UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

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

Título: Diseño e Implementación de la solución de software para Ingresos Comerciales en GINA.

Autor:

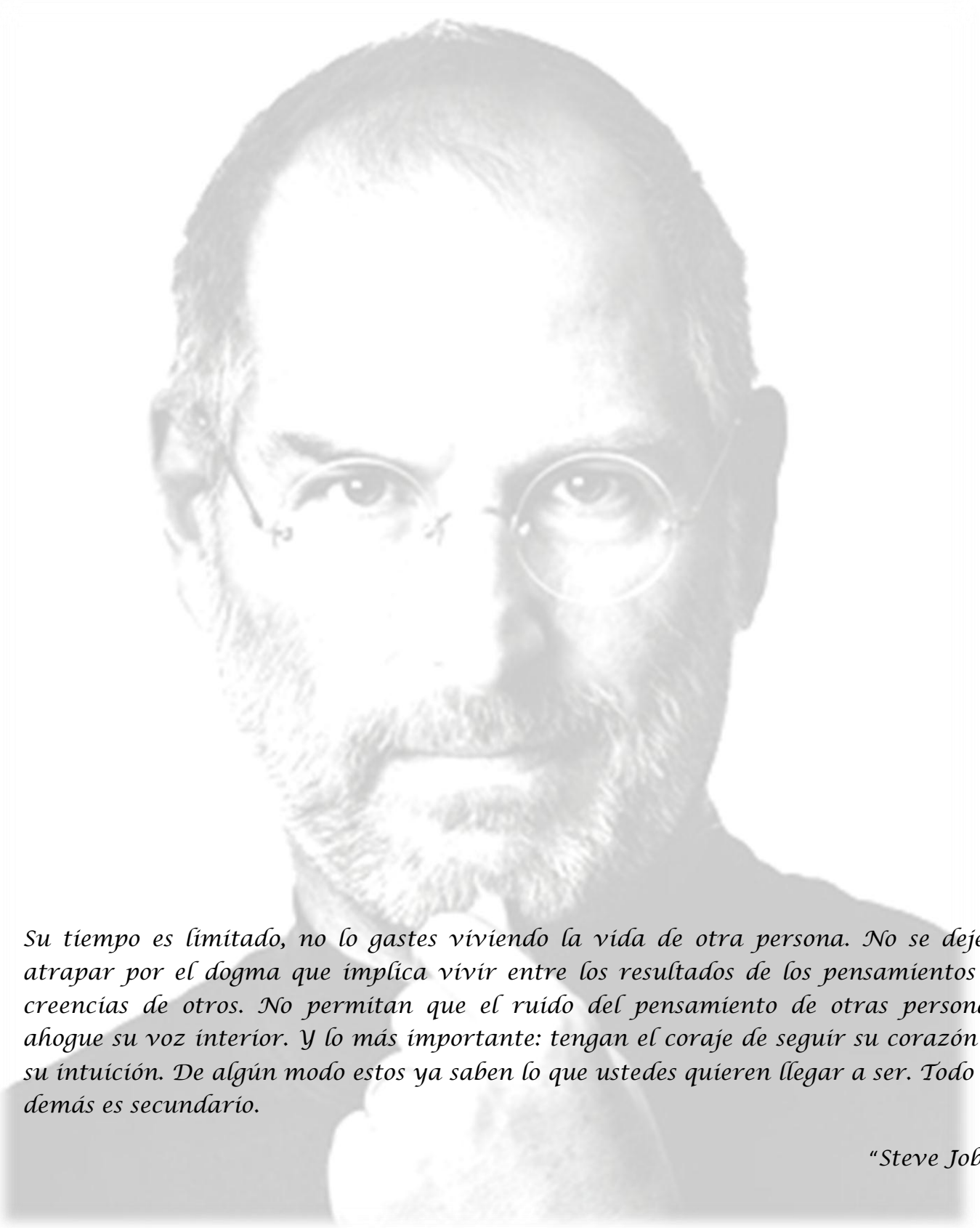
José Daniel Gainza Olivera

Tutor:

Julio César Bravo Rodríguez

Cuidad de la Habana, junio 2012

"Año 54 de la Revolución"



Su tiempo es limitado, no lo gastes viviendo la vida de otra persona. No se dejen atrapar por el dogma que implica vivir entre los resultados de los pensamientos y creencias de otros. No permitan que el ruido del pensamiento de otras personas ahogue su voz interior. Y lo más importante: tengan el coraje de seguir su corazón y su intuición. De algún modo estos ya saben lo que ustedes quieren llegar a ser. Todo lo demás es secundario.

"Steve Jobs"

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 **28** días del mes de **junio** del año **2012.**

_____ _____

Firma del autor

José Daniel Gainza Olivera

Firma del tutor

Julio César Bravo Rodríguez

Dedicatoria y Agradecimientos

Es bueno que existan estos días, es bueno tener la oportunidad de dirigirme a ti. Hoy quiero agradecerte Mami, no porque me diste todo lo que necesitaba, sino porque me enseñaste a valorar lo que tenía. No por llenarme de palabritas bonitas, sino porque me enseñaste a ver el amor en cada sonrisa, en cada mirada, en cada palabra. No por llorar conmigo en mis momentos difíciles, sino por enseñarme que más allá del llanto se necesitan soluciones. No por ocultar mis errores, sino por enseñarme a reconocerlos y enmendarlos. No por resolver mis problemas, sino por enseñarme que la responsabilidad de su solución era solo mía. A ti Mami y a Papi, les dedico éste día, éste momento, éstas palabras, porque gracias a ustedes me he convertido en la persona que soy; han sido, son y siempre serán pilar de mi existir. Una vez mas quiero decirte que "Te amo", no por ser un hoy un día especial, sino porque así lo siento todos los días de mi vida, porque solo contigo aprendí que Mamá se escribe con "A" de Amor. Me enseñaste a aceptar a los demás sin mirarles sus defectos, ahora miro más sus cualidades y me siento feliz por ello. Tú me enseñaste a amar, dialogar, a compartir, me enseñaste a vivir. Gracias por existir…

Quiero agradecer a todos los presentes por estar aquí, por compartir este momento conmigo. Gracias a los que de alguna manera, no importa como, hicieron posible este momento. Les aseguro que incluso una pequeña sonrisa de mucho de ustedes marcó la diferencia en muchas ocasiones, gracias. Gracias por lo buenos momentos, incluso los no tan buenos. A mis compañeros y compañeras de todos estos años, gracias por todo, incluso gracias por nada. Gracias Yarleny por exluir la palabra "no" de tu diccionario tantas veces por mí, por permitirme encontrar en ti mi única amiga. Gracias a todos.

Resumen

A la par del desarrollo del mundo, lo ha hecho también el comercio y la forma de llevarse a cabo. Es objetivo de la mayoría de los países, tratar de simplificar los trámites que se llevan a cabo por las partes implicadas en un intercambio de mercancías, sin dejar de mantener el control y administración. Con el surgimiento de las aduanas, se da cumplimiento a los objetivos anteriormente dichos, debido a que son el órgano responsable dentro de cada país de llevar el control, en materia de comercio. La Aduana General de la República de Cuba (AGR), que se encarga de poner en práctica los procedimientos aduaneros en las diferentes fronteras del país al cual representa. Dentro de los procedimientos que se llevan a cabo en la AGR se encuentra la gestión de Ingresos por la Vía Comercial. Esta área del departamento de Economía es de vital importancia para la AGR, a través de ella se gestionan los procesos referentes a cobros de deudas contraídas por conceptos de Aranceles y Servicios. Estos procesos, presentan soluciones en el Sistema Único de Aduana (SUA), que es actualmente el Sistema que está en pleno funcionamiento en la AGR, sistema que presenta dificultades de rendimiento, mantenimiento y funcionamiento, siendo las causas principales para que los especialistas del Centro de Informatización para la Dirección de la Aduana (CADI),en conjunto con la Universidad de las Ciencias Informáticas (UCI), se centraran en la creación del Sistema de Gestión Integral de Aduana (GINA), específicamente un módulo que gestione los Ingresos por la Vía Comercial, eliminando los problemas mencionados.

Como entrada para la solución del objetivo, se plantearon los requisitos funcionales, que fueron el punto de partida para el posterior diseño desarrollado. Se obtuvieron artefactos (diagramas de secuencia orientados a actividades del negocio, diagrama de clases persistentes, modelo entidad-relación, diagrama de componentes, diagrama de despliegue) que facilitaron el desarrollo de la solución. Se utilizaron diversas herramientas y metodologías, teniendo en cuenta las buenas prácticas utilizadas mundialmente, dentro de las que tenemos Visual Paradigm para el modelado, así como la creación de un entorno Cliente-Servidor por medio de tecnología Web, sobre la base del patrón arquitectónico Modelo Vista Controlador (MVC) y del lenguaje PHP, sin dejar de mencionar el importante uso de los frameworks ExtJs para el trabajo con las interfaces de usuario y Symfony como rector para el trabajo a nivel de servidor. Luego de concluir el desarrollo de la solución se realizaron pruebas para la validación de ésta, brindando resultados satisfactorios.

