Universidad de las Ciencias Informáticas.
Facultad 4

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

Título: Sistema de Registro y Control de Portadores Energéticos

Autor: Julio Cesar Bravo Rodríguez

Tutores: Ing. Alain Eduardo Rodriguez Arias
Dr. CT. Ramón Rodriguez Cabrera

Ciudad de la Habana, Junio del 2009.
"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Ing. Alain Rodríguez Arias **Dr. CT. Ramón Rodríguez Cabrera**

_____

Julio Cesar Bravo Rodríguez

"Hemos descubierto afortunadamente algo mucho más importante, el ahorro de energía, que es como encontrar un gran yacimiento."

Agradecimientos

A mis padres Julio Antonio e Ida María por todos estos años de amor, paciencia y comprensión, por haberme enseñado tantas cosas de la vida y apoyarme siempre en lo que he querido.

A mis abuelos, Melba y Toto que me tienen como lo más grande que pueda haber en la vida y que siempre me han dado hasta lo que no han tenido.

A mis abuelos Rafael y Luisa que me dieron su cariño incondicional en vida y que aun sin estar físicamente, lo sigo sintiendo… a todos los llevo en mi corazón.

A mi hermanita linda por quererme tanto y siempre preocuparse por mí.

A Leydis Ferrera Rodríguez por apoyarme tanto y convertirse en mi mejor amiga.

A mis tutores Alain y Ramón por la paciencia y por estar siempre dispuestos a ayudarme.

Agradecimientos

A mis Tías Estrella, Margatira y Xiomara por ser tan buenas conmigo, por todo el cariño que me dan y el apoyo que me brindan, en especial a mi tía Estrella por todo ese amor que siente por mi hermana y por mí y acogernos como unos hijos.

A todos mis Tíos, Carlos, Panchy, Modesto, por ser especiales y aportarme el granito de arena para llegar hasta aquí.

A todos mis primos y primas que los quiero mucho y que siempre he sido para ellos el primito menor y en especial para mi primo Elvis que me quiere como su hijo varón y a mi primo Yaidel que es como mi hermano.

A todos mis amigos, los de mi casa y los de la escuela, les agradezco los momentos juntos que me permitieron aprender

A la Revolución por darme la oportunidad de tener un título y ser un profesional solo con mi esfuerzo y dedicación.

Dedicatoria

A mi mama y mi papa por el apoyo que me han dado durante todo estos años de estudio, por todo el sacrificio que han tenido que hacer para que yo hoy estuviera en este lugar, por el amor que me han inculcado y la educación que me he recibido de ustedes, quiero que sepan que si volviera a nacer quisiera a Tony y a Ida Maria como mis padres…. Por esto y mucho más los quiero con el alma.

A mi hermanita del alma que le sirva de guía en esta etapa de su vida, y quiero que sepas que si no estuvieras a mi lado y me sintiera con obligación de hacer por ti hasta lo último de mi vida no sé si hoy estuviera discutiendo esta tesis, por todo el dinero que gastaste llamándome cada fin de semana por todo esto te dedico el fruto de 5 años de sacrificio.

A mis abuelos Melba y Toto por que siempre han estado al tanto de cómo he estado, de todos mi momentos difíciles y

Dedicatoria

felices de mi vida, por quererme tanto y defenderme siempre…por todas las razones del mundo les dedico todo mi esfuerzo durante mi carrera universitaria.

A mis primas Tahili, Rachel, Taimi, Anachel, Carlos Ernesto, Yulie, Thalia, Edevito, Lorena y Elvito(el chiquitico) que les sirva de guía para que estudien, en especial a mis niñas chiquita Tahili y Rachel a todos les dedico este gran esfuerzo de 5 años lejos de la casa.

A toda mi familia que siempre me apoyaron en todo, a ellos que llegue este pequeño estimulo de todo mi esfuerzo.

Resumen

La Aduana General de la República de Cuba tiene especial interés en incorporar al SUA (Sistema Único de Aduanas) un módulo que sea capaz de controlar los procesos de demanda, distribución y evaluación de sus Portadores Energéticos; desde hace años se lleva el control del uso de estos portadores con mecanismos de información y cálculos manuales, por lo que en ocasiones el factor humano introduce errores u omisiones. En este trabajo se elabora una solución para el Sistema que se necesita, de manera que se optimicen estos procesos y se logre una mayor eficiencia.

Después de haber realizado la captura de requerimientos, que para esto se utilizó como metodología de desarrollo RUP y se modeló el negocio con BPMN, UML como lenguaje de modelado y la herramienta Visual Paradigm, donde los resultados fueron avalados por los clientes con total satisfacción a través de las descripciones textuales de los casos de usos asociados a estos requisitos y sus prototipo de interfaz de usuario que validan los mismos. Nos damos a la tarea de realizar la implementación por parte del desarrollador, de un sistema que sea capaz de dar respuesta a la necesidad de la Aduana General de la República.

Índice

Introducción

El desarrollo de un país en la actualidad está ligado a un creciente consumo de energía. En el caso de Cuba, por no tener abundantes yacimientos de combustible fósil y ante la realidad inminente de su agotamiento mundial, se hace necesaria la lucha contra el incremento del consumo de portadores energéticos derivados del petróleo, siendo esta una cuestión de importancia vital. Dentro del país, La Aduana General de la República de Cuba, para perfeccionar los mecanismos de demanda, distribución y control, tiene la necesidad de automatizar este proceso, con el objetivo de la utilización racional y eficiente de los Portadores Energéticos.

La Aduana General de la República es la encargada de regular el control aduanero aplicable a la entrada, el tránsito, el cabotaje, el trasbordo, el depósito y la salida del territorio nacional de mercancías, viajeros y sus equipajes, bienes y valores sujetos a regulaciones especiales, incluidas la flora y la fauna protegidas, así como los medios en que se transporten, además forma parte de la Administración Central del Estado y se subordina al Consejo de Ministros. Se designa a la Aduana como el órgano encargado de dirigir en materia aduanera, recaudar los derechos de aduanas y dar respuesta dentro de su jurisdicción y competencia a los hechos que incidan en el tráfico internacional de mercancías, viajeros, postal y los medios que los transportan, previniendo, detectando y enfrentando el fraude y el contrabando, así como contribuyendo a la protección nacional e internacional del medio ambiente.

Existen 13 unidades de aduanas aisladas entre sí en las diferentes regiones del país, cada una debe reportar mensualmente mediante diferentes modelos como se comportaron estos portadores energéticos y a su vez solicitar para el mes entrante un número estimado, lo más real posible de lo que necesitarán de estos portadores energéticos, este trabajo de registro y evaluación mediante modelos de forma manual lleva a que se introduzcan errores, que los números muchas veces no coincidan, que se realicen solicitudes innecesarias de algún portador energético determinado, que se dupliquen informaciones, que no coincida la demanda con la necesidad.

Luego de hacer un análisis detallado de las dificultades existentes, surge el siguiente **problema de la investigación**: La Aduana General de la República de Cuba realiza un proceso de demanda, registro de consumo y evaluación de indicadores de los

portadores energéticos manual y para lograr un uso más racional y eficiente se automatizará este proceso.

Mediante este Trabajo de Diploma, se quiere lograr que exista un proceso automatizado de registro y control de Portadores Energéticos, Haciendo un análisis detallado del tema.

En correspondencia con esto, el **objeto de estudio** estará enmarcado hacia los procesos de gestión de la información referente a los Portadores Energéticos en La Aduana General de la República de Cuba.

Los procesos de registro y evaluación de los Portadores Energéticos en los órganos rectores de La Aduana General de la República de Cuba, es la parte del objeto de estudio que se va a automatizar, es decir, **el campo de acción**.

El **objetivo general** de este trabajo es diseñar e implementar una aplicación que permita tener el control del uso de los portadores energéticos en la Aduana General de la República.

De acuerdo con esta propuesta se derivan algunos **objetivos específicos** como son:

1. Realizar el diseño de la aplicación a partir de los requerimientos identificados.

2. Desarrollar la aplicación usando las herramientas definidas en el polo.

Para poder darle cumplimiento a los objetivos planteados se proponen las siguientes

Tareas de la Investigación:

1. Estudiar las herramientas.
2. Estudiar las técnicas y patrones de diseño de aplicaciones.
3. Estudiar el lenguaje de programación.
4. Realizar el diseño de la aplicación.
5. Desarrollar la aplicación.

El presente trabajo se encuentra estructurado de la siguiente forma:

Capítulo 1. Fundamentación Teórica: En este capítulo se realiza un Análisis del estado del arte a nivel internacional y nacional acerca de los Portadores Energéticos. Además, se tratan aspectos fundamentales para la comprensión del sistema que se desea modelar, los conceptos más importantes, las tendencias actuales, las técnicas y tecnologías, entre otros aspectos de vital importancia para el desarrollo de esta investigación.

Capítulo 2. Diseño e Implementación del sistema: Se muestran los diagramas de clases del diseño, los diagramas de interacción específicamente los de secuencia, así como el diagrama de entidad relación de la base de datos incluyendo la descripción de las tablas, además de los diagramas de componente y despliegue de la aplicación.

Capitulo 1: Fundamentación Teórica

1.1. Introducción

El desarrollo de un país está ligado a un creciente consumo de energía, pero si se habla de desarrollo sostenible este aumento del consumo de energía debe tener un ritmo menor que el crecimiento del Producto Interno Bruto (PIB) o al crecimiento de cualquier otro indicador económico. En otras palabras, la intensidad energética debe ir disminuyendo con el desarrollo.

Entre los graves problemas que enfrenta la humanidad en los albores del tercer milenio, el suministro seguro, estable y controlado de portadores energéticos constituye una de las dificultades cruciales y además, resulta ineludible resolverlo adecuadamente para garantizar el desarrollo socioeconómico.

A lo largo de la historia, el hombre aprendió a centrar su atención en los aspectos del uso final de la energía y desarrolló el concepto de servicio energético, que es la vía mediante la cual la energía, un catalizador del desarrollo socioeconómico, se incorpora al valor agregado para sustituir el trabajo del hombre y los animales en la realización de tareas engorrosas, riesgosas o que demandan tiempo o esfuerzos excesivos.

Sin embargo, el suministro de portadores energéticos continúa atrayendo las investigaciones y el trabajo de los hombres a fin de obtener la energía que necesitan para respaldar su bienestar y prosperidad, al menos después de satisfacer sus necesidades más perentorias (alimentación y vivienda).

1.2. ¿Qué son los Portadores Energéticos?

Los Portadores Energéticos son aquellos elementos que son capaces de producir energía.

Algunos de ellos son, en nuestro caso:

-Combustibles (Diesel, Gasolina (Regular y Especial))

-Electricidad

-Lubricantes y solventes

-Gas licuado

-Gas manufacturado.

1.3. Otros Conceptos Fundamentales

Energía: Capacidad de realizar un trabajo

Eficiencia Energética: Habilidad de lograr un objetivo con la menor cantidad de energía consumida.

Indicadores Energéticos: Relación entre la cantidad de producto manufacturado con la cantidad de energía consumida, esta relación lleva por nombre indicadores energéticos. A nivel de empresa es común.

Recursos energéticos: Conjunto de medios con los que los países del mundo intentan cubrir sus necesidades de energía.

Intensidad Energética (IE): Se entiende por Intensidad Energética la cantidad de

energía usada por una unidad de medida de la producción o el servicio prestado.

Ahorro Energético: Es la cantidad de combustible, electricidad u otro que no se consume gracias a una mayor eficiencia u otra medida, sin afectar la función ni la calidad para la que se asigna la energía. (MAYNEGRA, 2007) (1)

1.4. Sus Principios

Como ya se conoce, el uso de las energías renovables no es un hecho novedoso, fueron ellas las primeras utilizadas por el hombre. La leña fue la primera fuente de energía para el ser humano, y la más importante durante la mayor parte de su historia. Era muy asequible porque en muchas partes del mundo crecían grandes bosques. En los tiempos antiguos también se utilizaban algunas otras fuentes de energía que sólo se encontraban en zonas puntuales: asfalto, carbón, turba de depósitos superficiales, y petróleo procedente de filtraciones de yacimientos subterráneos.

La situación cambió en la edad media cuando la leña se empezó a utilizar para fabricar carbón vegetal, que se empleaba para obtener metales a partir de sus minas. A medida que se talaban los bosques y disminuía la cantidad de leña disponible, en los comienzos de la Revolución Industrial, el carbón vegetal fue sustituido en la obtención de metales por el choque procedente del carbón. El carbón, que también se empezó a utilizar para propulsar las máquinas de vapor, se fue convirtiendo en la fuente de energía dominante a medida que avanzaba la Revolución Industrial.

Sin embargo la aparición de los combustibles fósiles relegó estas energías renovables por muchos años al olvido. En la actualidad el panorama ha cambiado, por una parte los problemas medioambientales debidos en un significativo por ciento a los procesos de conversión energética y en su totalidad a la acción indiscriminada del hombre sobre la biosfera y por otra parte la convulsa situación del mundo del petróleo (portador energético fundamental en la actualidad) que ha enfrentado tres crisis y en menos de 50 años han puesto de nuevo sobre el tapete las olvidadas energías renovables; y aunque es cierto que todavía enfrentan detractores cada día ganan más adeptos y aumenta su cuantía dentro de la satisfacción global de los requerimientos energéticos de la humanidad.

El alto costo de las inversiones iníciales a realizar limita en muchos países en vías de

desarrollo el empleo de las energías renovables; Cuba, dentro de estos países, tiene una privilegiada situación social debido a la alta conciencia energética de los cubanos, así como su educación medio ambiental inculcada desde las edades más tempranas, sin embargo no es ajena a las limitaciones económicas, a pesar de ello los cubanos no renuncian al empleo de estas fuentes de energía y mediante diversas vías en las que se incluyen los proyectos internacionales, se promueve el uso de las mismas.

Cuba consume unos 17 millones de toneladas de combustible convencional (ZAMORA, 2006) (2), a partir de 1990, debido a las limitaciones en la importación de los combustibles, el país ha venido incrementando las medidas de ahorro y uso racional y eficiente de la energía, constituyendo esta la principal fuente para el desarrollo, paralelamente se ha acrecentado la producción y el consumo de la energía procedente de fuentes nacionales. A éstas por convicción se les identifica como **fuentes alternativas de energía (FAE)** aunque también se le llama indistintamente energías alternativas o energía procedente de fuentes alternativas.

Las FAE hasta ahora consideradas en Cuba comprenden a las formas renovables y no renovables.

La energía eólica es ampliamente conocida y aplicada por diferentes pueblos desde la antigüedad en el desarrollo de la navegación, para moler granos y para el bombeo del agua. Fue remplazada por los fósiles baratos, pero mostró gran importancia, a partir de la crisis energética de la década del 70.

La energía solar posibilita la vida en todas sus formas, y la misma se presenta en diversas manifestaciones, en forma de alimento y en forma de combustible. El biogás constituye una abundante y barata fuente de energía y de fácil obtención a partir de desechos animales, vegetales e industriales. Esta energía puede ser utilizada en numerosos procesos que tienen incidencia en la economía, no solo por la generación de energía sino también por la producción de biofertilizantes de alta calidad.

Por la importancia de los combustibles tanto para el desarrollo económico como para el desarrollo medio ambiental en Cuba y el mundo es que se hace necesaria la gestión de los Portadores Energéticos particularmente para nuestro país con el objetivo de ser lo más racional posible evitando perdidas por robo o malversación y contribuyendo a realizar un estudio automatizado del consumo real de la entidad en

cuestión con el fin de lograr que se consuma lo necesario ayudando al desarrollo de la Revolución Energética y con esto apoyando la economía del país que disminuye sus importaciones de combustible.

