultado el: 15 de 01 de 2012] Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

19. **JACOBSON, G. B. J. R. I .** *El Proceso Unificado de Desarrollo de Software.* 2002.

20. **PRESSMAN, R. S.** *Ingeniería de Software: Un enfoque práctico.* 2005.

21**. Lasso, Ívan**. *Qué es el código fuente*. [En línea] 2008. [Consultado el: 03 de 12 de 2011.] Disponible en: http://www.proye ctoautodidacta.com/comics/que-es-el-codigo-fuente/.

BIBLIOGRAFÍA

1. **Langefors**, **Börje**. *Theoretical Analysis of Information Systems.* s.l. : Auerbach: s.n., 1973.

2.**Vega Briceño, Edgar Armando**. www.gestiopolis.com. *www.gestiopolis.com.* [En línea] 2005.[Consultado el: 8 de 02 de 2012] Disponible en: http://www.gestiopolis.com/Canales4/mkt/simparalas.htm.

3. **I.O., Angell y S., Smithson**. *Information Systems Management: Opportunities and Risks.* 1991.

4.**Power Studio. circutor.** *circutor.* [En línea] 2008. [Consultado el: 10 de Noviembre de 2011] Disponible en: http://www.circutor.es/novedades/powerstudio/powerstudio-sp.html.

5.**MIC, atenas.** *atenas.* [En línea] 2008. [Consultado el: 11 de 11 de 2011] Disponible en: http://www.atenas.cult.cu/?q=node/6415.

6.**Blogspot**. *Bloggerblogpost.* [En línea] [Consultado el: 11 de 11 de 2011] Disponible en: http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html

8. **CANÓS, J. H.; LETELIER, P. *et al.*** *Metodologías Ágiles en el Desarrollo de Software.* Valencia : DSIC -Universidad Politécnica de Valencia, 2005.

9. **Beck, K. *Extreme Programming Explained***. *Embrace Change.* s.l. : Pearson Education, 1999.

10. **Potencier, Fabien; Zaninotto, François.** *Symfony La guía Definitiva.* France : Sensio S.A, 2008.

11.**Oracle**.*Oracle* [En línea] 2010. [Consultado el: 18 de Noviembre de 2011] Disponible en: http://www.oracle.com/subscribe/index.html&prev=/search%3Fq%3DOracle%26hl%3Des&rurl=translate.google.com.cu&usg=ALkJrhjxkLOVlt5MBNinQjSWvVV4qKfQqQ.

12. **Vargas Rio**, **Anelis; Machado Estévanez, Mayte** . *Análisis, Diseño e Implementación de un Mercado de Datos para el Departamento de Población de la Oficina Nacional de Estadísticas.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. Universidad de las Ciencias Informáticas . La Habana : s.n, 2010.

14. **RUMBAUGH**, **J**.; **JACOBSON**., I. *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l : A. Wesley, 200. ISBN 84-7829-037-08.

15. **WELLING**, **L.** y **THOMSON**, L. *Desarrollo Web con PHP y MySQL.* Madrid : Grupo Anaya, S.A , 2006. ISBN 84-415-1569-7.

16. **GALLEGO VÁZQUEZ**, J. A. *Desarrollo Web con PHP y MySQL.* Madrid : Grupo Anaya S.A. : E. Tuya Feijó, 2003. ISBN 84-415-1525-5.

17. **Oracle**. **java**. *java.* [En línea] 2008. [Consultado el: 12 de Noviembre de 2011] Disponible en: http://www.java.com/es/download/faq/whatis_java.xml..

18. **ORG, NETBEANS**. *Welcome to NetBeans*. [En línea] 2008. [Consultado el: 18 de Noviembre de 2011] Disponible en: http://www.netbeans.org.

19. **Group**, **Development**. **PostgreSQL**. *PostgreSQL Global.* [En línea] 1996-2010. [Citado el: 22 de Noviembre de 2011.] Disponible en: http://www.postgresql.org/&ei=zuiHS72hNcP08Qbb8JGYDw&sa=X&oi=translate&ct=result&resnum=1&ved=0CA0Q7gEwAA&prev=/search%3Fq%3Dpostgre.