Contenido

Introducción

El siglo XXI, la era de las Tecnologías de la Información y la Comunicación (TIC). El rápido avance de las tecnologías ha provocado que a nivel mundial se demanden organizaciones inteligentes y sistémicas, “El desafío de la Gerencia se versa en que las Organizaciones deben tener la capacidad de aprender, desaprender y reaprender”[1]. La competitividad de una Nación tiene como fundamento el nivel y calidad de educación de su gente. Las TIC han revolucionado la manera de comunicarse los individuos, de ver las empresas e incluso de hacer negocios. La manera en que hoy en día se entiendan los paradigmas existentes será la base del desarrollo económico y social de cualquier país. Cuando a cambios de paradigmas se refiere Cuba no se encuentra exenta a éstos, hoy en día se comprenden las necesidades de realizar transformaciones en la concepción que se tenía del sector económico y el país se encuentra inmerso en una etapa de informatización de sus empresas; entre ellas la Aduana General de la República de Cuba (creada en febrero de 1963, en lo adelante AGR).

Muchas veces las empresas tienen diferentes aplicaciones de negocio desplegadas durante el transcurso del tiempo en diferentes lenguajes, que usan diferentes tecnologías, se despliegan en diferentes plataformas de hardware y sistemas operativos con interfaces de usuario inconsistentes. El resultado es funcionalidad aislada, múltiples instancias de los mismos datos, actividades manuales redundantes, costos más altos y respuestas ineficientes para sus clientes. Además existe la necesidad creciente de integrar con sus socios de negocio y otras compañías fuera y dentro de las fronteras de su empresa (1). La AGR está hace más de 10 años informatizando la mayoría de sus sectores, áreas y procesos para brindar a sus empleados una vía más cómoda y segura de trabajo, con el objetivo de ofrecer un servicio de mejor eficiencia y calidad, entre ellos los procesos del departamento de Economía. La aplicación informática Ingresos Comerciales la cual está siendo utilizada para gestionar los procesos de este departamento forma parte, en su conjunto, del Sistema Único de Aduana (SUA)[2] desde el año 2008, la cual posee una arquitectura incompatible al Sistema que actualmente se desarrolla para beneficio de la AGR, Gestión Integral de Aduanas (GINA)[3], esto hace al actual sistema incapaz de lograr integrarse con los nuevos sistemas GINA y Ventanilla Única. Entre las características que avivan la imposibilidad de integrar estos sistemas antes mencionados se destaca el hecho de que el sistema Ingresos Comerciales no está implementado con las ventajas que ofrecen los Framework de

1Peter Drucker: “Desafíos de la gerencia en el siglo XXI”
2 Sistema vigente en la Aduana General de la República de Cuba.
3 solución de software cubana para la gestión de los procesos aduanales, desarrollado por la Universidad de las Ciencias Informáticas en trabajo unido a especialistas del Centro de Automatización de la Dirección y la Información de la AGR, CADI

desarrollo, en su lugar está concebido en PHP estructurado y que las consultas a la base de datos sea a través de procedimientos almacenados provocando que el mantenimiento sea engorroso. Con bases en esta necesidad real el trabajo que se presenta tiene como **problema a resolver:** El sistema en explotación en la Aduana General de la República para los procesos de ingreso por la vía comercial no cumple con los requerimientos de arquitectura necesarios para garantizar integración con los sistemas GINA y Ventanilla Única actualmente en desarrollo. Teniendo como **objeto de estudio** los procesos de diseño e implementación de sistemas de software y tomando como **campo de acción** los procesos de diseño e implementación de sistemas de software para GINA. El **objetivo general** del presente trabajo se define como: Desarrollar una solución de software a partir de los requisitos ya identificados que garantice integración y funcionamiento correcto bajo las definiciones arquitectónicas del Departamento de Soluciones para Aduana, desglosado específicamente en:

- Elaborar el marco teórico que incluya la revisión de los elementos descritos en la arquitectura y el diseño de los sistemas que se encuentran en estos momentos en desarrollo en el Departamento de Soluciones para Aduana así como los requisitos descritos por la solución de ingresos.
- Diseñar la nueva solución.
- Implementar la solución modelada.
- Validar la solución mediante pruebas unitarias.

Con el fin de dar cumplimiento al objetivo propuesto, controlar y evaluar el proceso se plantean las siguientes **tareas investigativas:**

- Fundamentación del estudio sobre los diferentes sistemas de Ingresos Comerciales, enfocado en las aplicaciones y limitaciones existentes.
- Estudio sobre otros sistemas para identificar ventajas y tendencias actuales.
- Comprensión del funcionamiento real de los sistemas Ingresos Comerciales, GINA y Ventanilla
- Identificación y descripción del diseño adecuado para la solución deseada.
- Descripción de los diferentes patrones y herramientas de diseño de Software aplicables a la estructura general de la solución a implementar.
- Estudio y utilización de técnicas de optimización algorítmica durante el desarrollo de la solución planteada.
- Diseño y aplicación de casos de prueba con la finalidad de comprobar el funcionamiento del sistema una vez concluido.

El presente documento posee la siguiente **estructuración del contenido**: Está compuesto por tres capítulos que dan respuesta a los objetivos y tareas planteados. Cada uno de ellos contiene los elementos necesarios para llegar en conjunto a la solución final, partiendo el estudio del problema planteado.

Durante el transcurso del **capítulo I** se complementan los aspectos teóricos más relevantes de los sistemas Ingresos Comerciales y GINA, así como el estudio de sus principales características y limitaciones, partiendo además del principio de funcionamiento de los mismos. De esta forma se persigue como objetivo principal el planteamiento de las tecnologías, métodos y herramientas utilizados para lograr la materialización de la solución.

En el **capítulo II** se describe la propuesta de solución, y está enmarca fundamentalmente en la arquitectura definida para los actuales sistemas de gestión para la AGR. Serán detallados también durante el transcurso del mismo, los aspectos fundamentales del diseño. Se abordan temas correspondientes al desarrollo de la solución, como la organización e interacción de los componentes utilizados durante su construcción.

En el **capítulo III** del trabajo, se notificarán los resultados de un conjunto de pruebas unitarias realizadas durante las diferentes fases de construcción, tanto en entornos controlados como de despliegue piloto.

1. Capítulo I

1.1. Introducción

En el presente capítulo se detallan las características arquitectónicas por lo cual es necesario la realización de un nuevo sistema para la gestión de Ingresos por la Vía Comercial para la AGR a partir del nuevo diseño; partiendo del estudio que se realiza en este capítulo. Se plasmará los resultados de la investigación ejecutada sobre las tecnologías y herramientas a utilizar para el desarrollo de la solución, incluyendo diferentes patrones y metodologías de diseño.

1.2. Aduana cubana

La Aduana es el organismo responsable de la aplicación de la Legislación Aduanera y del control de la recaudación de los derechos de Aduana y demás tributos; encargados de aplicar en lo que concierne la legislación sobre comercio exterior, generar las estadísticas que ese tráfico produce y ejercer las demás funciones que las leyes le encomienda. (2)

En el contexto mundial las aduanas son entidades, muy antiguas, que fueron creadas para el control del intercambio comercial con el exterior, las naciones modernas desarrollan sus actividades a través de las aduanas, en el sentido de regular la entrada y salida de mercancías, para alcanzar principalmente sus objetivos económicos, es decir, son como las llaves que abren o cierran el comercio exterior; en consecuencia es innegable la gran importancia que ellas tienen en la actividad económica de cada país. (2)

No es función de la aduana ni de la administración que gira alrededor de ella, el establecer trabas al Comercio Internacional, más bien todo lo contrario. En las diferentes fases que abarca, las operaciones, como embarque de productos, documentación, liquidación de derechos e impuestos, valoración, clasificación, entre otros, su objetivo es facilitar el comercio, hacerlo más práctico, expeditivo y funcional. (2)

En Cuba, la Aduana constituye un órgano de control en frontera y en la actividad interna vinculada al comercio exterior, que garantiza la seguridad y protección de la sociedad socialista y de la economía nacional, así como la recaudación fiscal y las estadísticas del comercio exterior, a través del

cumplimiento de las políticas estatales de competencia aduanera para el tráfico internacional de viajeros, mercancías y medios de transporte. (3)

1.2.1. Otros Conceptos

Recargo por mora: recargo que se le aplica al contribuyente por realizar el pago de la deuda tributaria fuera de los términos establecidos en la legislación vigente. (3)