1.5. Sistemas automatizados existentes

En el plano internacional existen sistemas informáticos que se encargan de administrar y controlar el consumo combustible entre otras funciones como por ejemplo:

SoftFlot es un Software para administrar flotillas de cualquier clase de vehículos, como puede ser Transporte Público, Transporte Pesado, Transporte de Carga, etc., el sistema crea una base de datos con toda la información de sus vehículos, llevando un control detallado de bitácora de combustible, control de mantenimientos preventivos y fallas, control de costos y presupuestos, llantas, trámites de gestoría, refacciones, depreciaciones, logística, liquidaciones, pago a proveedores, administración de contratos . SoftFlot requiere de **Windows** XP, 2003 o Vista, Espacio en Disco de 300 MB, Memoria RAM 512 MB para XP 1 G para Vista, Microsoft SQL Server 2000, 2005 para Clientes Remotos.

SoftFlot no cumple con las funcionalidades que se quieren desarrollar para resolver el problema del control de consumo y distribución de portadores energéticos aunque este sistema implementa funcionalidades que tributan al ahorro de portadores como bitácora de combustible, control de mantenimientos preventivos y fallas y otras, que permiten tener el control del consumo de combustible y el estado de los vehículos, además de que desarrolla otras funciones como son control de costos y presupuestos, trámites de gestoría, refacciones, depreciaciones, logística, liquidaciones, administración de contratos que se alejan de nuestro problema en cuestión y que no contribuyen a nuestro desarrollo, como también constituye una limitante los requisitos de instalación ya que nos obliga a trabajar sobre plataforma windows y se aleja de los intereses de la Aduana General de República de Cuba y el país de migrar a plataforma libre para independizarnos tecnológicamente. (3)

Gestión Estaciones de Servicio: Es una herramienta para la gestión integral de su estación de servicio. Sus características principales Panel gráfico de gestión de playa, con la distribución de islas y surtidores, Control de aforadores de combustible, Mini mercado y servicios, Manejo de turnos con apertura y cierres de partes de caja,

Control de playeros, Control de stock, Cuentas corrientes de clientes y proveedores, Multidepósitos, Gestión de Artículos, agrupamientos y subagrupamientos. Además todos los módulos de gestión administrativa básicos: facturación, compras, cuentas corrientes, caja, formas de pago, bancos, impuestos, Libro IVA ventas y compras, contabilidad integrada. Gestión Estaciones de Servicio requiere para su instalación PC Compatible. 256 MB RAM. 80 MB de espacio en disco rígido. Monitor con resolución de 1024x768, Cualquiera de las impresora para listados o fiscales existentes., Sistema Operativo Windows lo que impide seguir la política de la Aduana General de la República de Cuba de emigrar a plataforma libre, por lo que no es viable para darle solución a nuestro problema.

Gestión Estaciones de Servicio implementa funciones para el manejo de combustible haciendo uso eficiente de las funcionalidades para el ahorro de combustible pero todo esto desde el aforador de despacho de una estación de servicio y nuestro objetivo es controlarlo pero desde el nivel de la empresa además que sus requisitos de instalación no cumplen con la política del país ni de la Aduana General de la República de migrar a plataforma libre para lograr la menor dependencia tecnológica. (4)

WinPYME es un programa de gestión empresarial completo (producción, facturación, almacén, ingresos, gastos, proyectos, pedidos, beneficios, informes...) que, además, permite llevar un control de los costes de mano de obra y productos usados en un trabajo. **WinPYME** es un programa útil para cualquier tipo de empresa (autónomos, profesionales, industrias, etc.). Con WinPYME(Pequeña y Mediana Empresa) controlas: Clientes, Proveedores, Empleados, Pagos y Cobros, Productos y Componentes, con soporte para tallas y colores, lotes y fabricaciones, Cuentas, Anticipos, Incidencias de trabajadores, Entradas y salidas de almacén, Proyectos, Trabajos y Obras (costes y facturación: beneficio), Tiempos de mano de obra y herramientas empleados en los trabajos, obras y proyectos, los materiales y otros conceptos, así como los importes de coste y facturación para obtener el beneficio, Ventas - Ingresos (Cobros), Compras - Gastos (Pagos), Pedidos a proveedores, Los datos introducidos, con sus utilidades anti fraude (NIF, CIF, NIE, CCC, IBAN). (5)

WinPYME aunque sea una aplicación útil para cualquier tipo de empresas, no es idónea para resolver nuestro problema porque a pesar de que controla Clientes, Productos, Compras - Gastos (Pagos) y Los datos introducidos, con sus utilidades anti fraude, características que son importantes dentro de nuestra solución, no es viable

porque también tiene funcionalidades completamente fuera de objetivo de desarrollo en nuestra aplicación como Cuentas, Anticipos, Incidencias de trabajadores, Entradas y salidas de almacén, Proyectos, Trabajos y obras (costes y facturación: beneficio),

Tiempos de mano de obra y herramientas empleadas en los trabajos, obras y proyectos, los materiales y otros conceptos, así como los importes de coste y facturación para obtener el beneficio además de que no cuenta con el mayor de nuestros objetivos y requisitos de desarrollo que es el control de los portadores energéticos. (6)

Mantenimiento de flota de vehículos: este sistema tiene fichero de vehículos; dónde darán de alta los vehículos con control de seguro, consumo de combustible, consumo de ruedas y viajes, Gastos e ingresos varios, Mantenimiento preventivo, por km o por fechas, Control de reparaciones, Archivo documental, donde introduce todos los documentos recibidos para ese vehículo, Fichero de empleados, consumo de combustible por empleados siniestros por empleado, el gastos e ingresos por empleado, viajes por empleado, equipo de protección individual, con control de pago a empleados, Listados por pantalla o impresora, con selección o filtro a crear por el usuario, requisitos de instalación Windows 95 o superior y 32 Megas de memoria, como requerimientos mínimos para su instalación dejando a un lado los intereses de la Aduna General de la República de migrar sobre plataforma libre .

Mantenimiento de flota de vehículos tiene funcionalidades que tributan a los requisitos que debe cumplir nuestra aplicación como consumo de combustible, consumo de ruedas y viajes por carro que es la columna vertebral de nuestro desarrollo aunque no con el mismo énfasis y sobre los mismo mecanismo que se quiere implementar nuestra solución, además de que tiene otras funcionalidades que pudieran en un futuro ser objetivo de estudio para una nueva versión, aunque este es un sistema con características muy parecidas a nuestra posible solución no es utilizable para resolver el problema que nos lleva a esta investigación además de que no es una herramienta que trabaje sobre plataforma libre por lo que nos hace dependiente tecnológicamente de las grandes transnacionales del software y no cumple los objetivos de nuestro país ni de la Aduana General de la República de Cuba de migrar a plataforma libre. (7)

En el ámbito nacional existen sistemas en otras entidades que si implementan temas

como son el ahorro de energía entre otros tantos en el ámbito nacional, como fuera de nuestras fronteras, entre ellos: Servicios Técnicos y Suministros (STS), Grupos Eléctricos y Servicios Eléctricos (GEYSEL) del Mintur, Empresa de

Automatización Integral (CEDAI), por ejemplo este último implementa temas como:

- Optimización de climatización (60%)
- Control de alumbrado
- Control de bombeo, ventilación y extracción
- Supervisión cámaras frías
- Medición de portadores energéticos
- Reportes, análisis y soluciones

Y a nivel de aduana están:

El Sistema Aduanero Automatizado (SIDUNEA), Sistema de Información Aduanero (SIA), Sistema Informático Aduanero Uniforme Centroamericano (SIAUCA), Tecnología de la Información para el Control Aduanero (TICA), que implementan diversas funciones dentro del marco de la aduana como son prevenir, investigar y perseguir las infracciones a las legislaciones aduaneras, control y administración de la gestión aduanera, controlar efectivamente la recaudación de los impuestos aduaneros, entre otras funcionalidades, sin embargo, ninguno aborda el tema de los portadores energéticos.

Ninguno de estos Sistemas antes mencionado tanto a nivel de país como de Aduanas guarda semejanza con el trabajo que se quiere realizar en la Aduana General de la República de Cuba, que se basa en el control de la utilización eficiente de los Portadores Energéticos. Siendo esa una de las razones por la que se hace necesario un estudio y análisis que permita desarrollar el tema. (8)

1.6. Descripción de la solución.

El polo productivo Sistema Tributarios y Aduanales está interesado en conjunto con la Aduana General de la República de Cuba en desarrollar una aplicación web que sea capaz de controlar y administrar el proceso de demanda, asignación y distribución de portadores energéticos en los 13 puntos aduanales del país regido por documentos oficiales estipulados en la constitución del país para esta operación como el CDA-001 y otros, mostrando hoja de resumen en todos los casos, donde se le asignara combustible al carro que esté funcionando, donde se lleve el control del carro que está en reparaciones y las funciones que este desarrolla. Este sistema debe ser capaz de interactuar con el SUA (Sistema Único de Aduana) a la hora de distribuir en las tarjetas de los carros el combustible tiene que existir uniformidad en los datos distribuidos en este sistema, el planificador y el económico del SUA.

1.7. Tendencias y Tecnologías actuales

En la actualidad, la tendencia en el desarrollo de soluciones informáticas está dirigida hacia el desarrollo Web por las ventajas que trae una solución de este tipo por ejemplo simplifica el código, reduce el tamaño de los archivos y el tiempo de desarrollo, etc., se hace necesario un estudio de las diversas metodologías y lenguajes de diseño y desarrollo que más se usan a nivel mundial y enfocarnos en las mejores prácticas, de manera que se adecuen al trabajo que se quiere desarrollar.

1.7.1 Metodologías y herramientas de diseño usadas en el mundo.

La evolución del diseño de software, como parte del proceso de desarrollo de software, es un proceso continuo que se ha ido produciendo durante las últimas tres décadas. Los primeros trabajos sobre diseño se centraron sobre los criterios para el desarrollo de programas modulares y los métodos para mejorar la arquitectura del software de una manera descendente. Los aspectos procedimentales de la definición del diseño evolucionaron hacia una filosofía denominada *programación estructurada*. Posteriores trabajos propusieron métodos para la traducción del flujo de datos o de la

estructura de los datos, en una definición de diseño. Nuevos enfoques para el diseño proponen un método *orientado a objetos* para la obtención del diseño.

Ante un nuevo proyecto se pueden adoptar distintas formas de aproximación. A esto se le llama metodologías del diseño. Estas metodologías se pueden emplear de forma unitaria o combinándolas entre sí, para obtener un nuevo enfoque. Se debe tener en cuenta que también influye el ámbito de aplicación, y las disciplinas implicadas en el proceso del proyecto.

1.7.1.1 Metodologías de diseño

- **Diseño Axiomático**, cuando se emplean axiomas para tomar decisiones. Este enfoque se basa principalmente en las necesidades funcionales del cliente.
- **Diseño Orientado al Uso**, cuando el diseño se centra en las tareas asociadas al uso y sus objetivos. Este enfoque penaliza la usabilidad, ya que incrementa la curva de aprendizaje, pero logra solventar problemas complejos o de alta criticidad.
- **Diseño Centrado en el Usuario** (UCD User Center Design) que es la metodología de diseño más habitual en el mundo. Este enfoque coloca todas las necesidades, deseos y limitaciones del usuario como núcleo del proceso de diseño. Por lo cual esta metodología conlleva por definición investigación y análisis del usuario. Así mismo implica que el proyecto debe ser faseado para garantizar los resultados. Dentro del UCD hay metodologías especificas, como son KES (Kansei Engineering System) o QFD (Quality Function Deployment).
- **Enfoque Ascendente**, que consiste en partir de un elemento o de las características individuales de los elementos para desarrollar la totalidad del producto. Este es el método empleado por Google en sus adds de productos.
- **Enfoque descendente**, consistente en generar el producto desconociendo los elementos que lo configuran, esta técnica se emplea para desarrollar por ejemplo templates web.

Algunos principios que pueden resultar de ayuda a la hora de tomar decisiones son:

- **Navaja de Ockham**, consiste en eliminar los elementos innecesarios del diseño con el fin de obtener productos lo más sencillos posibles, de modo que se reduce la posibilidad de que existan incoherencias.
- **Principio KISS** (*Keep It Short and Simpl*e o *Keep it simple,stupid*) se emplea cuando la sencillez es un objetivo del diseño. El principio Kiss no es en sí mismo una metodología sino una estrategia de diseño y sigue el reduccionismo metodológico de Ockham.
- **Timtowtdi**, (*There is more than one way to do it*) hace referencia a que existen múltiples soluciones a un problema, es un principio muy popular en programación. (9)

1.7.1.2 Patrones de Diseño

"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera, que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma". (10)

Un patrón de diseño es una solución estándar para un problema común de programación, una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios, una estructura de implementación que logra una finalidad determinada, manera práctica de describir aspectos de la organización de un programa. (11)

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 no ayuda a no cometer errores ya descritos (12).

Modelo Vista Controlador (MVC), es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC, donde la vista es la página HTML y el código que

provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos.

Este patrón divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

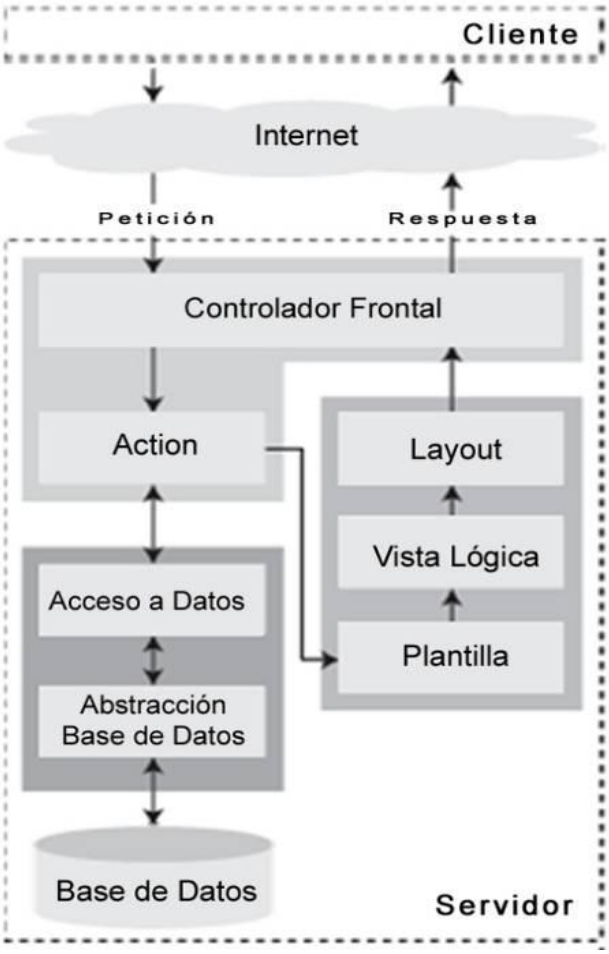


Figura 1.1 Implementación del Patrón MVC por Symfony

EXPERTO: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada (13).

CREADOR: Este patrón como su nombre lo indica es el que crea, el guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

1 B contiene A

2 B es una agregación (o composición) de A

3 B almacena A

4 B tiene los datos de inicialización de A (datos que requiere su constructor)

5 B usa A.

A la hora de crear objetos se deben tener en cuenta las características de la clase (14).

BAJO ACOPLAMIENTO: El acoplamiento es una medida de fuerza con que un elemento esta a, tiene conocimiento de, confía en, otros elementos. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema (15)

ALTA COHESION: La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento, Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no

desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo (16)

CONTROLADOR: Es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos. Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo (17).

DECORATOR: Responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto nos permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera.