20. **Ecured.** *ecured.* [En línea] 2012. [Consultado el: 25 de Noviembre de 2011.] Disponible en: http://www.ecured.cu/index.php/Servidores_Web.

21. **CodeIgniter_Spanish_UserGuide**. *CodeIgniter_Spanish_UserGuide.* [En línea] [Consultado el: 12 de Noviembre de 2011.] Disponible en: http://lax.franhp.net/CodeIgniter_Spanish_UserGuide.pdf.

22. **Ecured.** *ecured***.** [En línea] [Consultado el: 22 de Enero de 2011] Disponible en: http://www.ecured.cu/index.php/Sencha_Ext_JS.

23. **Ecured**. *ecured*. [En línea] [Consultado el: 12 de Febrero de 2012.] Disponible en: http://www.ecured.cu/index.php/Sistemas_de_control_de_versiones#GIT.

24. **Tecnoretales**. *Tecnoretales. Introducción al ORM* . [En línea] [Consultado el: 23 de Noviembre de 2011.] Disponible en: http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.

25. **Escobar Domínguez, R. R.** *Desarrollo de un Almacén de Datos para el Control del Consumo de Portadores Energéticos en la Oficina Nacional de Estadística.* Trabajo de Diploma para Optar por el título de Ingeniero en Ciencias Informáticas. La Habana : s.n, 2010.

26. **Potencier, Fabien** . *El tutorial Jobeet.* 2009.

27. **Larman,Craig.** *UML y Patrones.* Mexico : Editorial Mexicana, 1999. 970-17-0261-1.

28. **Jacobson, Ivar; Booch, Grady; Rumbaugh, James**. *El Proceso Unificado de Desarrollo de Software.* España : Adison-Wesley.

29. **Corp, IBM**. *Rational Software Architect.* 2006.

30. **Pérez Vázquez, Maridalia; Rodríguez Hidalgo, Guillermo William**. *Desarrollo de componentes de Interfaz de usuarios para la representación de información mediante gráficas en el polo Petrosoft.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana : s.n., 2010.

31. **Pressman, Roger S**. *Ingeniería del Software, un enfoque práctico.* 1997.

32. **Buschmann, Frank** et al.: *Pattern Oriented Software Architecture,* Volume1: A System of Patterns, Willey & Sons, 1996.

ANEXOS

Anexo 1: Diagramas de Componentes.