Operación: modalidad que se emplea para ejecutar un trámite. (3)

Nota de Crédito: documento en el cual el comerciante envía a su cliente, con el objeto de comunicar la acreditación en su cuenta de una determinada cantidad, por el motivo expresado en la misma. Este tipo de comprobante se emite para modificar las condiciones de venta originalmente pactadas (anular operaciones, devoluciones, descuentos, bonificaciones, subsanar errores). Debe contener los mismos requisitos y característica de los comprobantes de venta que dan derecho a crédito tributario. (3)

Instrumento de pago: documento, forma o medio de pago que se emplea por las personas naturales o jurídicas, para el pago de obligaciones fiscales o como resultado de los compromisos de pagos contraídos en las relaciones monetarias mercantiles entre personas jurídicas o por la prestación de un servicio o venta. (3)

Factura: documento económico o contable mediante el cual se determina los importes a cobrar por un servicio o venta realizada. (3)

Documento: formulario de obligatorio cumplimiento con la información o datos necesarios para la formalización o trámite de determinadas operaciones, ventas o servicios, que se prestan en la gestión u objeto social de la entidad. (3)

Declaración de mercancía: manifestación en la forma prescrita por la Aduana, por la que los interesados indican el régimen aduanero que se ha de aplicar a las mercancías y proporcionan los datos que la Aduana exige para la aplicación de este régimen. (3)

Crédito Comercial: facilidad de pago por un término mayor de 30 días concedido por la parte que presta un servicio al cliente, el que se calcula mediante un porciento según la moneda en que pague el cliente. (3)

Derecho de tonelaje: Impuesto no tributario que se cobra mediante tasa que se calcula por el tonelaje neto de los buques que arriben a puerto. (3)

Concepto: tipo de tributo (impuesto o tasa) a cobrar por la deuda fijada como resultado de una operación comercial, no comercial o de servicios prestados. (3)

Apremio: cobro forzoso que se le aplica al contribuyente cuando de forma voluntaria no se realiza la contribución del pago de la deuda al presupuesto. (3)

Alcance: determinación por la Aduana de la diferencia entre el importe pagado por concepto de derechos de aduana y lo que realmente se debió percibir. Es la consecuencia del Reparo cuando éste se declara procedente. (3)

Acuerdo de Aplazamiento: pago aplazado de la deuda tributaria con independencia de la forma en que haya sido determinada esta y el período voluntario o forzoso en que se encuentre el deudor para efectuar dicho pago. Esta puede ser con o sin fraccionamiento. (3)

1.3. Sistemas Estudiados

Los sistemas extranjeros estudiados son considerados muy eficientes cuando de manejar información a nivel de bases de datos se trata, lo cual aportará elementos significativos que pueden servir para lograr la optimización de los datos y la eficiencia de los procedimientos que serán necesario implementar.

1.3.1. Data Integration Suite (Suite para la Integración de Datos)

Data Integration Suite brinda a las organizaciones un método más inteligente y perfeccionado para la distribución de datos diversos en toda la empresa, lo cual permite una toma de decisiones más rápida e informada y reduce la complejidad y los costes operativos. Ofrece un conjunto integral y modular de tecnologías, con herramientas avanzadas e integradas de modelado, desarrollo y administración. Esta arquitectura ayuda a acelerar el flujo de datos permanentes para abordar los desafíos más complejos de integración que enfrenta una organización, ya sean relacionados con la actividad comercial, o la administración basada en la relación con los clientes (CRM) o centrados en la tecnología, como el

desarrollo de aplicaciones orientadas a los servicios (4). Lo que posibilita el uso de estas tecnologías de componentes es una capa integrada de modelado y metadatos. Entre las **ventajas** más significativas podemos señalar que:

- **Acelera el diseño, desarrollo y entrega de flujos continuos de datos**:

Herramientas comunes de modelamiento, gestión de metadatos, desarrollo, administración y otros servicios, permiten que los arquitectos de datos modelen rápidamente flujos de datos, de principio a fin, involucrando múltiples fuentes, entendiendo el lineamiento de la información y el impacto de los cambios, y llevando a cabo los cambios con mayor control y agilidad.

- **Extenso y flexible:**

Los clientes pueden elegir la técnica o combinación de técnicas más conveniente de integración de datos, replicación, federación o integración basada en eventos, para construir flujos flexibles de datos. Ya que Data Integration Suite es modular, usted puede comenzar con los proyectos actuales y escalar para responder a los nuevos retos de integración de datos, sin importar que tan complejos o grandes sean.

Los componentes de Data Integration Suite trabajan en conjunto para proporcionar una visualización integrada del cliente. (4)

Componentes de la Tecnología de Integración.

- **Replication:**

Replication es líder en replicación de datos heterogéneos y cuenta con el apoyo de los arquitectos de datos por su potencial de sincronización de datos bidireccionales y en tiempo real, lo cual garantiza la disponibilidad fiable de datos en toda la empresa. Entre los diversos entornos y soluciones para la distribución de datos, Replication es compatible con la obtención y captura inteligente de datos y proporciona funciones de integración de datos en tiempo real. (4)

- **Real-Time Events:**

Esta tecnología transfiere datos para los que el tiempo es crucial, desde bases de datos de empresa heterogéneas a arquitecturas de mensajería, por lo que se eliminan los “retrasos de información” creados por la actualización por lotes o por procesos de recopilación discontinuos. Real-Time Events conecta también cualquier fuente de datos directamente a cualquier aplicación mediante la creación de flujos de procesos automatizados y dinámicos que vinculan múltiples fuentes de datos y aplicaciones. (4)

Modelado, Metadatos y Desarrollo.

- **PowerDesigner:**

A medida que los entornos de datos empresariales se tornan más variados y complejos, los especialistas en administración de datos recurren a herramientas de modelado para tomar el control. Para ello, Sybase adoptó un método basado en modelos para la integración de datos, y creó Data Integration Suite a partir de una base sólida de funciones de administración de metadatos y modelado de datos. Sybase PowerDesigner, la solución de modelado líder del mercado, incluye dichas funciones en el paquete. Al utilizar PowerDesigner, los arquitectos de datos y las empresas pueden generar y capturar los metadatos necesarios para describir su entorno de datos futuro: requisitos, modelos de datos, flujos de datos, modelos de procesos, y mucho más. Mediante el uso de estos metadatos, Data Integration Suite ofrece un conjunto de herramientas para diseñar arquitecturas de integración de datos y luego implementarlas utilizando la tecnología de componentes que resulte más adecuada. Además, los arquitectos pueden aplicar ingeniería inversa a los flujos de datos existentes y así modelar e implementar cambios en dichos flujos, según lo requieran los nuevos procesos de negocios. Estas funciones permiten a la empresa anticipar los cambios con mayor proactividad y hacer ajustes rápidos de forma central, de ser necesario. (4)

- **WorkSpace:**

WorkSpace representa el futuro de las herramientas y brinda el entorno de desarrollo integrado (IDE) para Data Integration Suite. Los desarrolladores y arquitectos encargados de la integración de datos necesitan un marco de trabajo más flexible capaz de abarcar una serie de tareas de desarrollo, mantenerse a la par de los últimos estándares y ocultar la complejidad de tecnologías subyacentes a la infraestructura. WorkSpace tiene como base el marco de código abierto de Eclipse. Se trata de un grupo de herramientas Java que forma parte del paquete muy importantes para el desarrollo de bases de datos y aplicaciones Web, arquitecturas orientadas a los servicios y aplicaciones móviles. Los arquitectos, al utilizar los modelos de datos creados en PowerDesigner, pueden recurrir a WorkSpace para desarrollar e implementar las tecnologías de Data Integration Suite que resuelvan con mayor eficiencia los desafíos de la integración de datos. (4)

Administración de Servicios de Datos.

- **Data Services Administrator:**

Con Data Services Administrator es posible administrar múltiples servicios de datos en todos los recursos de datos de la empresa a través de una consola centralizada. El almacenamiento en caché de datos, la seguridad y la instrumentación de servicios se pueden administrar en las diferentes tecnologías de integración. También es posible administrar las funciones de creación de perfiles y limpieza de datos que ofrecen las soluciones de terceros.

1.3.2. Enterprise Connect Data Access (ECDA)

ECDA es la solución de datos empresariales que permite tener acceso directo y transparente a bases de datos heterogéneas. Esto permite que las aplicaciones puedan interactuar con bases de datos, para acceder a la información, consolidarla y/o actualizarla. Enterprise Connect Data Access proporciona los cimientos esenciales para lograr la conectividad entre clientes LAN y de Internet y fuentes de datos de empresas.