- Añadir objetos individuales de forma dinámica y transparente
- Responsabilidades de un objeto pueden ser retiradas
- Cuando la extensión mediante la herencia no es viable
- Hay una necesidad de extender la funcionalidad de una clase, pero no hay razones para extenderlo a través de la herencia.
- Hay la necesidad de extender dinámicamente la funcionalidad de un objeto y quizás quitar la funcionalidad extendida.

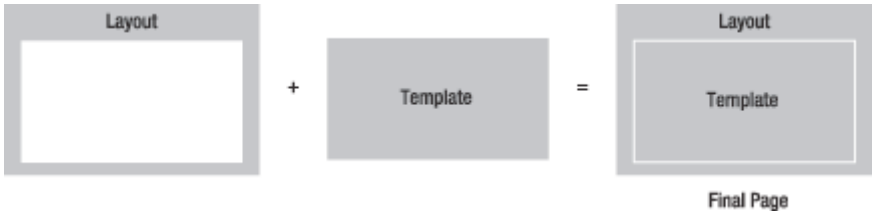


Figura 1.2 Ejemplo de Implementación del patrón Decorator en Symfony

FACTORY METHOD (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear (18).

1.7.1.3 Herramienta de diseño

Herramienta: Aplicación empleada para la construcción de otros programas o aplicaciones. (Definición de Herramienta, 2004) (19)

Herramienta CASE (Ingeniería Asistida por Computadora, CASE):

Es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de CASE para automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas. (SANCHEZ, 2007) (20)

El desarrollo de software ha logrado un gran avance en el mundo moderno, parte de este logro es gracias a las herramientas de modelado ya que es el medio donde se modela el sistema a desarrollar, guiándose por una metodología y utilizando algún lenguaje de modelado.

Herramientas escogidas para desarrollar este trabajo:

Visual Paradigm: Es una herramienta CASE que permite tanto la ingeniería directa como inversa, la misma soporta varios lenguajes de programación para hacer la generación de código o la ingeniería inversa. También permite el diseño en el lenguaje UML (Unified Modeling Language), BPMN (Business Process Modeling Notation) y permite usar la metodología BPM (Business Process Management). Por otro lado esta herramienta se integra en gran variedad de IDEs (Entornos de desarrollo integrados). (COLOMA, 2007) (21)

Algunas características de Visual Paradigm:

- Es profesional
- Es amigable
- Contiene facilidades para redactar especificaciones de casos de uso del

sistema.

- Sincronización entre diagramas de entidad-relación y diagramas de clases
- Generación de código / Ingeniería inversa:
- Soporte de UML versión 2.1
- Interoperabilidad con otras aplicaciones

1.8 Herramientas utilizadas para desarrollar el sistema.

En la actualidad la efervescencia de herramientas que ayudan a desarrollar aplicaciones está bastante marcada, si tenemos en cuenta que la tendencia al desarrollo de aplicaciones está encaminada a la tecnología web, pues tendríamos tela por donde cortar para decidir que entorno de desarrollo utilizar, pero la selección se reduce a los objetivos del polo de desarrollo Sistemas Tributarios y Aduanales de la universidad de las ciencias informáticas.

1.8.1 Lenguaje de programación a utilizar.

En los últimos tiempos han surgido variedades de lenguaje de programación cada vez con más facilidades para el desarrollador y el educador, como el java, como php 5, como .NET, que son tecnologías más amigables a la hora de desarrollar aplicaciones de todas las envergadura, pero para que un lenguaje atraiga a desarrolladores, vendedores y educadores debe resolver un problema particular tan bien que sea capaz de superar la resistencia de adoptarlo.

La programación orientada a objetos (POO) ha tenido un desarrollo de cerca de 30 años; es una técnica que se enfoca en los datos (objetos) y en la manera de llegar a ellos (interfaces), no en las herramientas que se utilizan para manejarlos, a fin de crear “módulos” que interactúen entre sí para lograr el objetivo del programa.

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

¿Qué es PHP?

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas Web dinámicas.

Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo (22).

Características de PHP:

Gratuito. Al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.

Versatilidad. PHP puede usarse con la mayoría de sistemas operativos, ya sea basados en UNIX (Linux, Solares, FreeBSD...), como con Windows, el sistema operativo de Microsoft.

Enorme eficiencia. Con escaso mantenimiento y un servidor gratuito (en nuestro caso, Apache), puede soportar sin problema millones de visitas diarias.

Sencilla integración con múltiples bases de datos. Esencial para una página Web verdaderamente dinámica, es una correcta integración con base de datos. Aunque MySQL es la base de datos que mejor trabaja con PHP, puede conectarse también a PostgreSQL, Oracle, dbm, filePro, interbasem o cualquier otra base de datos compatible con ODBC.

Partiendo de que nuestro sistema tiene que interactuar con SUA que es una aplicación PHP, decidimos que para el problema a resolver y con la idea de desarrollar una aplicación lo más libre de licencias posible y valorando que PHP cubre todas las necesidades se puede aceptar este lenguaje como el ideal para el desarrollo, lenguaje definido en la línea base de la arquitectura del polo de desarrollo Sistema Tributarios y Aduanales del que nuestro trabajo está integrado y regido (23).

1.8.1.1 Framework a utilizar.

Un Framework es un conjunto de clases o estructuras que implementan los componentes de una aplicación genérica, así como también componentes concretos que cumplen a cabalidad tareas concretas. Para desarrollar programas completos, los desarrolladores buscan e instancian los componentes apropiados.

Realmente no existe una definición oficial de Framework, pero todos los autores coinciden en la utilización de un tema común: la reutilización. Una definición dada por R. E. Johnson, and B. Foote en 1988 en su publicación “Designing Reusable Classes (24).

"Un Framework es un conjunto de clases que personifican un diseño abstracto para soluciones de una familia de problemas relacionados..."

Symfony Project

Symfony es un Framework para PHP5 patentado bajo licencia MTI, es compatible con la mayoría de gestores de bases de datos, MySQL, PostgreSQL, Oracle y SQL Server de Microsoft, entre otros en dependencia del tipo de abstracción de la Base de Datos que se utilice.

Symfony 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. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que

se crea una nueva aplicación web.

Características de Symfony

- Proyecto de Symfony es un framework muy amplio, e incluye un verdadero ORM, de nombre Propel, que es otro proyecto de código abierto y, probablemente, una de las mejores soluciones ORM para PHP.
- Incluye Creole para la capa de abstracción de base de datos y Mojavi para la capa Model-View-Controller.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicación empresarial y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros (25).

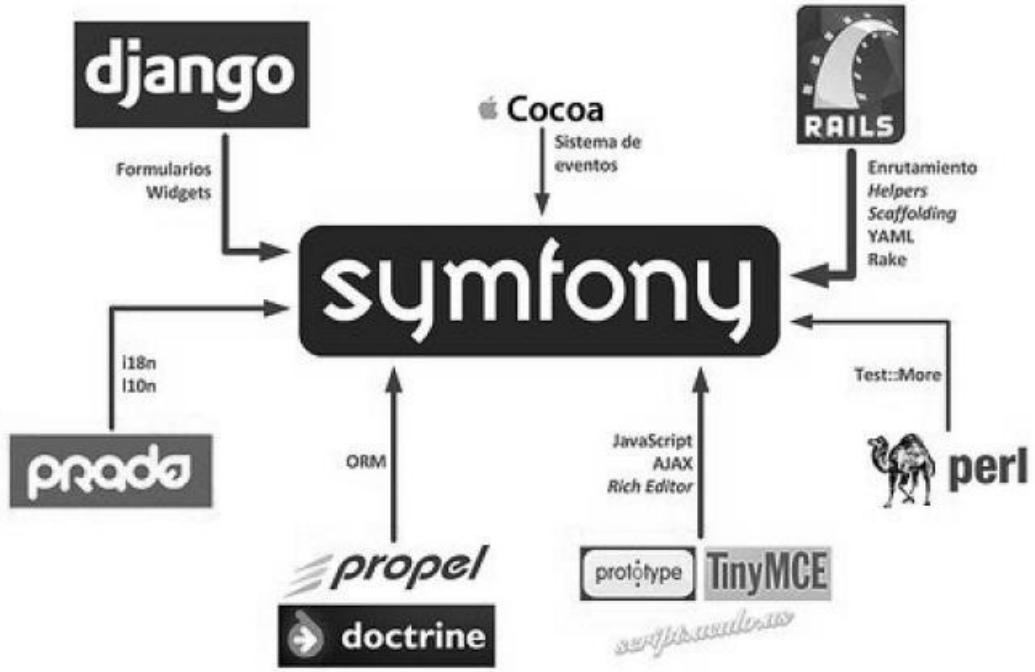


Figura 1.3 Composición de la estructura de Symfony por otros Frameworks.

Utilizaremos el framework de php Symfony 1.1 porque es el más viable de acuerdo con sus características para desarrollar, es el más completo de los frameworks de php para el desarrollo de sistemas grandes a diferencia de Cakephp que es más bien para el desarrollo de aplicaciones pequeñas, y admite Object-Record-Mappe (ORM), tiene soporte integrado para ajax además de que tiene módulo integrado para manejar autentificación de usuario todo esto a diferencia de Zend que es otro de los framework estudiados por nuestro equipo, además de que el polo sistemas tributarios en su arquitectura base definió que este es el framework a utilizar (26).

1.8.1.2 Framework de JavaScript a utilizar

Ext JS es un Framework de JavaScript que permite realizar aplicaciones Web enriquecidas basándose en tecnología AJAX, JSON, DHTML y DOM. Ext 2.0 está patentado bajo licencia LGPL lo que posibilita su uso para aplicaciones empresariales privativas de código cerrado.

Ext JS brinda la posibilidad de utilizar un gran número de componentes visuales que mejoran considerablemente la calidad de las aplicaciones. Brinda la posibilidad de validaciones de formularios de todo tipo, basándose en expresiones regulares y tipos

de datos. Trae implícitos componentes como vista en arboles, arrastrado y soltado, cambio de tamaño de imágenes, rejillas, paginado, agrupado de objetos, tabs, asistentes, entre otros muchos.

La utilización de un Framework de Javascript como Ext JS facilita la separación de las capas de la vista con la del controlador desde el punto de vista productivo ya que el código utilizado en la primera es solamente JavaScript y no es necesario utilizar ningún tipo de código PHP, así los desarrolladores pueden centrarse más en el aprendizaje de un solo lenguaje. Además al soportar serialización de objetos mediante tecnología JSON permite que los datos enviados desde el controlador como respuesta a la vista contengan solo las propiedades de dichos objetos, pero no el comportamiento, minimizando los posibles errores de programación y los accidentes de que los objetos sean modificados erróneamente desde la vista.

1.8.2 Oracle

El SGBD Oracle, fabricado por Oracle Corporation, utiliza la arquitectura cliente/servidor. Ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Así ofrece un servidor de bases de datos híbrido. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad.

Los tipos objeto de Oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos. (Costilla, C, 2002) (27).

A partir de la versión 10 del año 2004, se añade a los servidores la capacidad de funcionar según el paradigma de “Grid” (o rejilla) y se ofrecen mejoras en la administración e integración de algunos elementos que previamente no funcionaban correctamente juntos.

1.8.2.1 Ventajas de Oracle.

- Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia.
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente.
- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos.
- Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix.
- Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal. Las copias de la Base de Datos productiva pueden estar en modo de lectura solamente.

1.8.2.2 Desventajas

- Es un producto de elevado precio por lo que por lo general se utiliza en empresas muy grandes y multinacionales
- Los costos de soporte técnico y mantenimiento son elevados
- Vulnerabilidades en la seguridad de la plataforma, se hace necesario aplicar parches de seguridad

1.9 Conclusiones Parciales

En este capítulo se presentaron los conceptos más significativos para la correcta comprensión de este trabajo. Se realizó un estudio en el que se demuestra que el desarrollo de aplicaciones Web hoy en día está regida por la utilización de Framework implementados bajo el patrón arquitectónico MVC que faciliten el diseño e implementación de los sistemas y el estudio realizado por los Framework para PHP más significativos demuestra que el que más se ajusta a las necesidades de las aplicaciones empresariales de gestión que requieran de Base de Datos Oracle 11 y PostgreSQL es Symfony utilizando Propel para manejar la persistencia de los datos.

Capitulo 2: Diseño e Implementación del sistema

2.1 Introducción

En el presente capitulo se desarrollan los flujos de trabajo diseño e implementación, teniendo como principal objetivo obtener los diagramas de clases del diseño, diagrama de entidad relación y diagramas de secuencia para guiar al equipo de desarrollo en la implementación de la aplicación con mayor precisión y en el menor tiempo posible además de obtener los diagramas de despliegue y de componente para posibilitar el mantenimiento y reutilización del código de manera más eficiente.

2.2 Modelo del Diseño

El Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso especificando cómo los requisitos funcionales y no funcionales tienen impacto en el sistema. Este artefacto constituye la entrada fundamental utilizada para el correcto desarrollo de las actividades de implementación. Entre los propósitos del Modelo de Diseño se encuentran:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo

2.2.1 Modelado mediante Estereotipos Web

Para la realización de los Diagramas de Clases del Diseño será utilizada la extensión de UML para el modelado de aplicaciones Web, (28). Esta extensión presenta como elementos más significativos a tres clases UML: Server Page, Client Page y Form empleadas para el código servidor, código cliente y formularios respectivamente, permitiendo además representar ficheros contenedores de sentencias script.

A continuación se brinda una explicación de cómo son usados estos estereotipos en el diseño de la propuesta del sistema y qué representa cada cual:

<<Server Page>>: Representa la clase que tiene código que se ejecuta en el servidor, la cual se encarga de construir (build) o generar el resultado HTML y/o realizar peticiones a la capa inferior.

<<Client Page>>: Es una página Web con formato XHTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para cualquier función dentro de la página. Cada página cliente es construida por una sola página de servidor.

<<FormHTML>>: Es una colección de elementos de entrada que están contenidos en la página cliente. Sus atributos son los elementos de entrada del formulario (input boxes, text areas, radio buttons, check boxes, hidden fields, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST y se comunican con las páginas servidores mediante submit.

En la Tabla 2.1 están representadas las relaciones que se establecen entre los tres elementos claves mencionados anteriormente.

RELACIONES ENTRE LOS ELEMENTOS DE DISEÑO

DESDE/HASTA	Client Page	Form	Server Page
Client Page	<<link>> <<redirect>>	aggregation	<<link>>
Form	aggregation by		<<submit>>
Server Page	<<build>>		<<include>>

Tabla 2.1 Relaciones entre los elementos de diseño.

- Entre las páginas servidoras pueden existir relaciones de inclusión (<<include>>)
- Las páginas servidoras construyen el resultado XHTML que conforma el código cliente (<<build>>)
- Los formularios forman parte del resultado XHTML (<<aggregation/ aggregation by)
- Los formularios envían los datos al código servidor para su procesamiento (<<submit>>)
- Entre las páginas clientes pueden existir vínculos (<<link>>) o re-direccionamientos (<<redirect>>)
- Las paginas clientes pueden incluir ficheros script (<<include>>)

2.2.2 Diagramas de clases del diseño

Un diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de diseño de un sistema para crear el diseño conceptual de la información que se manejara en el mismo y los componentes que se encargaran su funcionamiento así como la relación entre ellos como se muestra en la **figura 2.1**.

Diagrama de clase del diseño Gestionar Demanda.