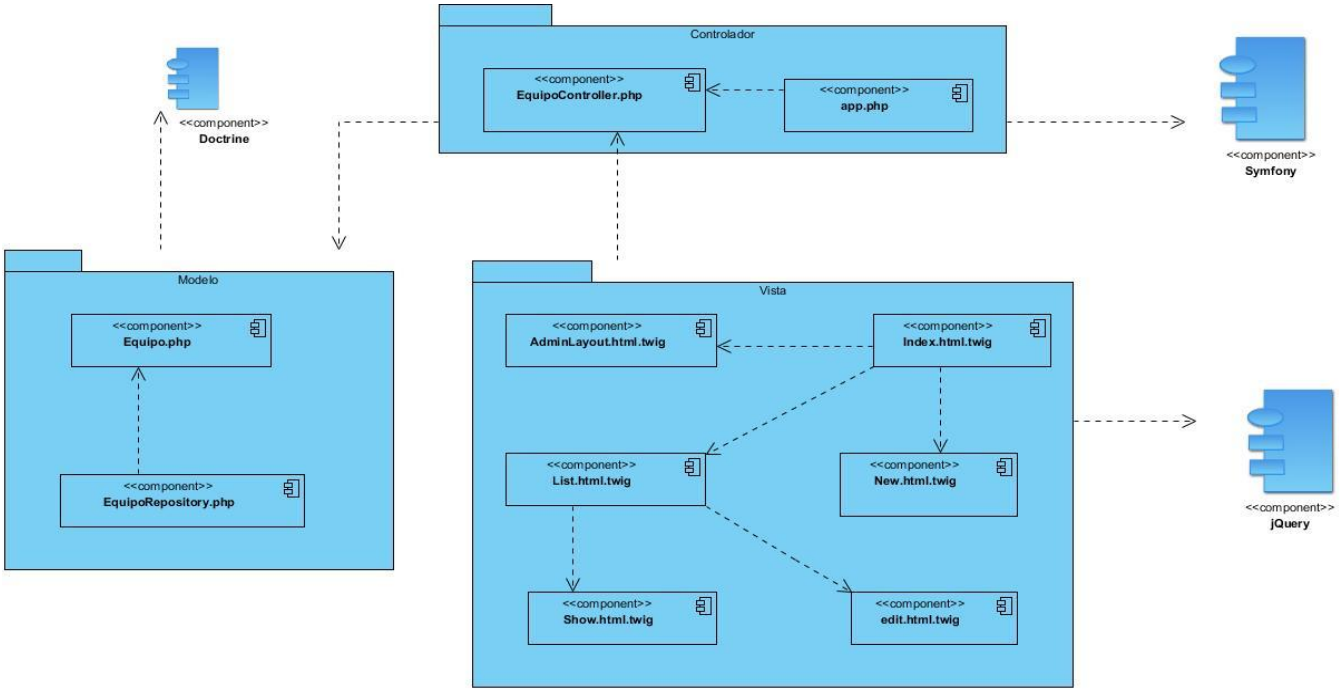

Figura 21: Diagrama de Componentes del CU Gestionar datos del equipo.

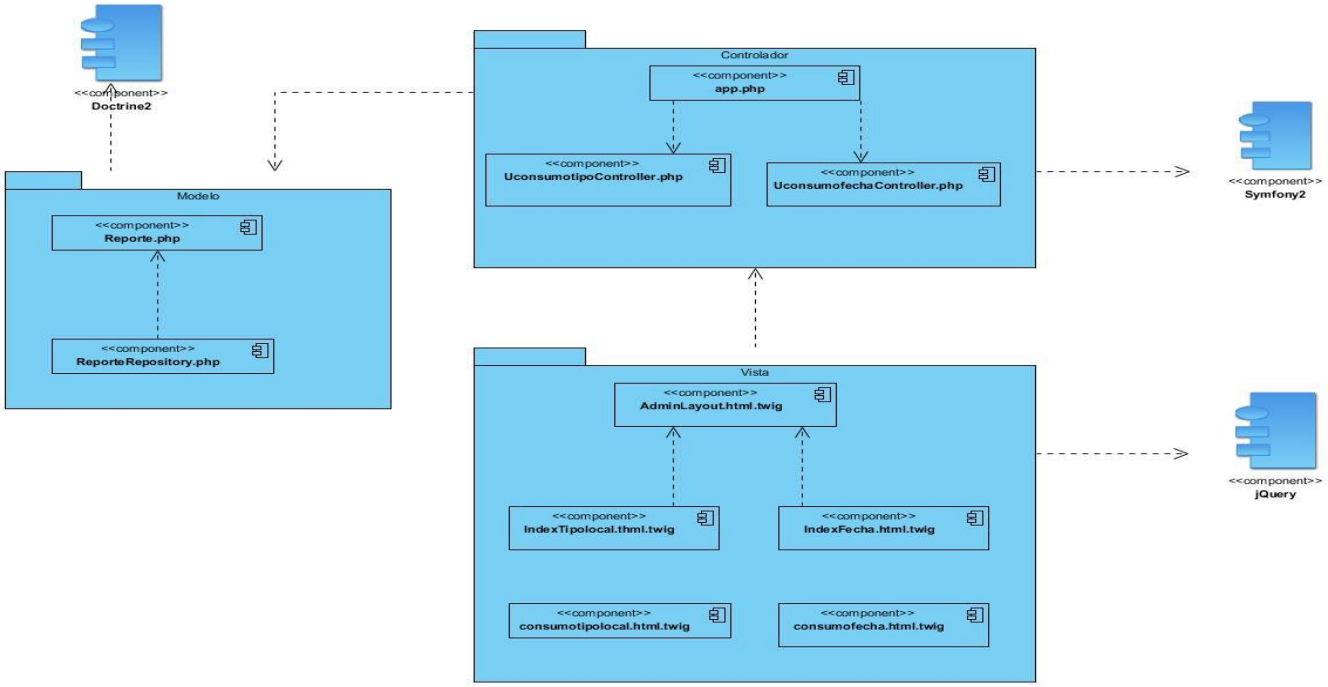

Figura 22: Diagrama de Componentes del CU Mostrar consumo estimado.