ECDA para Oracle:

Esta opción proporciona Adaptive Server Enterprise (ASE) [4] y los usuarios de replicación del servidor el componente de Conexión Directa necesaria para la conectividad con bases de datos Oracle. Incluye conectividad de Oracle nativa y tiene todo lo necesario para el acceso de Oracle. (4)

ECDA para Oracle funciona con el servidor ASE o de replicación para proporcionar un acceso dinámico a datos de Oracle. También incluye la opción de ECDA de DTM (Administrador de transacciones distribuidas) que, cuando se utiliza con la ASE, ofrece en dos fases soporte para Oracle y las transacciones a través de ASE. Además, la interfaz de Sybase Central gráfica de usuario basada, Gerente de DirectConnect, ayuda a configurar, administrar, solucionar problemas y controlar su entorno de acceso a datos de Oracle.

Para organizaciones que necesitan acceder a bases de datos heterogéneas que no son de Oracle, la opción de ECDA para ODBC se puede usar con el producto base para unirse a las bases de datos de forma transparente. Se utiliza Servidor de Replicación, puede proporcionar bi-direccional, casi en tiempo real el movimiento de datos a través de bases de datos heterogéneas aún siendo estas diferentes de Oracle.

Mediante ECDA, las aplicaciones conectadas pueden enviar una única solicitud y combinar datos de varias fuentes como si todos los datos estuvieran almacenados en una única base de datos. Sus beneficios se plasman a continuación. (4)

- Uniones heterogéneas optimizadas.
- Plena compatibilidad transaccional: lectura y escritura.
- Seguridad de alto nivel, que incluye SSL y control de acceso a nivel de fila.
- Plena compatibilidad con XML y Java.

[4] Motor de bases de datos insignia de la compañía Sybase.

Por tanto esta integración hecha con un propósito específico puede cumplir con los requisitos existentes. Sin embargo, a medida que las empresas cambian, las organizaciones flexibilizan sus herramientas creadas con un fin específico para darles un uso más amplio, lo cual genera resultados poco satisfactorios. Generaciones de herramientas disímiles confluyen en una infraestructura compleja y frágil. La falta de agilidad y los costes elevados llevan a las empresas a un punto crítico. Se ven imposibilitadas de cambiar la infraestructura de datos lo suficientemente rápido como para mantenerse a la par de las actividades del sector.

1.3.3. SIDUNEA

Es un sistema de gestión de las operaciones aduaneras automatizado que cubre la mayor parte de trámites de comercio exterior. El sistema propone una solución integrada para el proceso de manifiestos, declaraciones de aduana, procedimientos de contabilidad, tránsito y regímenes suspensivos. SIDUNEA genera datos comerciales que pueden ser utilizados para análisis estadísticos y económicos. Tiene en cuenta los códigos y normas internacionales desarrollados por la ISO (Organización Internacional de Normalización), OMD (Organización Mundial de Aduanas) y Naciones Unidas. (3)

Características tecnológicas:

Gracias a la tecnología 100% Java, SIDUNEA permite el uso de tarjetas Inteligentes con procesadores y tecnología Java para controlar los accesos al sistema y los pagos electrónicos.

Además cuenta con:

- Modernos conceptos de seguridad (PKI)
- Conceptos DOM (Document Object Model), XML.
- Directorios de mensajes XML estándar que hacen posible la cooperación internacional entre sistemas y la creación de la red Customs Global
- Protocolo REWI extendido, TCP/IP
- Interfaces usuario amigable (WYSIWYG)
- Extensión e implementación dinámica

- Adaptabilidad (según el número de operaciones)
- Aspectos de seguridad ya integrados
- Funciones especiales como múltiples idiomas, Gestión, Propiedad de documentos y Auditoría. (5)

Ventajas:

El estudio del sistema SIDUNEA permitirá trabajar con más mayor comodidad para elaborar recaudos cuando más convenga, dentro de los tiempos permitidos, con o sí conexión a la red; así como la posibilidad de revisar los datos ingresados tantas veces como usted lo necesite, sin pérdida de tiempo ni recursos, todo desde un computador. Ya que estas son características muy favorables que brinda SIDUNEA. Además cuenta con un módulo para la gestión de ingresos. Este es el Módulo de Aduanas y trabaja principalmente con la Declaración de Mercancías de Exportación (DME), su ingreso al sistema, su verificación local y remota, registro, aplicación del resultado de selectividad y validación. Adicionalmente, contiene opciones de reporte para verificar el estado de bienes declarados bajo regímenes suspensivos, tales como el Depósito de Aduanas. El Módulo para Aduana se describe como el módulo central del sistema SIDUNEA, es la base de los procedimientos de control, del cobro y liquidación de los impuestos, es el sistema para el procesamiento de la declaración y su estudio será un gran será una gran base para nuestro futuro Sistema.

1.3.4. Sistema SOFIA

Es un sistema, de despacho aduanero informatizado que interactúa en forma directa con sus usuarios: Despachantes de Aduana, Empresas de Transporte, Depositarios, Funcionarios de Aduana y con los Organismos vinculados al comercio exterior.

El Sistema de información se trata de la construcción de un conjunto operacional que toma como base y punto de partida el Sistema SOFI (Sistema de Computación para el Flete Internacional) francés, que va incorporando nuevas funcionalidades para satisfacer las necesidades de gestión de la Aduana Paraguaya, conforme a la evolución impuesta por el comercio globalizado.

Cuenta con un módulo Recaudaciones el cual es el encargado de toda la gestión de valores: depósitos en efectivo o bancarios, créditos tributarios o aduaneros. En particular administra la:

- Percepción de valores.
- Afectaciones de valores a una liquidación aduanera.
- Generación de comprobantes de pago.
- Transferencia a las cuentas recaudadoras.
- Control de garantías, etc.

El estudio de dicho sistema y sus características aportará conocimiento para entender mejor el funcionamiento de los sistemas aduanales en general además muchas funcionalidades de las que presenta SOFIA podrán ser estudiadas.

1.3.5. Sistema Único de Aduana (SUA)

Un gran peso en la modernización de la Aduana cubana lo ha tenido el desarrollo de la informática y las comunicaciones, que unido a la consolidación de una amplia red propia de datos, ha permitido la automatización de la mayoría de los procesos administrativos y de control, todos con programas desarrollados completamente sobre software libre y por personal de la propia institución (3). El SUA es el sistema que presenta la AGR actualmente, automatiza los procesos aduaneros, posee una base de datos única y centralizada, a la que acceden en línea todas las aduanas del país.

El actual sistema que gestiona los procesos de Ingresos por la vía Comercial se desarrolló de forma estructurada por lo que su mantenimiento se vuelve costoso en tiempo y personas, además a este sistema no se le terminó la implementación de algunas de las nuevas funcionalidades que se deseaban y al no estar implementado con las ventajas que brindan los novedosos Framework de Desarrollo es incapaz de ser integrado con el actual sistema que se desarrolla para beneficio de la AGR, por lo que se tomó la decisión de desarrollar un sistema que satisfaga todas las necesidades de los clientes cumpliendo con la nueva estructura organizativa del proyecto en la universidad.

En el mundo existen muchos sistemas para los procesos de ingresos comerciales en las aduanas, pero ninguno se ajusta a las características propias de nuestro país, ni a los procesos que en este se desarrollan, incluso estos sistemas son diferentes en cada país al implementar las leyes propias del país en cuestión.

1.4. Principios

En las primitivas organizaciones sociales se encuentran ya los impuestos (directos), tanto en su forma personal, como en el servicio militar, en su forma real (parte del botín que se adjudica al jefe de la tribu). Posteriormente aparecen los tributos en especie, como la capitación, y los tributos sobre los rendimientos de la agricultura y de la ganadería (diezmos). Mucho después, cuando crecen las necesidades del Estado, el impuesto adopta la forma indirecta. Entre otros impuestos indirectos, el de aduana parece haber sido conocido en la India, así como en Persia y Egipto. En Grecia existieron, junto con el de capitación (sobre los extranjeros), el de consumo, sobre las ventas.

Las aduanas existieron en todos los pueblos de la antigüedad, según la importancia del comercio de cada ciudad, con la finalidad de controlar la entrada y salida de mercancías, y como forma de recaudar fondos.

En Cuba, la Aduana constituye un órgano de control en la frontera y de fiscalización en la actividad vinculada al comercio exterior. (3)

Entre sus misiones está garantizar la seguridad y protección de la sociedad socialista y de la economía nacional, así como la recaudación fiscal y la emisión de las estadísticas del comercio exterior, a través del cumplimiento de las políticas estatales de competencia aduanera para el tráfico internacional de viajeros, mercancías y medios de transporte. (3)

La Aduana cubana suscribió en 1995 el Convenio Internacional para la Simplificación y Armonización de los Regímenes Aduaneros (Convenio de Kyoto) y en 2009, ratificó el Convenio de Kyoto Revisado, siendo Cuba el único país signatario de Latinoamérica.