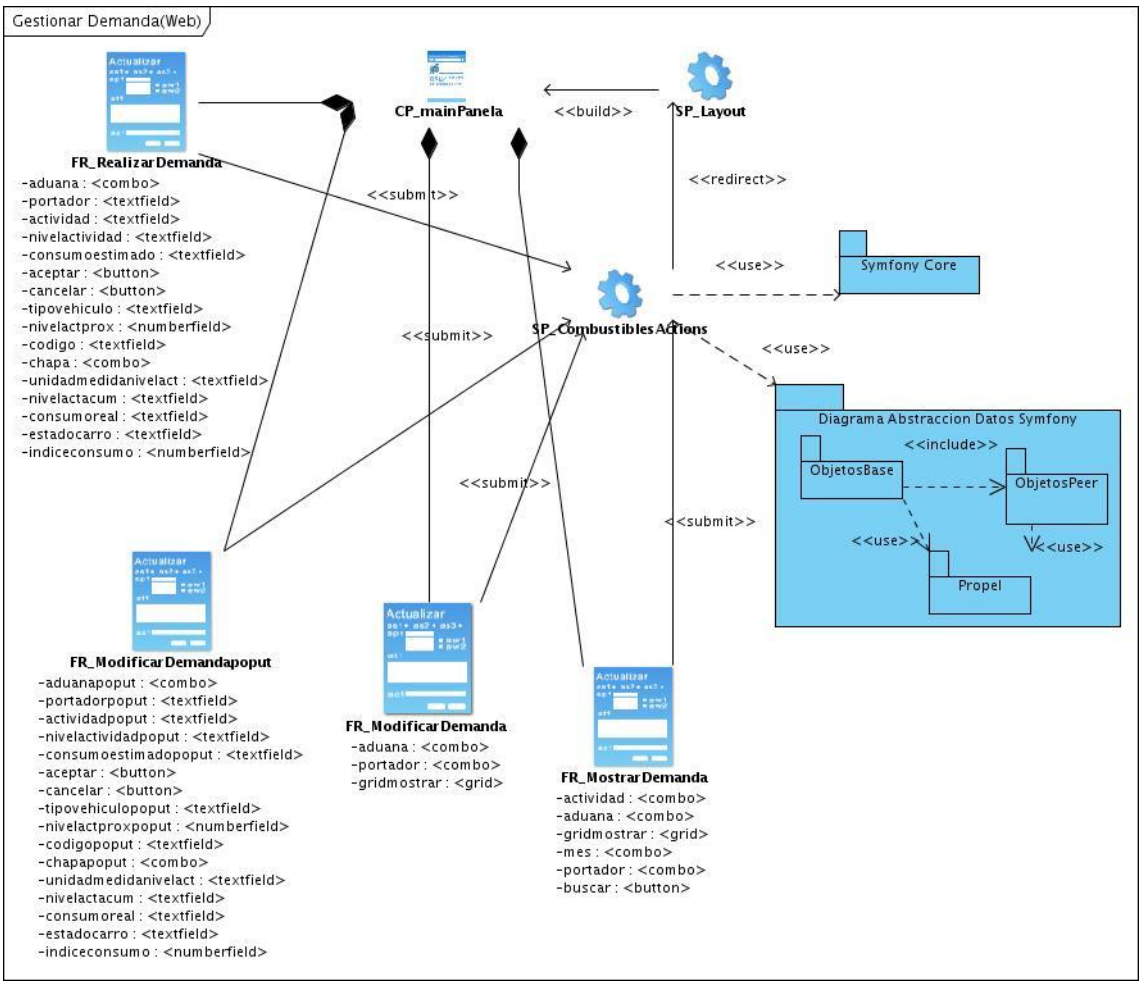


Figura 2.1 Diagrama de clase del diseño Gestionar Demanda

El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades, y su principal objetivo es la elaboración de los diagramas de clases de diseño, que muestra las clases participantes en la realización en un caso de uso con

todos sus atributos. Los diagramas de las clases del diseño correspondientes a los casos de uso se muestran en el Anexo I.

2.2.3 Diagramas de Interacción del Diseño

Los Diagramas de Interacción son los artefactos que UML brinda para expresar las interacciones entre objetos para cumplir con los requerimientos del sistema. Entre los aspectos más importantes a considerar para la realización de estos diagramas se encuentra:

- Se clasifican en 2 tipos: Diagramas de Secuencia y Diagramas de Colaboración
- Modelan aspectos dinámicos del sistema
- Se utilizan para realizar un traza de la ejecución de un escenario
- A cada escenario le corresponde un diagrama de interacción
- Una interacción es un conjunto de objetos y sus relaciones, incluyendo los mensajes mediante los cuales pueden establecer comunicación
- Un diagrama de secuencia destaca la ordenación temporal de los mensajes
- Un diagrama de colaboración destaca la organización estructural de los objetos que envían y reciben mensajes
- Un diagrama de interacción contiene:
  - Objetos
  - Enlaces.
  - Mensajes
- Un mensaje de interacción puede contener:
  - Notas
  - Restricciones

2.2.3.1 Diagramas de Secuencia

Los diagramas de secuencia como se muestran en las **figuras 2.2, 2.3, 2.4** son aquellos diagramas de interacción que destacan la ordenación temporal de los mensajes mediante los cuales pueden establecer comunicación los objetos representados en el mismo. Las características más distintivas de los diagramas de secuencia son las que se enuncian a continuación:

- Se realiza un diagrama de secuencia por cada escenario.
- Permite en las fases iniciales del diseño:
    - Analizar detalladamente el comportamiento de un escenario.
    - Obtener nuevas clases y objetos en un escenario (enriquecimiento del diccionario de clases).
    - Detectar las funcionalidades de las clases para llevar a cabo la tarea encomendada en el escenario.
- Se utilizan en las fases de prueba para validar la codificación.

Diagrama de secuencia Realizar Demanda

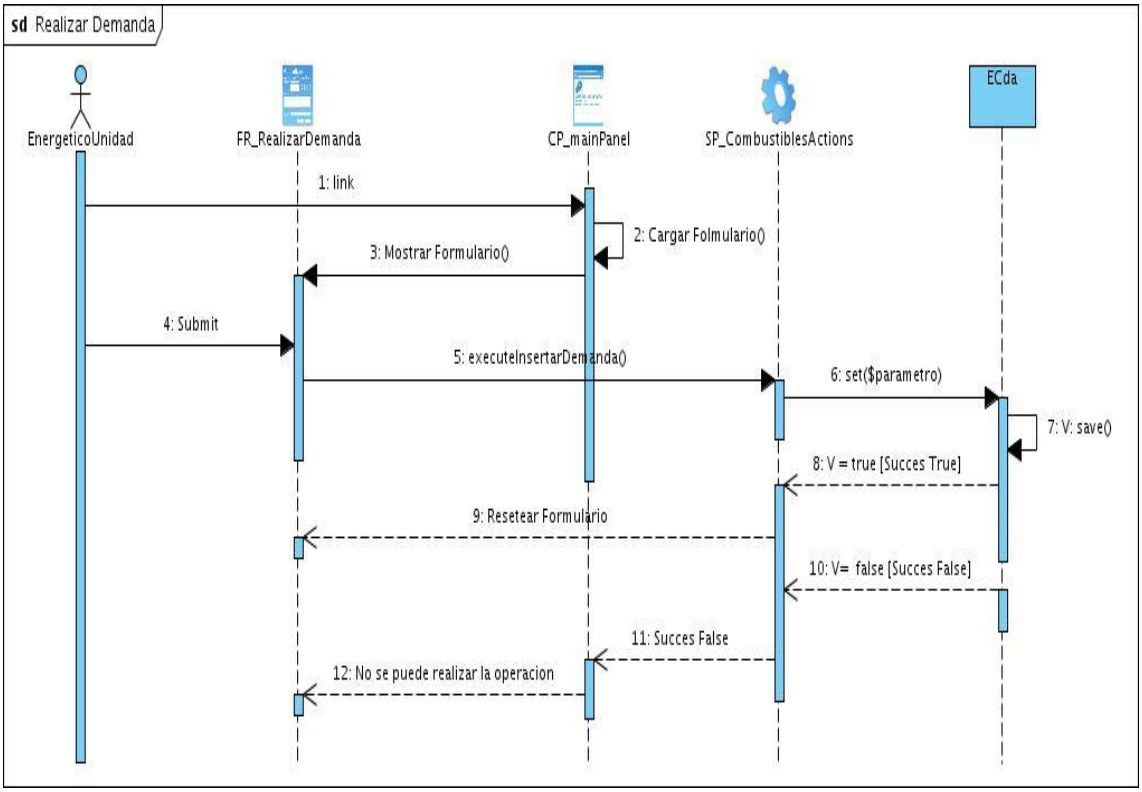


Figura 2.2 Diagrama de secuencia Realizar Demanda

Diagrama de secuencia Modificar Demanda

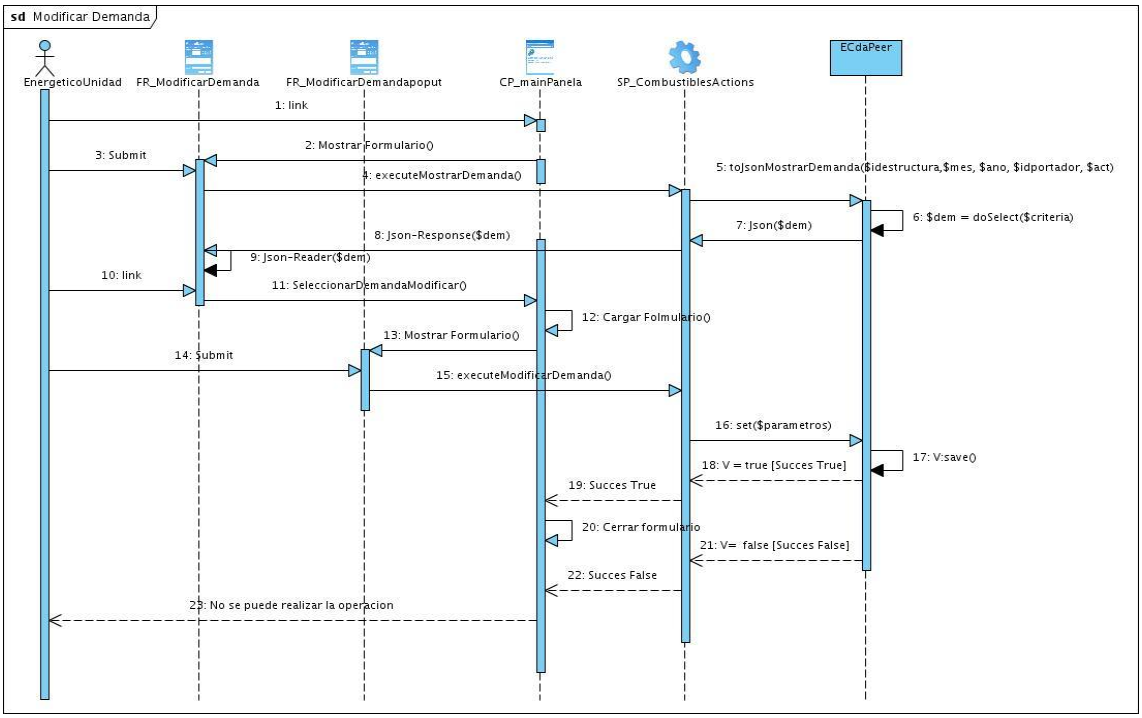


Figura 2.3 Diagrama de secuencia Modificar Demanda

Diagrama de secuencia Mostrar Demanda

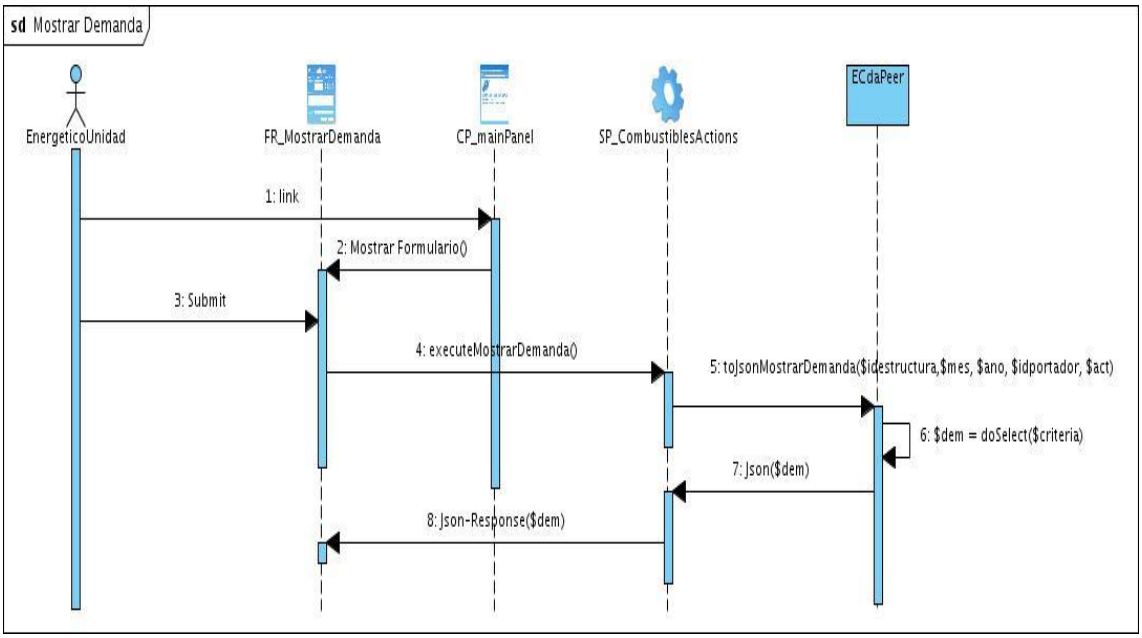


Figura 2.4 Diagrama de secuencia Mostrar Demanda

Los diagramas de secuencia correspondientes a los casos de uso arquitectónicamente significativos se muestran en el Anexo II.

2.3 Diseño de la Base de Dato

Durante el flujo de trabajo de diseño, concebir e implementar la base de datos, es uno de los hitos fundamentales de la elaboración de todo sistema que gestione información abundante. La base de dato en el desarrollo de este trabajo es de vital importancia ya que se trabaja sobre clases generadas por el framework utilizado, a continuación en la **figura 2.5** se muestra el diagrama entidad relación desarrollado ver descripción de las tablas en Anexo III.