GLOSARIO DE TÉRMINOS

A

Ada: lenguaje de programación orientado a objetos que se usa principalmente en entornos en los que se necesita una gran seguridad y fiabilidad.

AJAX: Acrónimo de Asynchronous JavaScript And XML (en español: JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (del inglés: Rich Internet Applications).

C

Código abierto: en inglés; *open source*, es el término con el que se conoce al *software* distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código.

Clase: es la unidad básica que encapsula toda la información de un objeto

Caso de Prueba: conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

E

Estilos Arquitectónicos: indican los tipos de componentes y conectores involucrados. Patrones y restricciones de interconexión o composición entre ellos. Asociados a cada estilo hay una serie de propiedades que lo caracterizan.

F

Framework: estructura conceptual y tecnológica de soporte definida, normalmente, como artefactos o módulos de *software* concretos, con base en la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

G

GNU/Linux: Sistema operativo libre de tipo UNIX,

H

HTML: acrónimo de HyperText Markup Language (en español: Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

HTTP: acrónimo de HyperText Transfer Protocol (en español: Protocolo de Transferecia de Hipertexto), es el protocolo usado en cada transacción de la World Wide Web. Define la sintaxis y semántica que utilizan los elementos de *software* de la arquitectura web (clientes, servidores, proxis) para comunicarse entre sí.

HTTPS: acrónimo de Hypertext Transfer Protocol Secure (en español: Protocolo Seguro de Transferencia de Hipertexto), es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

J

jQuery: es una biblioteca o *framework* de Javascript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

L

Licencia BSD: es una licencia de *software* otorgada principalmente a los sistemas Berkeley Software Distribution (BSD). Es una licencia de software libre permisiva y tiene menos restricciones en comparación con otras, estando muy cercana al dominio público. La licencia BSD permite el uso del código fuente en *software* no libre.

M

Multiplataforma: es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de *software*, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en los sistemas operativos Windows, en GNU/GNU/Linux y en MacOS.

O

Orientado a objetos: hace referencia a la Programación Orientada a Objetos, un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de ordenador. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento.

ORM: acrónimo de Object-Relational Mapping (en español: Mapeo Objeto-Relacional), es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

P

Patrones: normas de comportamiento, características que identifican una situación. En informática un patrón es una solución a un problema de diseño no trivial que es efectiva y reutilizable. Es posible aplicar a diferentes problemas de diseño en distintas circunstancias.

Patrones de diseño: son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño.

PHP: lenguaje de programación orientado a objetos.

S

Script: conjunto de instrucciones escritas en un lenguaje script ejecutadas por un intérprete de comandos.

SCADA: un sistema SCADA (acrónimo de Supervisión, Control y Adquisición de Datos) es una aplicación de software, diseñada con la finalidad de controlar y supervisar procesos a distancia.

U

UCI: Universidad de las Ciencias Informáticas.

UML: acrónimo de Unified Modeling Language (en español: Lenguaje Unificado de Modelado) es un lenguaje de modelado para especificar o describir métodos o procesos.

URL: acrónimo de Uniform Resource Locator. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

V

Vista: una vista de base de datos es un resultado de una consulta SQL de una o varias tablas; también se le puede considerar una tabla virtual. Las vistas tienen la misma estructura que una tabla: filas y columnas. La única diferencia es que sólo se almacena de ellas la definición, no los datos. Los datos que se recuperan mediante una consulta a una vista se presentarán igual que los de una tabla.

W

Windows: hace referencia a Microsoft Windows; una serie de sistemas operativos desarrollados por la corporación Microsoft desde 1981.

X

XML: acrónimo de Extensible Markup Language (en español: lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.