Paralelamente y como parte del proceso de modernización emprendido, se han adoptado un grupo de medidas para la facilitación y rapidez del despacho mercantil tales como la autovaloración y autoliquidación por el propio declarante, inspecciones de origen y destino, otorgamiento de facilidades al despacho y pago mediante convenios, sistemas de selectividad automatizada y la calificación y actualización en esa materia. (3)

1.4.1. Diseño de Software

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 (6). Cada metodología de diseño de software introduce heurísticas y notaciones propias, así como una visión algo particular de lo que caracteriza a la calidad del diseño. Sin embargo, todas las metodologías tienen varias características comunes:

- Un *mecanismo* para la traducción de la representación del campo de información en una representación de diseño.
- Una *notación* para representar los componentes funcionales y sus interfaces.
- *Heurísticas* para el refinamiento y la partición, y c*riterios* para la valoración de la calidad.

diseño puede definirse como:

Proceso iterativo de tomar un modelo lógico de un sistema junto con un conjunto de objetivos fuertemente establecidos para este sistema y producir las especificaciones de un sistema físico que satisfaga estos objetivos. (Gane − Sarson).

Actividad por la cual un relevamiento de datos y funciones de un sistema (modelo esencial) se traduce en un plan de implementación. El modelo es volcado en una tecnología determinada....el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física. (E. Taylor).

Objetivos del diseño

El objetivo más importante es:

Entregar las funciones requeridas por el usuario (Satisfaga una especificación funcional dada).

Pero además para lograr esto deben considerarse los aspectos de:

- **Rendimiento***:*

Cuán rápido permitirá el diseño realizar el trabajo dado un recurso particular de hardware. Es decir que contemple las limitaciones del medio donde será implementado el sistema, y alcance los requerimientos de performance y uso de recursos.

- **Control***:*

Protección contra errores humanos, máquinas defectuosas, o daños intencionales.

- **Confiabilidad**:

Facilidad con la cual el diseño permite modificar el sistema.

Generalmente estos tres factores trabajan unos contra otros, un sistema con muchos controles tenderá a degradar su rendimiento, un sistema diseñado para un alto rendimiento solo podrá ser cambiado con dificultad, etc. Además deberá:

- Satisfacer criterios de diseño sobre la forma interna y externa del producto obtenido.
- Satisfacer restricciones sobre el proceso de diseño en sí mismo, tales como su tiempo o costo, o las herramientas disponibles para hacer el diseño.

Una vez establecidos los requisitos del sistema, el diseño es la primera de tres actividades técnicas (*diseño*, *codificación* y *prueba*). Cada actividad transforma la información de forma que finalmente se obtiene un software para computadora validado.

El contexto en el que se ha desarrollado el software está fuertemente ligado a la evolución de los sistemas informáticos. Un mejor rendimiento del hardware, una reducción del tamaño y un costo más bajo, han dado lugar a sistemas informáticos más sofisticados. En el diseño se modela el sistema y se encuentra la forma para que soporte todos los requerimientos, incluyendo los requisitos no funcionales y otras restricciones, se asienta en el núcleo técnico del proceso de ingeniería del software y se aplica independientemente del paradigma de desarrollo utilizado. En la elaboración de la solución es preciso tener presente ciertos estándares, estilos y patrones que son muy útiles en la obtención de un diseño de software robusto. (7)

1.4.2. Arquitectura de Software

En el desarrollo de aplicaciones informáticas la arquitectura de software es un término muy frecuente y de gran referencia ya que sirve para guiar la implementación y el desarrollo del sistema desde etapas tempranas del diseño. Existen muchas definiciones acerca del tema, y aunque todas son consideraciones especializadas de distintos autores, coinciden en que a grandes rasgos se refiere a la estructura del sistema, constituida por componentes de software y relaciones entre estos.

Algunas definiciones formales han sido expuestas como por ejemplo. *"La arquitectura del software de un programa o sistema de computación es la estructura o estructuras del sistema que comprende los elementos del software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos".* (6)

"La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones". (8)

Esta definición es un poco amplia, por lo que una de las más aplicadas es la establecida por la IEEE STD 1471-2000 (*Software Engineering Standards Committee of the IEEE Computer Society, 2000*): *"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."*

1.5. Framework de Desarrollo

Se han realizado importantes progresos orientados a la reusabilidad del software a través de la aplicación del paradigma de la programación orientada a objetos y mediante el uso de los componentes tecnológicos. Sin embargo, estas tecnologías solo proveen el re-uso en el nivel individual, frecuentemente a menor escala. Con la aparición de los patrones de diseño se ha demostrado la reutilización de soluciones para resolver problemas de escala mayor enfocados en la solución de problemas sencillos. Sin embargo, el más complejo problema de la reutilización de soluciones es en el nivel de los grandes componentes, para que estos se puedan adaptar para las solicitudes individuales.

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

Un Framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un Framework proporciona estructura

al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

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

1.5.1. Symfony

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.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows (9). Symfony se diseñó para que se ajustara a los siguientes requisitos:

- 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 aplicaciones empresariales y adaptables 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 (9).

Symfony es el Framework utilizado para la creación del GINA. Brinda facilidades para el desarrollo con el lenguaje PHP. Utiliza metodologías y patrones de diseño que brindan la posibilidad de reutilizar el código. Presenta buena seguridad, es multiplataforma y sencillo de configurar. Todo el soporte del sistema GINA está sobre la base de Symfony. Incluye la premisa de no inventar la rueda y es por eso que utiliza lo mejor de diferentes Framework para proporcionar un trabajo limpio sobre los proyectos a realizar. Permite por medio de Propel el mapeo de las base de datos, transformando las tablas en clases del negocio. Crea formularios seguros para la entrada de información, soporta internacionalización, incluye lo mejor de los Framework más usados en el mundo.

1.5.2. ExtJs

En el mercado actualmente existen múltiples librerías de Javascript que permiten realizar todo tipo de maravillas en el navegador web. Quién iba a pensarlo, un lenguaje al que a pocos agrada termina haciendo más por la web de lo que podríamos haber imaginado.

Una de estas librerías, sobre la que hablaremos hoy día, se llama ExtJS y tiene muchas cualidades que la hacen interesante.

De acuerdo a la definición de la página web, ExtJS es una librería Javascript que permite construir aplicaciones complejas en Internet. Esta librería incluye:

- Componentes UI del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open source y comerciales.

Ventajas:

Antes de poder entrar a examinar ExtJS primero tenemos que hablar sobre RIA, acrónimo de Rich Internet Applications (Aplicaciones Ricas en Internet). Lo que RIA intenta proveer es aquello de lo que siempre ha adolecido la web, una experiencia de usuario muy parecida o igual a la que se tiene en las aplicaciones de escritorio.

Las aplicaciones web tradicionales tienen problemas como la recarga continua de las páginas cada vez que el usuario pide nuevo contenido, o la poca capacidad multimedia, para lo cual se han hecho necesarios plug-ins externos.

Junto con el reto de llevar la experiencia RIA a los usuarios comenzó el debate sobre cuál sería el mejor modo de atacar el problema. La historia de los últimos años nos ha traído diversas tecnologías, basadas en Flash (Adobe), Java (Sun), Silverlight (MS). Todas muy interesantes, pero con la desventaja de necesitar algún tipo de extensión en los navegadores que podría no estar presente. Ha sido esta limitante lo que le ha dado la victoria (al menos por el momento) al casi dejado de lado Javascript y la "nueva" tecnología conocida como AJAX.

ExtJS encaja dentro de este esquema como un motor que permite crear aplicaciones RIA mediante Javascript. Si enmarcamos a ExtJS dentro del desarrollo RIA, éste sería el render de la aplicación que controla el cliente y que ese encarga de enviar y obtener información del servicio.

Una de las grandes ventajas de utilizar ExtJS es que nos permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).

Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros.

Usar un motor de render como ExtJS nos permite tener además beneficios como:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.

- Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

Desventajas:

Esta es una lista de algunos puntos que encuentro negativos en las aplicaciones RIA:

- Necesita una plataforma. En este caso dependemos de ExtJS para mostrar los componentes y hacer el render de la aplicación.
- El javascript no es tan rápido como quisiéramos, sin embargo con la entrada de Google Chorme y el nuevo motor de Mozilla las cosas van mejorando.
- Descargas lentas. Al ser aplicaciones grandes, especialmente porque cargan todo al inicio, hace que el tiempo de descarga sea mayor al de una aplicación web tradicional.
- Problemas con los motores de búsqueda. Los motores de búsqueda indexan el contenido web estático por lo que los textos cargados de manera dinámica no serán encontrados.
- Accesibilidad. Estas aplicaciones tienen problemas con los programas de accesibilidad pues, al igual que los motores de búsqueda, no trabajan bien con texto cargado dinámicamente.
- No se pueden usar fuera de línea. Por su naturaleza web estas aplicaciones no pueden ser usadas en el cliente como cualquier otra aplicación.