Modelo entidad relación

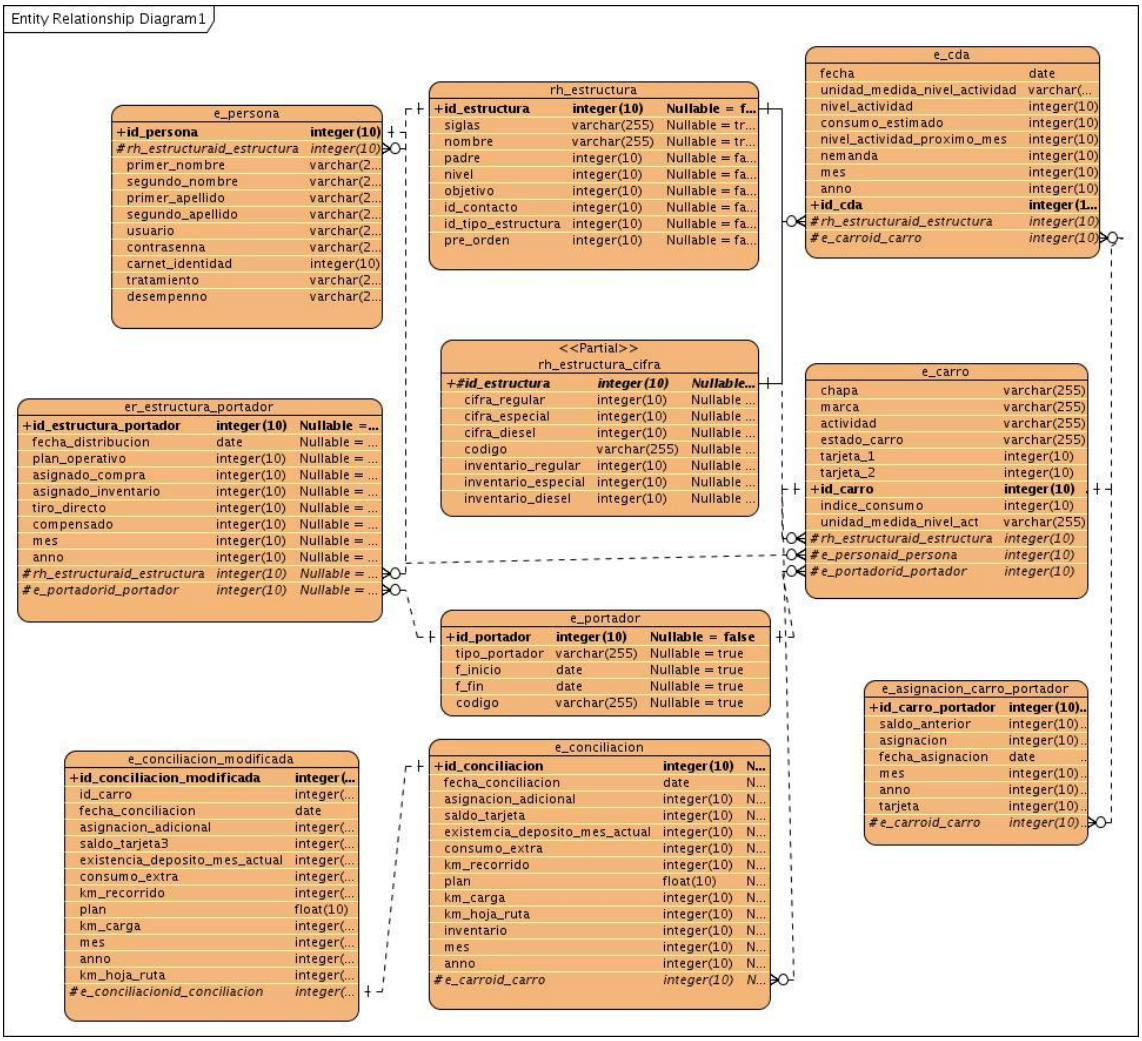


Figura 2.5 Modelo entidad relación

2.4 Principios de diseño que se aplican

El diseño del sistema está orientado directamente hacia los usuarios finales, por lo que se considera que es una parte fundamental dentro del proceso de desarrollo, pues es la parte del sistema con que el usuario interactúa y que le facilita además el acceso a los recursos.

En general los usuarios finales no están familiarizados con las ciencias de la computación, por lo que se puede decir que no están interesados en la parte interna de la aplicación (el código), sino en cómo se le muestra y cómo usarla. De acuerdo con lo anterior y en concordancia con las características de los usuarios se trazaron los siguientes principios de diseño:

- Mostrar la menor cantidad de elementos posibles en cada interfaz, evitando siempre que sea posible que el usuario tenga que desplazarse horizontalmente dentro de la interfaz mostrada.
- Se seguirán patrones de diseño para los elementos que conforman las interfaces, en cuanto a colores, tamaño, formas; además no se utilizarán colores fuertes ni brillantes.
- El diseño debe proporcionar un entorno agradable que contribuya al entendimiento por parte del usuario de la información presentada.
- La interfaz ha de ser simple, intuitiva, fácil de entender y de aprender; independientemente de la experiencia, conocimientos, nivel cultural, capacidades físicas y mentales de los usuarios finales del sistema.

2.5 Tratamiento de errores

Para garantizar el correcto funcionamiento de cualquier sistema es imprescindible identificar y controlar los posibles errores que se pueden presentar a la hora de interactuar con el software. En el sistema propuesto se tratan estos errores de forma tal que las interacciones con la base de datos (inserción, eliminación, modificación, etc.) se realicen de forma correcta. Para lograr esto se establecieron mecanismos de validación que comprueben la corrección de los datos a tratar; además en los formularios se insiste en que el usuario introduzca la menor cantidad posible de datos, evitando así incoherencias e incorrecciones en los mismos, en el caso de la entrada de datos por parte del usuario se implementarán funciones que validen dicha entrada

para que, de existir errores, se muestren mensajes que ilustren la incorrecta inserción, modificación o mala manipulación de datos en general.

2.6 Estándar de codificación

Actualmente se hallan estándares de codificación para la mayoría de los lenguajes de programación existentes. El uso de ellos partiendo de las convenciones definidas permite una mejor comunicación entre los programadores creando condiciones para la reusabilidad y mantenimiento de los sistemas.

Para definir el estilo de codificación a seguir en la aplicación se utilizó la notación estándar establecida para aplicaciones desarrolladas en PHP (PHP Coding Standard), que mayormente está basada en el estándar de código para aplicaciones en C++ (C++ Coding Standard) [COD03].

Las etiquetas de apertura y cierre del lenguaje serán de la forma <?php ¿>, ya que siempre están disponible en cualquier configuración.

Para nombrar las variables se seguirá la regla de escribir los identificadores con letras minúsculas y en español, no se usará separador entre palabras tratando de usar nombres sugerentes a la acción de la variable.

Los campos id son palabras que identifiquen el campo. Ejemplo estructuraasig. Los arreglos empezarán con el identificador array y las palabras no se separarán.

En el caso de las clases se pondrá delante el identificador del módulo. Ejemplo: ENombreClase y para los métodos las palabras continuas deben comenzar con mayúsculas. Ejemplo: MostrarDemanda.

Para comentar el código se utilizará, en el caso de una línea, al final de la misma el carácter “//“y seguido el comentario y en el caso de un bloque se utilizarán los caracteres “/* */”. Se usará una identificación en el código de cuatro espacios para facilitar la lectura de éste.

Las llaves se usarán poniendo la llave inicial en una línea para ella sola, y en su respectiva columna la llave final también en una línea.

Los nombres de las tablas se escribirán en minúscula con el identificador del módulo separado por línea abajo “_”. (Ejemplo: e_cda), en el caso de ser un nomenclador que sea la conexión al esquema del SUA se mantendrá el mismo formato con que está definido en el polo.

2.7 Comunicación entre capas

La comunicación entre las capas en las que se divide la arquitectura está dada por mantener los paradigmas de la Programación Orientada a Objetos y la posibilidad de realizar sistemas con mayor portabilidad se utiliza entre el cliente y el Controlador Frontal, y viceversa, tecnología JSON, con esta se serializan los datos pasados por el usuario en Objetos PHP. Esto permite utilizar una interfaz de cualquier tipo para interactuar con el servidor de aplicación además de la seguridad que brinda este tipo de comunicación en materia de desarrollo, ya que los implementadores de interfaz de usuario no tienen acceso al comportamiento de los objetos enviados desde la capa controladora. Siempre que esto sea posible la comunicación será utilizada en combinación con AJAX, para permitir mayor velocidad y dinamismo en la interacción de los sistemas por vía Web. La utilización de AJAX con comunicación basada en XML, se utilizará para los casos en los que sea necesario por no ser factible utilizar JSON. La comunicación entre las capas del Controlador y el Modelo, gracias a la implementación del ORM, está dada por el envío de objetos que contienen la información necesaria para manejar las peticiones del usuario o las acciones a realizar. En caso de que los arreglos de datos a utilizar contengan múltiples registros serán manejados mediante arreglos que contienen los Objetos referentes a las clases del Modelo. Para manejar la comunicación con la Base de Datos se utiliza Creole, librería que trae el Propel 1.2 para la abstracción de la BD.

2.7.1 Ejemplo de comunicación entre capas

El usuario interactúa con la pagina principal (**ver figura 2.6**) del sistema donde está la barra de menú con las funcionalidades que brinda la aplicación, el usuario selecciona la acción a realizar, el sistema muestra la pantalla correspondiente, para ejemplificar mejor esta comunicación se muestra el caso de uso Gestionar Demanda comenzando por el flujo Realizar Demanda como se ve en la **figura 2.7**.

Figura 2.6 Pantalla Principal

Figura 2.7 Pantalla Realizar Demanda

Para realizar la demanda el usuario debe cargar el formulario con los datos correspondientes, con este fin se realizan peticiones AJAX para llenar los campos Nivel Actividad y Consumo Real ambos referidos al acumulado hasta el momento, además de que se utiliza la creación de JSON para cargar los campos mostrables al usuario como se ve en la **figura 2.8** que muestra el código Java Script correspondiente a la petición AJAX para el campo Nivel Actividad y en la **figura 2.9** la petición a la

controladora para cargar un combo pasándole un objeto JSON.

```
    listeners: {
        'select': function(obj, record, index){
            var a = record.data.actividad;
            Ext.getCmp('acti').setValue(a);
            var b = record.data.marca;
            Ext.getCmp('tipoveiculo').setValue(b);
            var c = record.data.unidad;
            Ext.getCmp('unidadmedidanivelactividad').setValue(c);
            var d = record.data.tipoportador;
            Ext.getCmp('portadordemanda').setValue(d);
            var e = record.data.estado;
            Ext.getCmp('estadocarrodemada').setValue(e);
            var f = record.data.indice;
            Ext.getCmp('indiceconsumo').setValue(f);

               Ext.Ajax.request({
                url: 'Combustibles/nivelActividadHasta',
                success: function(resp){
                    var r = Ext.decode(resp.responseText)
                    //Ext.getCmp('planoperativo').setValue(r.estructura);
                    Ext.getCmp('nivelactividadacumulado').setValue(r.nivelactividhasta);
                },
                failure: {},

                params: {
                    idcarro: obj.value,
                                    fecha: Ext.getCmp('mesaplanificar').getValue(),

                }
            });
        }
    },
    blankText: 'Este campo es obligatorio',
    allowBlank: false
}, {
```

Figura 2.8 Código de la petición AJAX Nivel Actividad

```
}, {
    x:6,
     xtype: 'combo',
     fieldLabel: 'Aduana',
     id: 'estructurademanda',
     width: 140,
     mode: 'local',
     listWidth: 140,
     readOnly: true,
     displayField: 'nombre',
     valueField: 'id',
     triggerAction: 'all',
     hiddenName: 'aduanademanda',
     store: new Ext.data.Store({
         storeId: 'storeaduana',
         autoLoad: true,
         url: 'Combustibles/estructuraCifra',
         reader: new Ext.data.JsonReader({
             totalProperty: 'count',
             root: 'estructura'
         }, ['id', 'nombre', 'cod']),
         remoteSort: true
     }),
```

Figura 2.9 Código para cargar un combo pasándole un objeto JSON

Si el usuario no introduce los datos correspondientes en la pantalla el sistema muestra un mensaje de error lanzado desde el javascript como se muestra en la **figura 2.10 a)** y **2.10 b).**

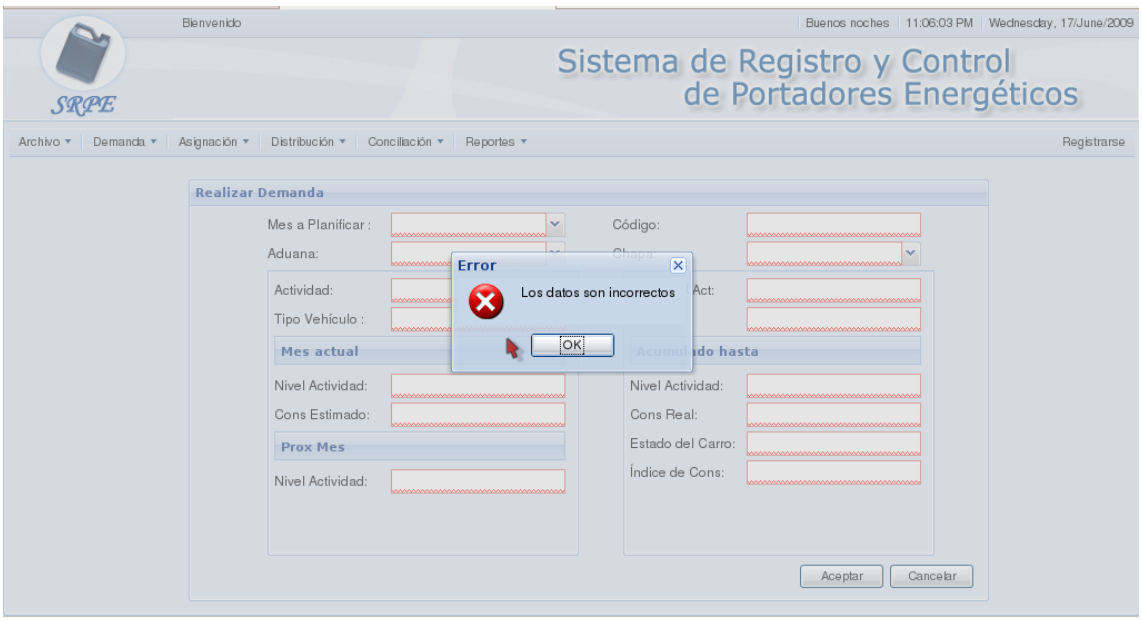


Figura 2.10 a) Pantalla de error que lanza cuando los datos no son correctos

Figura 2.10 b) Pantalla de notificar error en los campos.

Después que el usuario introduce los datos de la pantalla satisfactoriamente como se muestra en la **figura 2.11** y acepta, estos son enviados a la controladora `CombustiblesActions` en el evento submit del formulario RealizarDemanda por el método POST como se ve en la **figura 2.12 a)** donde se muestra la dirección del action que recibirá los datos en la controladora a través del atributo url y **2.12 b)** donde se muestra el evento submit del formulario.

Figura 2.11 Pantalla con los datos correctos

```
Demanda = function(){
    Demanda.superclass.constructor.call(this, {

        region: 'center',
        margins: '3 3 3 0',
        activeTab: 0,
        id: 'formdemanda',

        items: [{
            xtype: 'form',
            layout: 'form',
            height: 350,
            width: 720,
            title: 'Realizar Demanda',
            id: 'formdem',
            url: 'Combustibles/insertarDemanda',
            style: 'margin-left: 161px; margin-top: 20px; ',
            frame: true,

            items: [{
                layout: 'column',

                items: [{
                    layout: 'form',
                    style: 'margin-left: 65px; ',
```

Figura 2.12 a) Código del encabezado del formulario

```
        buttons: [{
            id: 'aceptar',
            text: 'Aceptar',
            handler: function(n){
                if (Ext.getCmp('formdem').form.isValid()) {
                    Ext.getCmp('formdem').form.submit({
                        waitMsg: 'Guardando...',
                        waitTitle: 'Espere por favor',
                    success: function(resp){
                                    Ext.getCmp('formdem').form.reset();
                                },
                                failure: function(res){
                                    Ext.Msg.show({
                        title: 'Error',
                        msg: 'No se pudo insertar el carro',
                        buttons: Ext.Msg.OK,
                        icon: Ext.MessageBox.ERROR
                        });
                                }
                    });
                }
                else {
                    Ext.Msg.show({
                        title: 'Error',
                        msg: 'Los datos son incorrectos',
                        buttons: Ext.Msg.OK,
                        icon: Ext.MessageBox.ERROR
                    });
                }

            }
        }, {
            id: 'cancelar',
            text: 'Cancelar',
                handler: function(){
              Ext.getCmp('formdem').form.reset();
            }
```

Figura 2.12 b) Código de los botones Aceptar y Cancelar

Luego de ser enviado los datos a la clase controladora esta los recibe en el action especificado como se muestra en la **figura 2.13** con el método `getRequestParameter('chapa');` pasándole por parámetro el nombre con que es enviado desde la cliente el atributo que se quiere recibir, aquí la controladora se crea el objeto que se desea insertar en este caso una demanda `$dem= new ECda();` se setean los parámetros y se llama al método `$dem->save();` de la clase `BaseECda` generada por symfony al mapear la base de dato, que a la vez es padre de la clase `ECda` mientas esto ocurre al sistema le muestra al usuario la pantalla que se visualiza en la **figura 2.14**.

```
//Accion que inserta una demanda

  public function  executeInsertarDemanda(){

    $idcarro=$this->getRequestParameter('chapa');
    $mes= $this->getRequestParameter('mesplanicicar');
    $estructura=$this->getRequestParameter('aduanademanda');
    $unidadmedidaactividad= $this->getRequestParameter('unidadmedida');
    $nidelactividad= $this->getRequestParameter('nivelactividadatual');
    $consumoestimado= $this->getRequestParameter('consumoestimado');
    $nivelactividadproximo= $this->getRequestParameter('nivelactividadproximo');
    $indice= $this->getRequestParameter('indice');
    $demanda= $nivelactividadproximo/$indice;
    $ano= date('Y');
    $fecha=date('d-m-Y');

    $dem= new ECda();

    $dem->setIdCda(8);
    $dem->setIdCarro($idcarro);
    $dem->setFecha($fecha);
    $dem->setUnidadMedidaNivelActivdad($unidadmedidaactividad);
    $dem->setNivelActividad($nidelactividad);
    $dem->setConsumoEstimado($consumoestimado);
    $dem->setNivelActividadProximoMes($nivelactividadproximo);
    $dem->setIdEstructura($estructura);
    $dem->setDemanda($demanda);
    $dem->setMes($mes);
    $dem->setAnno($ano);

    $dem->save();


        return $this->renderText("{success:true}");

}
```

Figura 2.13 Código del action Insertar Demanda

Figura 2.14 Pantalla de espera al aceptar los datos de la demanda

2.8 Modelo de Implementación

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, ficheros de código binario, ejecutables y similares. La mayor parte de la arquitectura del sistema es capturada durante el diseño, siendo el propósito fundamental de la implementación el desarrollar la arquitectura y el sistema como un todo. Entre los propósitos que persigue se encuentran, planificar las integraciones del sistema necesarias en cada iteración, distribuir el sistema asignando componentes ejecutables en los nodos del diagrama de integración. Además se implementan las clases encontradas durante el diseño y se prueban individualmente todos los componentes, integrándolos, compilándolos y enlazándolos en uno o más ejecutables. Se realizan los release y las pruebas necesarias al sistema una vez concluida todas sus funcionalidades.

En el flujo de trabajo de diseño se propone crear un plano del modelo de implementación, por lo que sus últimas actividades están vinculadas a la creación del modelo de despliegue. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. Los diagramas de despliegue y componentes, que son artefactos generados en este

flujo de trabajo, conforman lo que se conoce como un modelo de implementación, al describir los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

2.8.1 Diagrama de Despliegue

El Modelo de Despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se construye la funcionalidad entre nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño.

Diagrama de despliegue

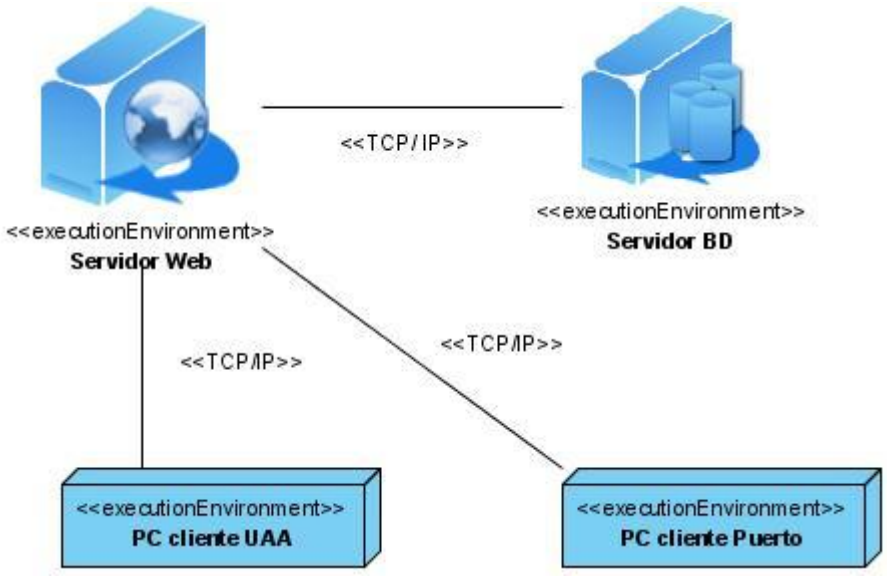


Figura 2.6 Diagrama de despliegue

2.8.2 Diagrama de Componente

Los componentes constituyen la parte modular del sistema, encapsulan implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.).

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. (Ivar Jacobson, 1999) (29).

Para dar solución a la propuesta del sistema se desarrollaron una serie de componentes, los cuales se reflejan en el Diagrama de Componentes que se presenta a continuación:

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases del modelo de diseño; los mismos son creados, modificados o eliminados en el proceso de implementación y constituyen la versión del producto. Un diagrama de componentes se representa como un grafo de elementos, tales como componentes y subsistemas unidos por medio de relaciones de dependencia. Cada diagrama describe un apartado del sistema. Uno de los usos principales es que puede servir para ver que componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

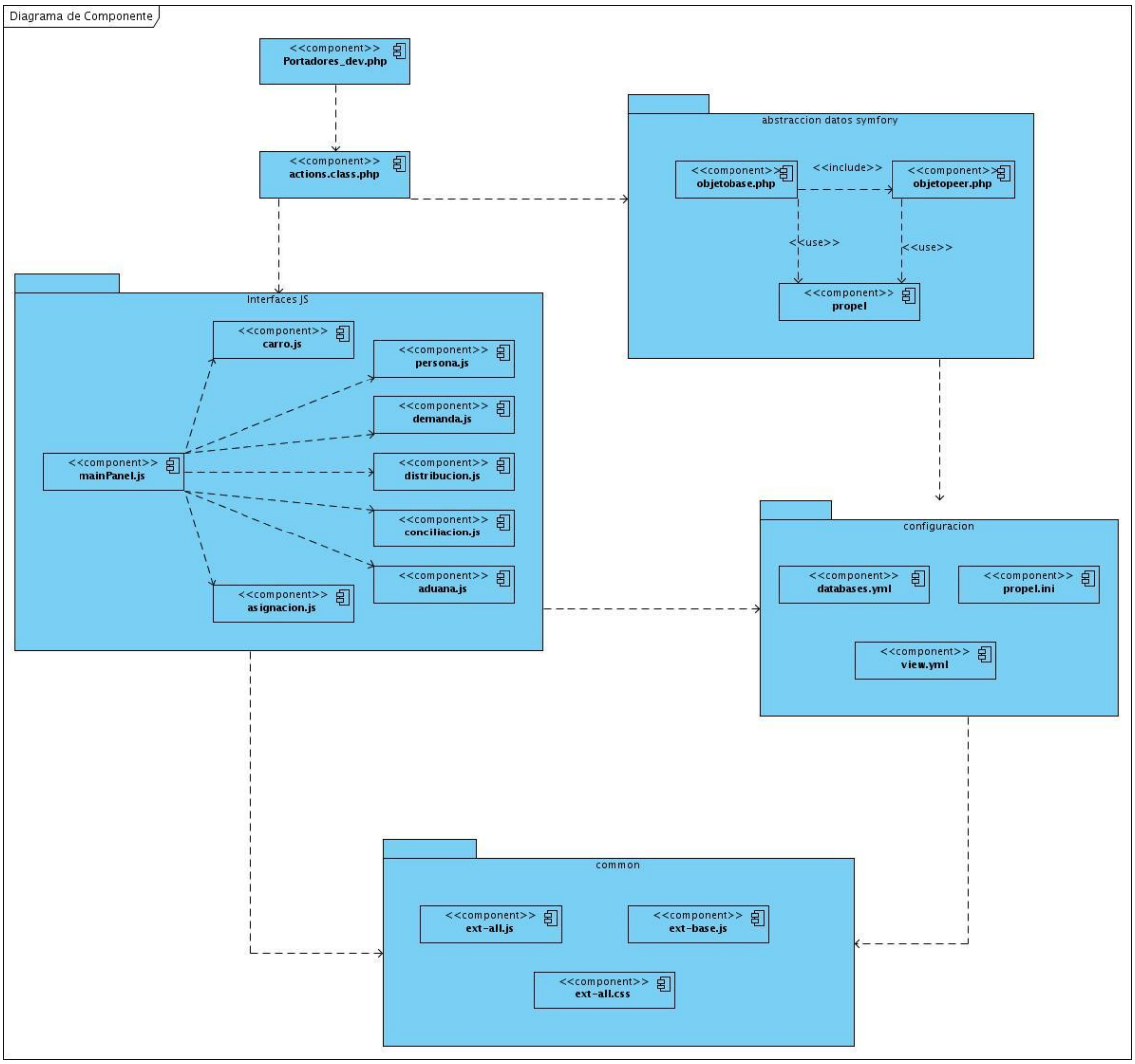


Figura 2.7 Diagrama de Componente

2.9 Conclusiones Parciales

En este capítulo se presentaron los diagramas más significativos para la correcta comprensión de este trabajo, exponiéndose los de clases del diseño, interacción particularmente los de secuencia, así como el diseño de la base de dato, incluyendo los de despliegue y componente, además de que se describe la comunicación entre las capas con un ejemplo práctico del sistema.

Conclusiones Generales

Al culminar el estudio sobre el control de portadores energéticos que se realiza actualmente en la Aduana General de la República de Cuba, con la visión clara y segura de mejorar a gran escala la gestión de dicho control y valorando las tareas y objetivos propuestos, se llegó a las siguientes conclusiones:

- Se realizó el diseño de la aplicación a partir de los requerimientos identificados
- Se desarrolló el Sistema de Registro y Control de Portadores Energéticos usando las herramientas definidas en el polo

Recomendaciones

Se recomienda que la aplicación pase por el equipo de calidad del polo sistemas tributarios y aduanales, que se continúen desarrollando los módulos de control de gas licuado y energía eléctrica, que se culmine el desarrollo del modulo Combustibles y que se exploten las facilidades que brinda el sistema.

Bibliografía

1. **MAYNEGRA.** PUESTOS CLAVES EN LA EFICIENCIA ENERGÉTICA Y EL AHORRO. 2007.
2. **Zamora, L.** [En línea] 2006. http://www.capraispana.com/curiosidades/biogas/consideraciones.htm.
3. **Desarrollo de Software InteraSystem.** Interasystem. [En línea] 2008. http://www.interasystem.com/.
4. **Grandi y asociados.** Grandi y asociados. [En línea] Grandi y asociados, 2009. http://www.grandiyasociados.com/software.asp?idSoftware=4#informacion .
5. **Gextion.** Gextion 2002. [En línea] 2002. http://www.gestion2002.com.
6. Winpyme. [En línea] 2002. http://winpyme.com.
7. **Mecalux.** logismarket. [En línea] 2000-2009. http://www.logismarket.es./cea-ordenadores/software-de-gestion-de-mantenimiento-de-flota-de-vehiculos/1101555875-1177393-p.html.
8. **Cobo Rodriguez, Jose Antonio.** Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas. 2008.
9. **Reyero, Eusebio.** Will Web For Food [WWFF. [En línea] 15 de January de 2009. http://wwff.thespacer.net/blog/metodologias-de-diseno/.
10. *Conferencia de Ingeniería de Software. Patrones de diseño.* **UCI.** 2005-2006.
11. **Larman.** *"UML y patrones".*
12. —. *"UML y patrones".*
13. —. *"UML y patrones".*
14. —. *"UML y patrones".*
15. —. *"UML y patrones".*
16. —. *"UML y patrones".*
17. —. *"UML y patrones".*
18. **Larman, Craig.** *UML Y PATRONES: INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJ.* 2009.
19. Mastermagazine. [En línea] http://www.mastermagazine.info.
20. **Sanchez.** TEMAS DE INTERES GERENCIALES INFORMATICOS. [En línea] 2007. http://felixruben2384.blogspot.com/2007/08/temas-de-interes-gerenciales.html..
21. **Coloma.** Bikermon-Informatico. [En línea] 2007. http://bikermon-informatico.blogspot.com/2007/11/visual-paradigm-la-herramienta.html..
22. **Gallego Vazquez, Jose Antonio.** *Desarrollo de web con PHP y MySQL* . 2003.
23. **Cobo Rodriguez, Jose Antonio.** Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas. 2008.
24. **Johnson y Foote.** *Designing Reusable Classes.* 1988.
25. **Potencier, Fabien y Zaninotto, François.** *Symfony la guia definitiva.* 2008.
26. **Cobo rodriguez, Jose Antonio.** *Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas.* 2007.
27. **Costilla, Carmen.** Características Objeto-Relacionales del Sistema de Gestión de Bases de Datos Oracle. [En línea] 2002. http://sinbad.dit.upm.es/docencia/grado/curso0607/BDOR%20Primera%20Parte%20documento%20Oracle%20en%202002%E2%80%93Nov%202006.pdf.
28. **Conallen, Jim.** *Modelling Web Applications Architectures with UML.* s.l. : ACM New York, 1999.
29. **Jacobson, Ivar, Booch, Gardy y Rumbaugh, James.** *El proceso unificado de*

desarrollo del software. 1999.

Glosario de Términos

Aduana: Oficina pública, establecida generalmente en las costas y fronteras, para registrar, en el tráfico internacional, los géneros y mercaderías importadas o exportadas, y cobrar los derechos que adeudan.

SUA: Sistema Único de Aduanas.