En el caso específico de ExtJS existe un factor que hay que tener en cuenta y que nace del hecho de ser una librería Javascript independiente de cualquier tecnología del lado servidor. No existe una forma fácil de realizar binding entre los component visuales con el respectivo modelo, lo cual genera que el programador tenga que escribir más código para validar y enlazar los formularios.

Es difícil hacer que el servidor haga push de información desde el servidor web hacia el navegador, haciendo que el cliente tenga que constantemente hacer pooling para obtener datos actualizados del servidor.

La falta de un diseñador gráfico limita la difusión de la aplicación al no proveer una forma fácil y rápida de desarrollar en él, la situación me recuerda mucho a los inicios de Java Swing donde la falta de un diseñador gráfico limitaba su alcance.

Un problema grave del layout manager es que si usas código personalizado dentro de un componente, como por ejemplo un iframe, entonces se hace complicado lograr que se redimensione junto con el layout pues cae fuera de su control, sin embargo hay formas de lograrlo mediante código. (10)

1.6. Patrones de diseño utilizados

Cada patrón describe un problema que ocurre y una y otra vez en un entorno determinado y describe también el núcleo de la solución del problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo. Durante el desarrollo de software suelen surgir problemas comunes, los cuales son resueltos con el uso de alternativas de soluciones que pueden ser muy efectivas, pues en la práctica y en el transcurrir de los años se ha demostrado que son eficientes frente a dificultades presentadas, estas variantes son como esquemas de situaciones que se presentan con frecuencia, y traen asociada la manera en que se solventan, estos elementos son llamados patrones. Dentro de las definiciones de patrones se encuentra la propuesta por Craig Larman:

"Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos sobre cómo aplicarlo en nuevas situaciones, o sea, un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados."

- **Solucionan un problema:**

Los patrones capturan soluciones, no sólo principios o estrategias abstractas.

- **Son un concepto probado:**

Capturan soluciones demostradas, no teorías o especulaciones.

- **La solución no es obvia:**

Los mejores patrones generan una solución a un problema de forma indirecta.

- **Describen participantes y relaciones entre ellos:**

Describen módulos, estructuras del sistema y mecanismos complejos.

- **El patrón tiene un componente humano significativo:**

Todo software proporciona a los seres humanos confort y calidad de vida (estética y utilidad).

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones que se implementan con Symfony se pueden

clasificar en patrones GRASP (General Responsibility Assignment Software Patterns) (11) y patrones GOF (Gang Of Four) (12) y de estos se tratan a continuación los más relevantes.

1.6.1. Patrones GRASP

Experto: Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Propel para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan. (13)

Creador: En la clase Actions se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase Actions es “creador” de dichas entidades. Ejemplos de algunas funciones utilizadas en la clase Actions son: doSelect (), retrieveByPK (), doSelectOne (). (13)

Alta Cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase Actions, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades. (13)

Bajo Acoplamiento: La clase Actions hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja. (13)

Controlador: Todas las peticiones Web son manipuladas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Este patrón se evidencia entre otros en los “actions” y el index.php del ambiente. (13)

1.6.2. Patrones GoF

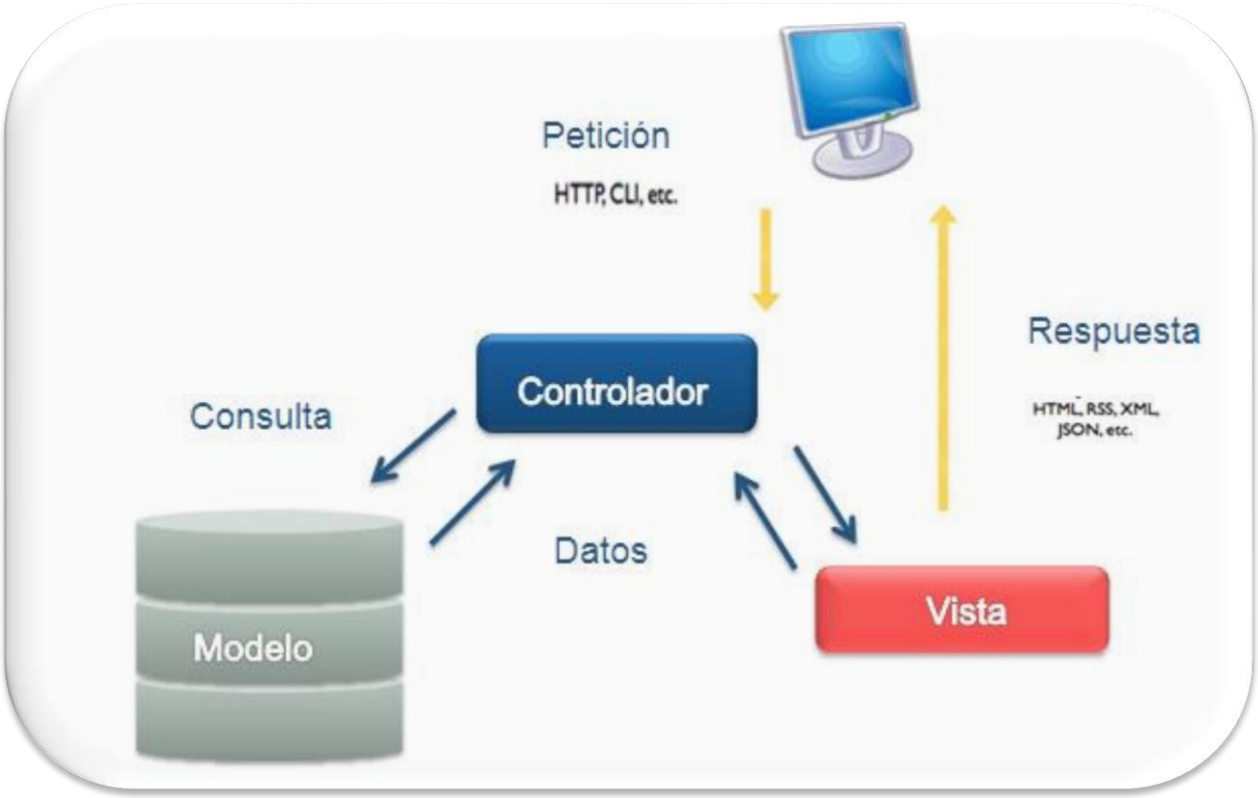
Patrón Singleton: Clase sfRouting – método getInstance Esta clase la utiliza el controlador frontal (sfWebFrontController) y se encarga de enrutar todas las peticiones que se hagan a la aplicación. El singleton sfRouting precisa otros métodos muy útiles para la gestión manual de las rutas: ClearRoutes (), hasRoutes (), getRoutesByName (). (9)

Patrón Command: Este patrón se observa en la clase sfWebFrontController, en el método dispatch(). Esta clase está por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Este patrón se aplica además en la clase sfRouting, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el Framework, la cual se puede activar o desactivar. En este método es parseada la URL con el objetivo de precisar los parámetros de la misma y de esta forma saber el Actions que debe responder a la petición. (9)

Patrón Decorator: Este método pertenece a la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado layout.php es el que contiene el Layout de la página. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este procedimiento es una implementación del patrón Decorator. (9)

Patrón Registry: Este patrón es muy útil para los desarrolladores en la Programación Orientada a Objetos. Este patrón es un medio sencillo y eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales. Este patrón se aplica en la clase **sfConfig**, que es la encargada de acumular todas las variables de uso global en el sistema. Symfony aplica además el patrón "Front Controller" (Controlador frontal), por lo que posee una estructura bien organizada de controladores, que comienza desde el "index.php" del ambiente y termina en los "**Actions**". Cada clase de esta capa tiene su responsabilidad y es única, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, y otros que se encargan de identificar mediante algunos datos las clases que deben realizar determinadas tareas (Patrón GoF **Command**, clase **sfRouting**) y las clases relacionadas con la configuración del sistema (**sfConfig**, y **sfConfigHandler**). (9)

1.7. Patrón Arquitectónico Modelo Vista Controlador



El uso del Framework que utiliza MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el Framework. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las

peticiones constituye el controlador. Aplicar el patrón MVC a una aplicación resulta bastante útil además de restrictivo. La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

SfController: Es la clase del controlador y se encarga de decodificar la petición y transferirla a la acción correspondiente.

SfRequest: Guarda los elementos que integran la petición (parámetros, cookies, cabeceras, etc.)

sfResponse: Posee las cabeceras de la respuesta y los contenidos. El contenido de este objeto se convierte en la respuesta HTML que se remite al usuario.

El singleton de contexto (que se obtiene mediante **sfContext: getInstance** ()): Guarda una referencia a todos los objetos que constituyen el núcleo de Symfony y puede ser accedido desde cualquier parte de la aplicación.