RUP: Proceso Unificado Racional (***R****ational* ***U****nified* ***P****rocess* en inglés) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

UML: Unified **M**odeling **L**anguage. Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

Framework: Un framework, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Software libre: El software libre es la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

XML: Siglas en inglés de *E****x****tensible* ***M****arkup* ***L****anguage* (lenguaje de marcas extensibles), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C)

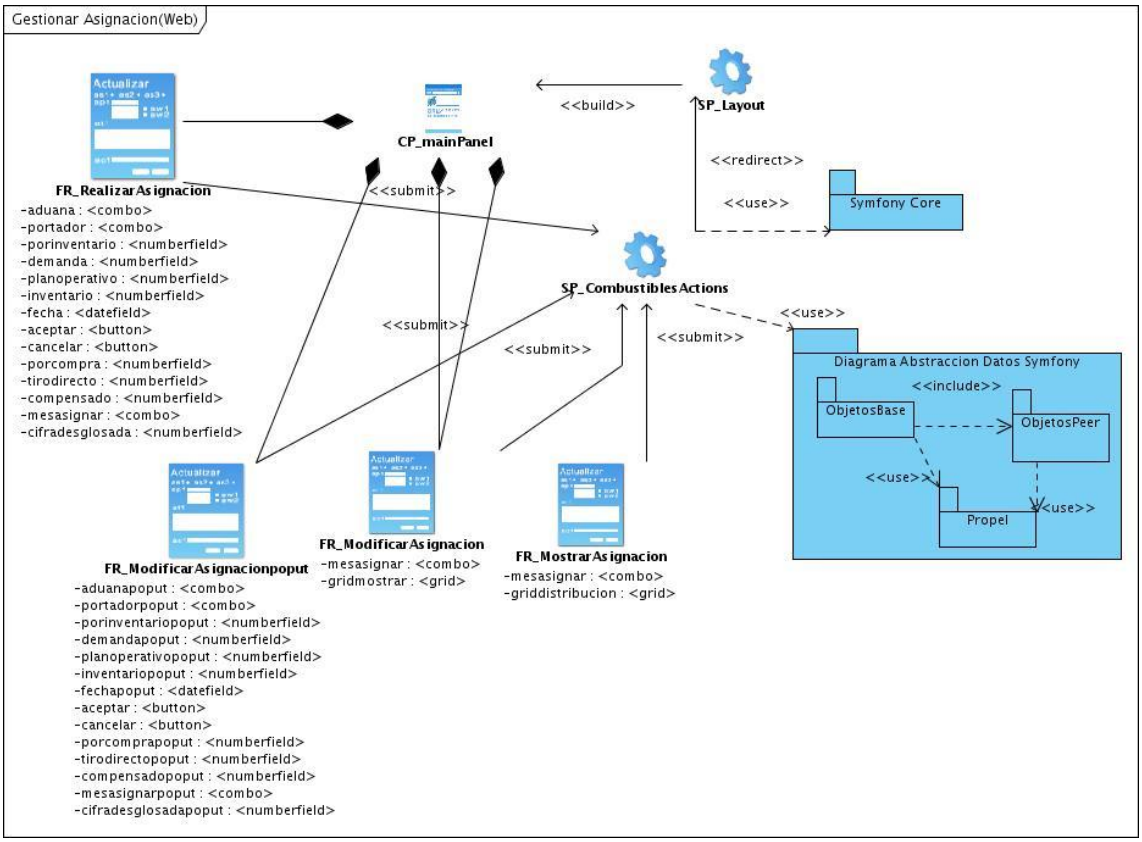
Ajax: Acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (javascript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o **RIA** (Rich Internet Applications)

JSON: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

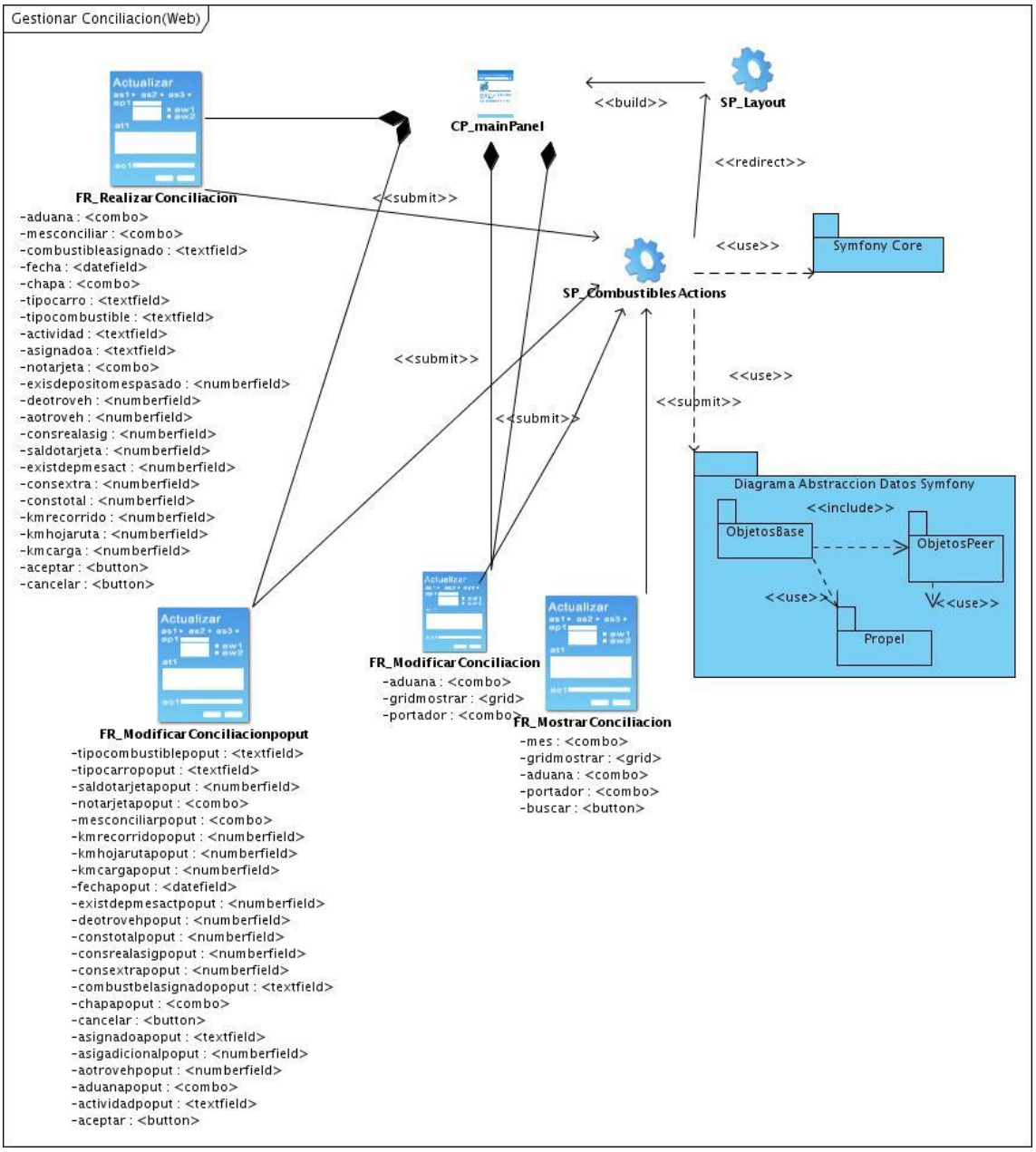
URL: Acrónimo de ***U****niform* ***R****esource* ***L****ocator*, es decir, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos.

SRPE: Sistema de Registro y Control de Portadores Energéticos

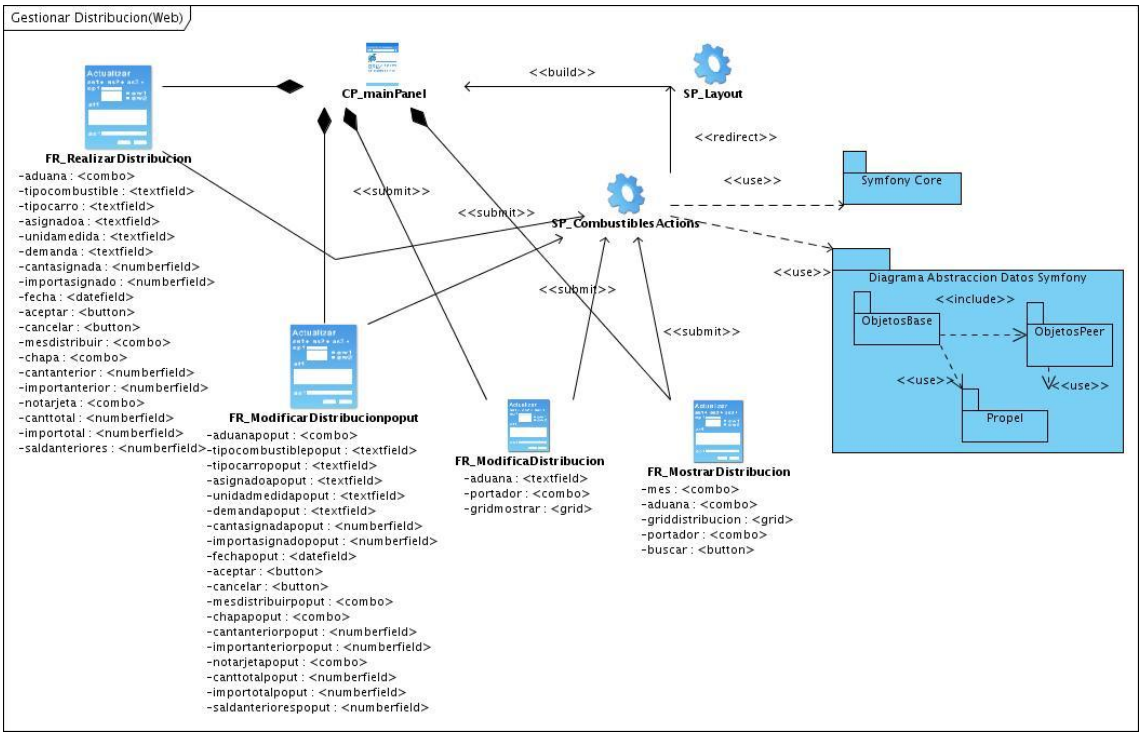
Anexo I Diagramas de clases del diseño



Gestionar Asignación

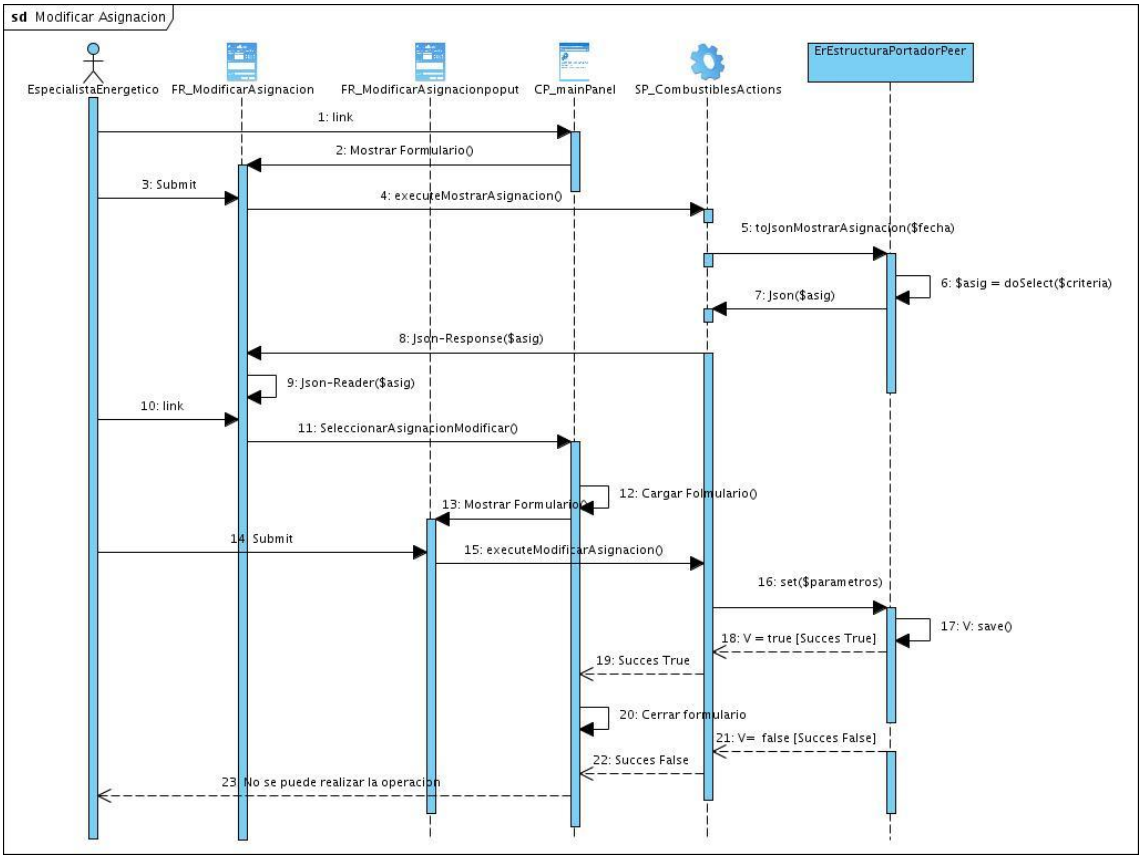


Gestionar Conciliación

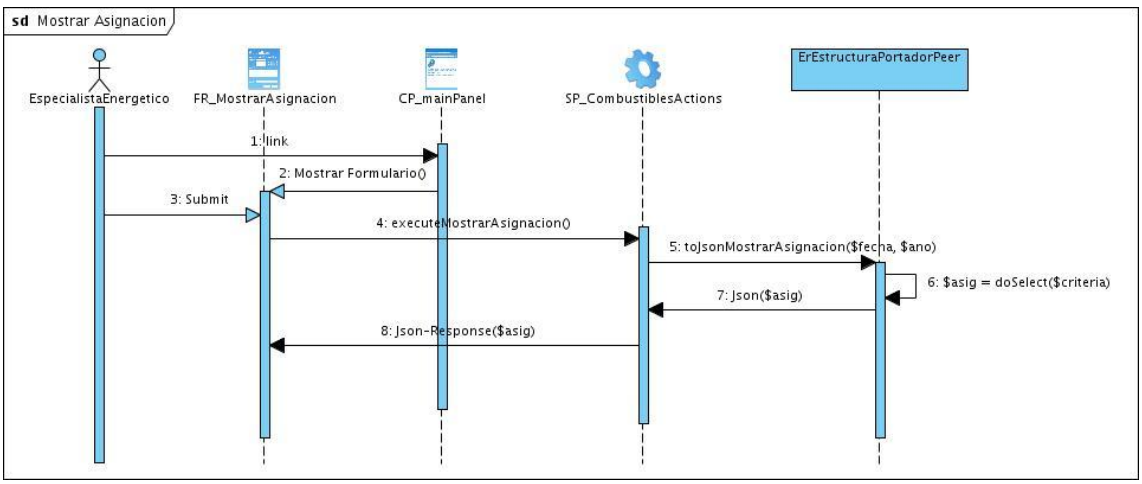


Gestionar Distribución

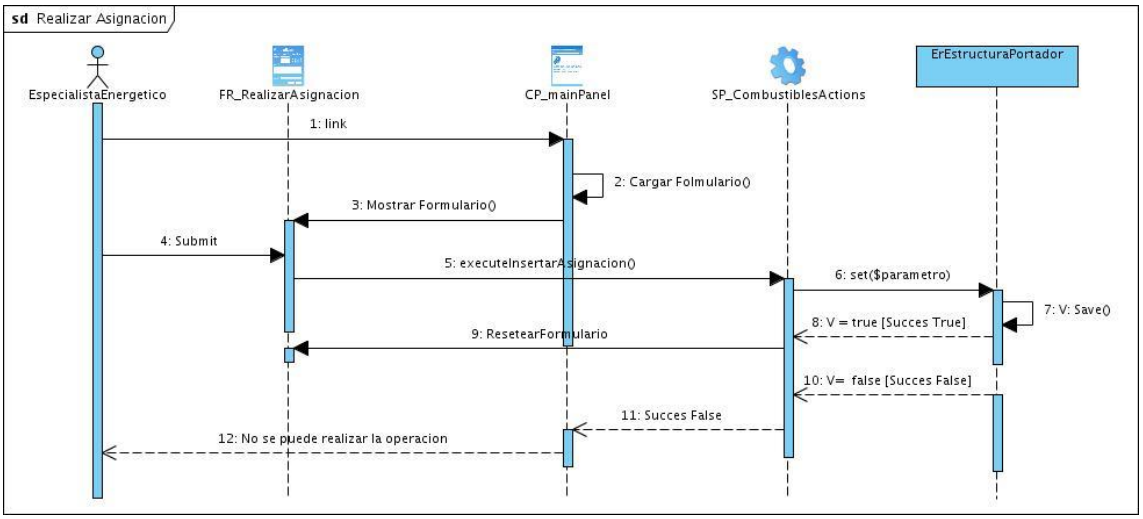
Anexo II Diagramas de secuencias.