Symfony toma lo mejor de la arquitectura MVC y la realiza de modo que el desarrollo de aplicaciones sea rápido y sencillo. En el controlador se encuentran las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica de la aplicación. Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición.

La vista es la encargada de originar las páginas que son mostradas como resultado de las acciones, donde se encuentra el layout, que es común para todas las páginas de la aplicación. La vista en Symfony está conformada por varias partes, preparadas cada una de ellas especialmente para ser fácilmente transformable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

1.8. Lenguajes de programación

1.8.1. PHP como lenguajes del lado del Servidor

El sistema GINA se ha implementado sobre la base del PHP como lenguaje de programación del lado del servidor.

El **PHP** es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas (14)

Cuenta con cuatro grandes características que hacen que este lenguaje sea distinguido entre los demás existentes en el mundo. (14)

- **Velocidad:** No solo la velocidad de ejecución, la cual es importante, sino además no crear demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otro software, especialmente bajo ambientes Unix, generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.

- **Estabilidad:** Ninguna aplicación es 100% libre de errores, pero PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.

- **Seguridad:** PHP provee diferentes niveles de seguridad, estos pueden ser configurados de archivos de configuración que se encuentran en el servidor.

- **Simplicidad:** Es un lenguaje de programación simple de implementar por lo que usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

  Otra característica que cabe citar además es la conectividad. PHP dispone de una amplia gama de librerías, y agregarle extensiones para extender su alcance es muy fácil. Esto le permite al PHP ser utilizado en muchas áreas diferentes, tales como encriptado, gráficos y XML.

  Ventajas adicionales de PHP:

  PHP se ejecuta en (casi) cualquier plataforma utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en alrededor de 25 plataformas, incluyendo diferentes versiones de Unix, Windows y Macs. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al sistema operativo.
  PHP es completamente expandible. Está compuesto de un sistema principal (escrito por Zend), un conjunto de módulos y una variedad de extensiones de código.
  Muchas interfaces distintas para cada tipo de servidor. PHP actualmente se puede ejecutar bajo Apache, IIS, AOLServer, Roxen y THTTPD. Otra alternativa es configurarlo como módulo CGI.

Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos. (14)

1.8.2. Propel integrado a Symfony Framework

Propel, que también es un proyecto de Software libre, es una de las mejores capas de abstracción objetos-relacional disponibles en PHP5. Propel está completamente integrado en Symfony e incluso es su ORM (Mapeo Objeto-Relacional) por defecto. Ambos desarrollan una combinación rápida y flexible en cuanto al acceso a la información de una base de datos. En la solución se utiliza Propel como capa de persistencia del modelo físico de datos. (9)

Las principales novedades de Propel son las siguientes:

- *Migraciones*. Si añades o eliminas tablas y columnas en tu base de datos, Propel se encargará de hacer los cambios necesarios para que la aplicación siga funcionando bien y sin perder información.
- *Nuevos comportamientos*. Doctrine2 ha eliminado todos los comportamientos, pero Propel cada vez tiene más. Los últimos son Versionable (que permite guardar una versión para cada cambio en un objeto) y i18n (para hacer aplicaciones multi-idioma sin depender de lo que ofrezca cada Framework).
- Importación y exportación de información en los formatos XML, YAML, JSON y CSV.
- Tipos de columna avanzados. Se han añadido los tipos enum, array y object. (13)

1.8.3. Lenguajes del lado del Cliente

CSS:

Es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas.

La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (15)

A través del CSS se logra una apariencia agradable a las vistas diseñadas e implementadas para el GINA. Describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

JavaScript:

Lenguaje de programación del lado del cliente inventado por Brendan Eich en la empresa Netscape Communications. Usado en las páginas web para agregar mayor funcionalidad, interacción o animaciones. Entre sus principales características podemos citar:

- Es interpretado por el cliente.
- Está basado en objetos.
- Su código se integra en las páginas XHTML, incluido en las propias páginas.
- Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila.
- Es independiente de la plataforma, por lo que se ejecuta en cualquier sistema operativo. (16)

En el sistema GINA se utiliza, principalmente, para manejar objetos dentro de las páginas web. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades y modificación de archivos.

1.9. Visual Paradigm

Hoy en día, muchas empresas se han extendido a la adquisición de herramientas CASE, con el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema, desde el principio, hasta el final y así incrementar su posición en el mercado competitivo. Algunas de estas herramientas tienen un valor económico muy alto y requieren costos de entrenamiento de personal muy altos, además se enfrentan la falta de adaptación de la herramienta, a la arquitectura de la información en la que esta compuesta y a las metodologías de desarrollo utilizadas por la organización. Por otra parte, algunas herramientas CASE no ofrecen o evalúan soluciones potenciales para los problemas relacionados con sistemas, o simplemente no llevan a cabo ningún análisis de los requerimientos de la aplicación (17).

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. Como herramienta de modelado posee licencia gratuita. Ayuda a una construcción más rápida de aplicaciones de calidad y permite el dibujo de todos los tipos de diagramas de clases. Ofrece un diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. (18). Sus características más acentuadas son:

- Soporta Ingeniería Inversa.
- Generación del modelo físico de datos.
- Gestiona de forma eficiente y organizada los artefactos generados durante la modelación de la solución.
- Soporta la versión UML 2.1.

La herramienta de modelada seleccionada fue Visual Paradigm, teniendo en cuenta las características antes expuestas y mediante un profundo estudio de la misma. Se definió utilizar como herramienta de desarrollo de software Visual Paradigm para UML por soportar múltiples plataformas y permitir un diseño de mejores aplicaciones a un menor costo. Presenta una interfaz de uso intuitiva y con muchas facilidades a la hora de modelar los diagramas, además de su robustez, usabilidad y portabilidad. Brinda la posibilidad de realizar ingeniería inversa y flexibilidad para integrarse con los principales entornos de desarrollo como el Eclipse y el NetBeans.

1.10. Entorno de Desarrollo Integrado (IDE)

Un IDE en un programa informático compuesto por un conjunto de herramientas, utilizadas por los programadores para desarrollar código, consisten en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Pueden dedicarse en exclusiva a un sólo lenguaje de programación o bien para varios.

NetBeans IDE 6.9.1: La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.

- Framework basado en asistentes.

NetBeans IDE es una herramienta libre y gratuita sin restricciones de uso, pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación (19). Existe además un número importante de módulos para extenderlo. Algunas de las características que presenta integrado a PHP son:

- Creación de proyectos PHP.
- Integración con Symfony y ZenFramework.
- Editor de código fuente.
- Integración con PHPUnit Testing.
- Depuración de PHP.
- Integración con MySQL, PostgreSQL y Oracle.
- Integración con Sistemas de Control de Versiones.

1.11. Sistema Gestor de Base de Datos

Oracle Database 11g proporciona nuevas e innovadoras funcionalidades que garantizan alto rendimiento, alta escalabilidad, fiabilidad y seguridad mediante el uso de plataformas grid, asegurando altos niveles de calidad de servicio e incrementos de la flexibilidad de negocio reduciendo además los costes de explotación. Con Oracle Database 11g los clientes pueden resolver las problemáticas de negocio más exigentes en todas las áreas, incluyendo aplicaciones transaccionales, de inteligencia de negocio y de gestión de contenidos (19).

Oracle Database11g es la primera base de datos del mundo en incluir funcionalidades que permiten hacer pruebas de cambios en aplicaciones simulando las cargas reales generadas por los usuarios en los entornos de producción. Permite reducir de manera drástica los tiempos, riesgos y costes derivados de la implantación de cambios, asegurando que las aplicaciones se comportarán de manera adecuada y predecible tras las modificaciones. Los clientes ganan en flexibilidad puesto que pueden responder de manera más efectiva a los requerimientos cambiantes del negocio y hacer una gestión del cambio más efectiva.

Principales 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. (19)

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. (19) desde el año 1997.

Es la única herramienta privativa que se ha ganado un lugar en la AGR debido a sus insustituibles características. Ha estado presente desde 1997 en la aduana siempre brindando buenos resultados.

1.12. Conclusiones Parciales

En el capítulo se explicaron todas las herramientas que están definidas para la realización de los objetivos, herramientas que se caracterizan por las facilidades, ventajas y optimización que le brindan a los desarrolladores, así como las mejores prácticas que existen mundialmente sobre patrones y metodologías para la creación de páginas web. También se evidenció la necesidad de realizar un módulo para la AGR que logre la gestión eficaz y eficiente de los Ingresos por la vía Comercial, teniendo en cuenta que no existe sistema que sea completamente compatible con esas necesidades.