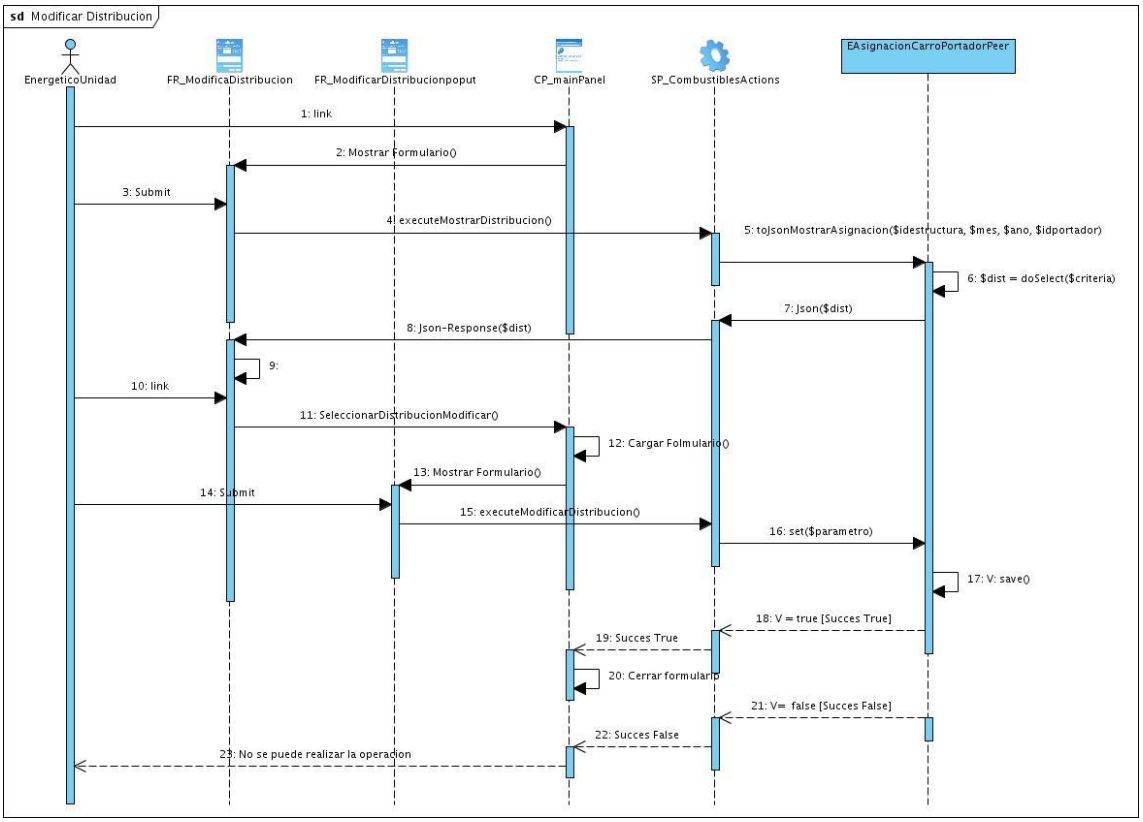
Modificar Asignación



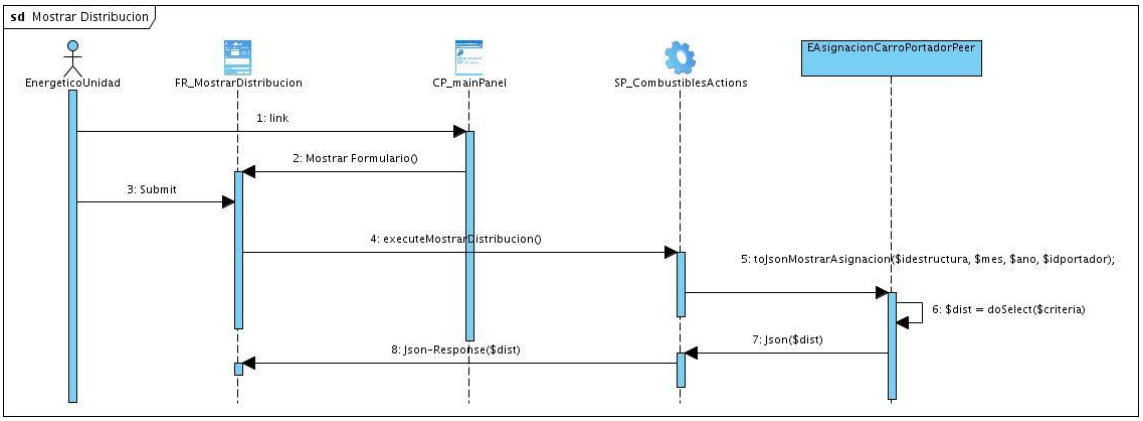
Mostrar Asignación



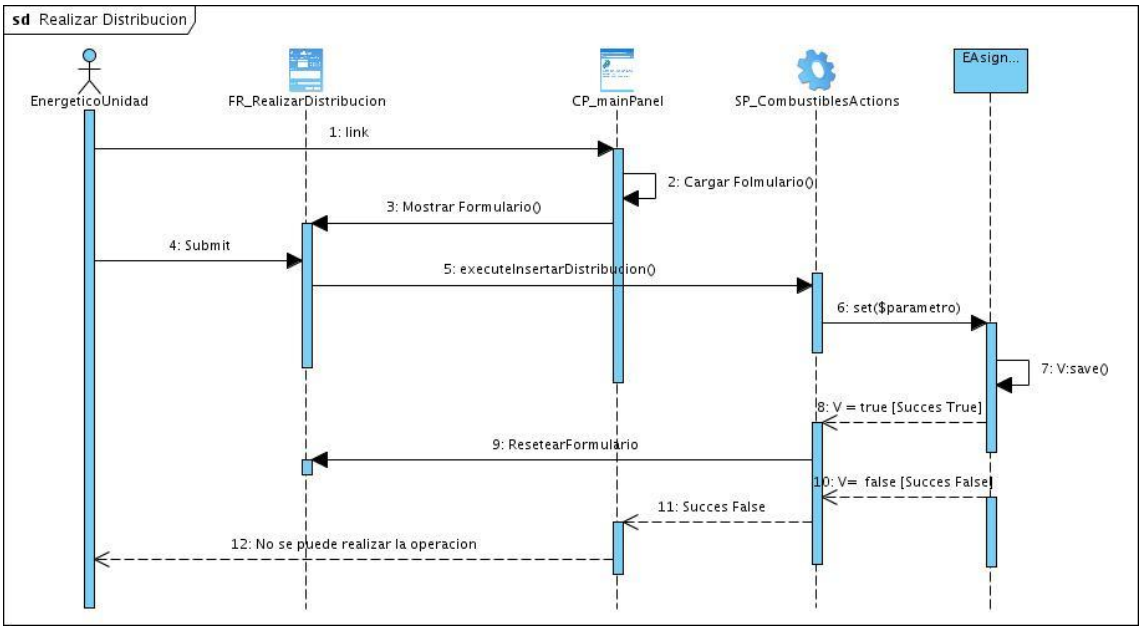
Realizar Asignación



Modificar Distribución



Mostrar Distribución



Realizar Distribución

Anexo III Descripción de las tablas de la base de datos.

Nombre: rh_estructura_cifra		
Descripción: Aquí se almacena las cifras de las estructuras aduanales de todo el país		
Atributo	**Tipo**	**Descripción**
id_estructura	integer	Llave primaria
cifra_regular	integer	Cantidad de gasolina regular que se define para una estructura en un año
cifra_especial	integer	Cantidad de gasolina especial que se define para una estructura en un año
cifra_diesel	integer	Cantidad de diesel que se define para una estructura en un año
codigo	varchar	Código de la estructura
inventario_regular	integer	Cantidad de gasolina regular que tiene en el inventario la estructura
inventario_regular	integer	Cantidad de gasolina especial que tiene en el inventario la estructura
inventario_regular	integer	Cantidad de diesel que tiene en el inventario la estructura

Tabla rh_estructura_cifra

Nombre: e_persona		
Descripción: Aquí se almacena los datos característicos de todas las personas que interactúan con el sistema o que poseen carros con asignación de combustible		
Atributo	**Tipo**	**Descripción**
id_persona	sequence	Llave primaria
id_estructura	integer	Llave foránea(identificador de la estructura a la que pertenece la persona)
primer_nombre	varchar	Primer nombre de la persona
segundo_nombre	varchar	Segundo nombre de la persona
primer_apellido	varchar	Primer apellido de la persona
segundo_apellido	varchar	Segundo apellido de la persona
usuario	varchar	Usuario para logearse
contrasenna	varchar	Contraseña para logearse
carnet_identidad	integer	Carnet de identidad
tratamiento	varchar	Tratamiento que se le da a una persona(Sr, Sra)
desempenno	varchar	Puesto en el que se desempeña la persona

Tabla e_persona

Nombre: e_carro		
Descripción: Aquí se almacena los datos característicos de todos los carros que existen en cada uno de las estructuras aduanales.		
Atributo	**Tipo**	**Descripción**
id_carro	sequence	Llave primaria
id_persona	integer	Llave foránea(identificador de la persona que es dueña del carro)
chapa	varchar	Matricula del carro
marca	varchar	Modelo del carro
actividad	varchar	Actividad a la que está destinado
estado_carro	varchar	Estado en que se encuentra el carro
tarjeta_1	integer	Tarjeta de combustible
tarjeta_2	integer	Tarjeta de combustible
id_estructura	integer	Llave foránea(identificador de la estructura a la que pertenece el carro)
id_portador	integer	Llave foránea (identificador del portador que consume el carro)

Tabla e_carro

Nombre: rh_estructura		
Descripción: Aquí se almacena los datos característicos de las estructuras aduanales de todo el país		
Atributo	**Tipo**	**Descripción**
id_estructura	sequence	Llave primaria
siglas	varchar	Sigla del punto aduanal
nombre	varchar	Nombre del punto aduanal
padre	integer	Estructura aduanal a la que se subordina
nivel	integer	Peso en el árbol de dependencia
objetivo	integer	Objetivo para el que fue creado
id_contacto	integer	Identificador del contacto al que el pertenece
id_tipo_estructura	integer	Identificador del tipo d estructura a la que pertenece
pre_orden	integer	Posición que le corresponde en la lista de recorrido en preorden de todas las estructuras recorridas

Tabla rh_estructura

Nombre: e_cda		
Descripción: Aquí se almacena los datos característicos de la demanda de combustible de los carros en cada una de las estructuras		
Atributo	**Tipo**	**Descripción**
id_cda	sequence	Llave primaria
id_carro	integer	Llave foránea(identificador del carro al que pertenece la demanda)
fecha	date	Fecha en que se realizo la demanda
unidad_medida_nivel_actividad	varchar	Unidad de medida del nivel de actividad que realiza un carro(km/h)
nivel_actividad	integer	Nivel de actividad que realiza un carro
consumo_estimado	integer	Consumo estimado de combustible de un carro
nivel_actividad_proximo_mes	integer	Estimado del nivel de actividad del próximo mes de un carro
demanda	integer	La cantidad de portador demandado para un carro
mes	integer	Mes en que se realiza la demanda
anno	integer	Año en que se realiza la demanda
id_estructura	integer	Llave foránea(identificador de la estructura a la que pertenece la demanda)

Tabla e_cda

Nombre: e_asignacion_carro_portador		
Descripción: Aquí se almacena los datos de la asignación de un portador a un carro en cada una de las estructuras aduanales		
Atributo	**Tipo**	**Descripción**
id_carro_portador	sequence	Llave primaria
id_carro	integer	Llave foránea(identificador del carro al que se le asigno el portador)
mes	integer	Mes en que se realiza la distribución del portador a un carro
anno	integer	Año en que se realiza la distribución del portador a un carro
tarjeta	integer	Tarjeta a la que se le distribuyo el portador
saldo_anterior	integer	Saldo del portador que le quedo del mes anterior
asignacion	integer	Cantidad de portador que se asigno a un carro
fecha_asignacion	date	Fecha en que fue asignado el portador

Tabla e_asignacion_carro_portador

Nombre: e_estructura_portador		
Descripción: Aquí se almacena los datos característicos de todos los carros que existen en cada uno de las estructuras aduanales.		
Atributo	**Tipo**	**Descripción**
id_estructura_portador	sequence	Llave primaria
id_estructura	integer	Llave foránea(identificador de la estructura a la que se le distribuyó el portador)
id_portador	integer	Llave foránea(identificador del portador que se le distribuye a cada una de las estructuras aduanas)
fecha_distribucion	date	Fecha en la que se distribuyo el portador a la estructura
plan_operativo	integer	Plan de entrega de los portadores
asignado_compra	integer	Cantidad de portador asignado por compra
asigando_inventario	integer	Cantidad de portador asignado por inventario
tiro_directo	integer	Cantidad de portador asignado por tiro directo
compensado	integer	Cantidad de portador que se le asigna a un carro compensado
mes	integer	Mes en que se realiza la asignación de portador a una estructura
anno	integer	Año en que se realiza la asignación de portador a una estructura

Tabla e_estructura_portador

Nombre: e_portador		
Descripción: Aquí se almacena los datos característicos de los portadores		
Atributo	**Tipo**	**Descripción**
id_portador	sequence	Llave primaria
tipo_portador	varchar	Tipo de portador
f_inicio	date	Fecha en que fue insertado
f_fin	date	Fecha en que fue eliminado
codigo	varchar	Código del portador

Tabla e_portador

Nombre: e_conciliacion		
Descripción: Aquí se almacenan los datos de la conciliación del combustible al finalizar cada mes		
Atributo	**Tipo**	**Descripción**
id_conciliacion	sequence	Llave primaria
id_carro	integer	Llave foránea(identificador del carro a la que se le realiza la conciliación)
fecha_conciliacion	date	Fecha en que se realiza la conciliación
asignación_adicional	integer	Cantidad de combustible asignado adicionalmente
saldo_tarjeta	integer	Cantidad de combustible que le queda a la tarjeta
existencia_deposito_mes_actual	integer	Existencia de combustible que queda en el depósito en el mes actual
consumo_extra	integer	Cantidad de portador extra que se le suministro al carro
km_recorido	integer	Cantidad de km recorrido por el carro
plan	float	Cantidad de combustible planificado para el mes
km_carga	integer	Cantidad de km recorrido con carga por un carro
km_hoja_ruta	integer	Cantidad de km que declarados en la hoja de ruta del carro
inventario	integer	Cantidad de portador que pasa al inventario
mes	integer	Mes en que se realiza la conciliación
anno	integer	Año en que se realiza la conciliación

Tabla e_conciliacion

Nombre: e_conciliacion_modificada		
Descripción: Aquí se almacenan los datos de la conciliación modificada del combustible al finalizar cada mes		
Atributo	**Tipo**	**Descripción**
id_conciliacion_modificada	sequence	Llave primaria
id_carro	integer	Identificador del carro al que se le va a modificar la conciliación
fecha_conciliacion	date	Fecha en que se realiza la conciliación modificada
asignación_adicional	integer	Cantidad de combustible asignado adicionalmente
saldo_tarjeta	integer	Cantidad de combustible que le queda a la tarjeta
existencia_deposito_mes_actual	integer	Existencia de combustible que queda en el depósito en el mes actual
consumo_extra	integer	Cantidad de portador extra que se le suministro al carro
km_recorido	integer	Cantidad de km recorrido por el carro
plan	float	Cantidad de combustible planificado para el mes
km_carga	integer	Cantidad de km recorrido con carga por un carro
km_hoja_ruta	integer	Cantidad de km que declarados en la hoja de ruta del carro
id_conciliacion	integer	Llave foránea(identificador de la conciliación que se va a modificar)
mes	integer	Mes en que se realiza la conciliación modificada
anno	integer	Año en que se realiza la conciliación modificada

Tabla e_conciliacion_modificada