2. Capítulo II

2.1. Introducción

La arquitectura de un Software es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos, los principios que orientan su diseño y evolución, así como el contexto en el que se implantarán. El objetivo principal de esta arquitectura es aportar los elementos claves para el desarrollo eficaz de la futura aplicación. La solución propuesta en este trabajo está enmarcada dentro de la arquitectura actual definida para el Sistema de Gestión Integral de la Aduana. Con el objetivo de lograr un mayor entendimiento acerca de la integración de la propuesta de solución con el actual dominio arquitectónico del Sistema GINA, se realizará una breve descripción que comprende los aspectos fundamentales de cada elemento presente en la arquitectura. Además se obtendrán una serie de artefactos[5] de diseño que contienen la solución detallada sobre las clases a utilizar, la comunicación del módulo con otros subsistemas, el flujo que debe seguir las diferentes peticiones del usuario, la composición que presenta la base de datos y los pasos a seguir para poder realizar la elaboración de los diferentes requerimientos. El objetivo de todos estos resultados es definir la vía correcta y el orden en que se van a realizar las actividades facilitando la comprensión y rapidez a los implementadores logrando un mejor desarrollo de la solución.

2.2. Requisitos Funcionales

Para la realización del diseño se toma como entrada una serie de requisitos funcionales los cuales son mencionados a continuación:

- Cobrar Documentos.
- Cobrar Documentos Aplazados.
- Arquear Caja.
- Cancelar Mora.
- Cancelar Servicios.
- Cerrar Caja.
- Cobrar Cheques Devueltos.

[5] Un artefacto puede ser un modelo, o un elemento de modelo, un documento, todo lo que puede ser generado en el proceso.

- Devolver Cheque.
- Insertar Acuerdo Aplazamiento.
- Modificar Acuerdo Aplazamiento.
- Cancelar Acuerdo Aplazamiento.
- Facturar.
- Cancelar Factura.
- Cancelar Nota Crédito.
- Preparar Depósito.
- Iniciar Vía Apremio.
- Cobrar Vía Apremio.

2.3. Modelo del Diseño

El diseño de sistemas se define como el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un sistema, con suficientes detalles como para permitir su interpretación y realización física. La importancia del diseño del software se puede definir en una sola palabra ***Calidad***, dentro del diseño es donde se fomenta la calidad del Proyecto. El diseño es la única manera de materializar con precisión los requerimientos del cliente. El diseño del software es un proceso y un modelado a la vez. El proceso de diseño es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del Sistema a construir. A lo largo del diseño se evalúa la calidad del desarrollo del proyecto con un conjunto de revisiones técnicas:

- El diseño debe implementar todos los requisitos explícitos contenidos en el modelo de análisis y debe acumular todos los requisitos implícitos que desea el cliente.
- Debe ser una guía que puedan leer y entender los que construyan el código y los que prueban y mantienen el Software.
- El diseño debe proporcionar una completa idea de lo que es el Software, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación

2.4. Diagrama de Secuencia Orientado a Actividades del Negocio

El Departamento de Soluciones para la Aduana con el propósito de orientar el diseño lo mejor posible a la etapa de implementación utiliza los diagramas de secuencia orientados a actividades del negocio. El objetivo principal de este diagrama es especificar la interacción entre las clases, así como el flujo a seguir de las actividades a realizar, experimentando un nivel de especificación tal que permita a los programadores lograr la implementación de la solución en menor tiempo y con mejor calidad. Se representa por calles, actividades y sus relaciones, incluyendo también comentarios para mayor comprensión. Cada requisito funcional puede tener asociado uno o varios diagramas en dependencia del nivel de complejidad y la cantidad de funcionalidades comunes. Es considerado un híbrido entre los diagramas de actividades y los de secuencias propuestos por RUP en la etapa de diseño. La figura 1 muestra un ejemplo de diagrama de este tipo, pertenece al requisito funcional Cobrar Documento, como se puede observar está compuesto por tres calles, cada una de ellas representa las áreas que abarca el negocio de esta funcionalidad, se inicia cuando se da clic en el botón Cobrar Documento.

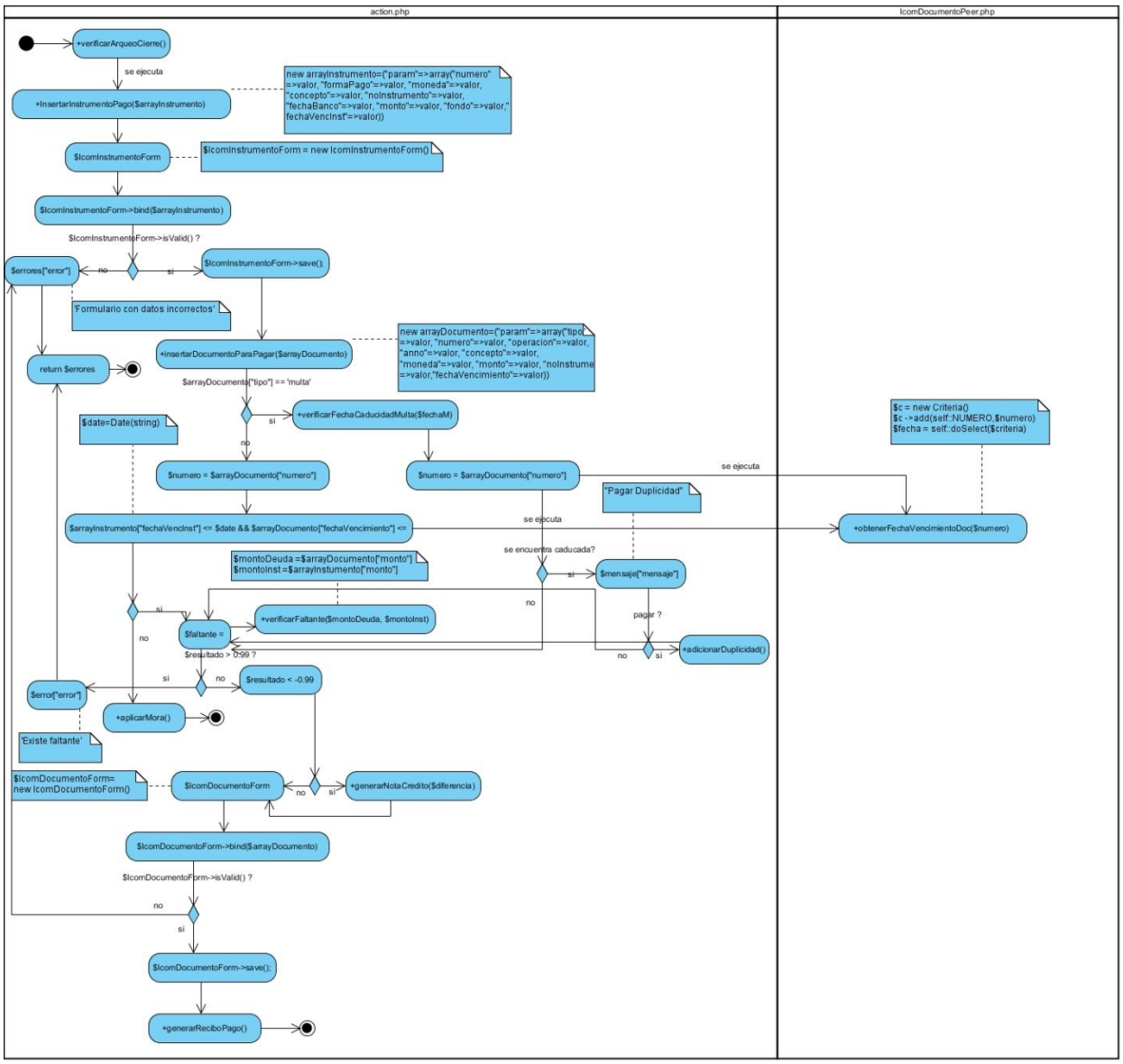


Figure 1-Diagrama de Actividades orientado al negocio. Caso de uso Cobrar Documento

2.5. Diseño de la Base de Datos

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda, el diseño de la base de datos. El diseño de la base de datos es el hito más importante que se realiza en el desarrollo de un sistema que necesite trabajar con información persistente. En general, el objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información. (20)

Son muchas las consideraciones a tomar en cuenta al momento de hacer el diseño de la base de datos, quizá las más fuertes sean:

- La velocidad de acceso.
- El tamaño de la información.
- El tipo de la información.
- Facilidad de acceso a la información.
- Facilidad para extraer la información requerida.
- El comportamiento del manejador de bases de datos con cada tipo de información.

2.5.1. Diagrama de clases persistentes

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema. Las clases están compuestas por los atributos pasivos y los activos, mientras que las relaciones pueden ser de diferentes tipos como herencia, Composición, Agregación, Asociación y Uso.
La **figura 2** muestra el diagrama de diseño de las clases persistentes del sistema, sirve como una aproximación al diseño definitivo del modelo de datos.

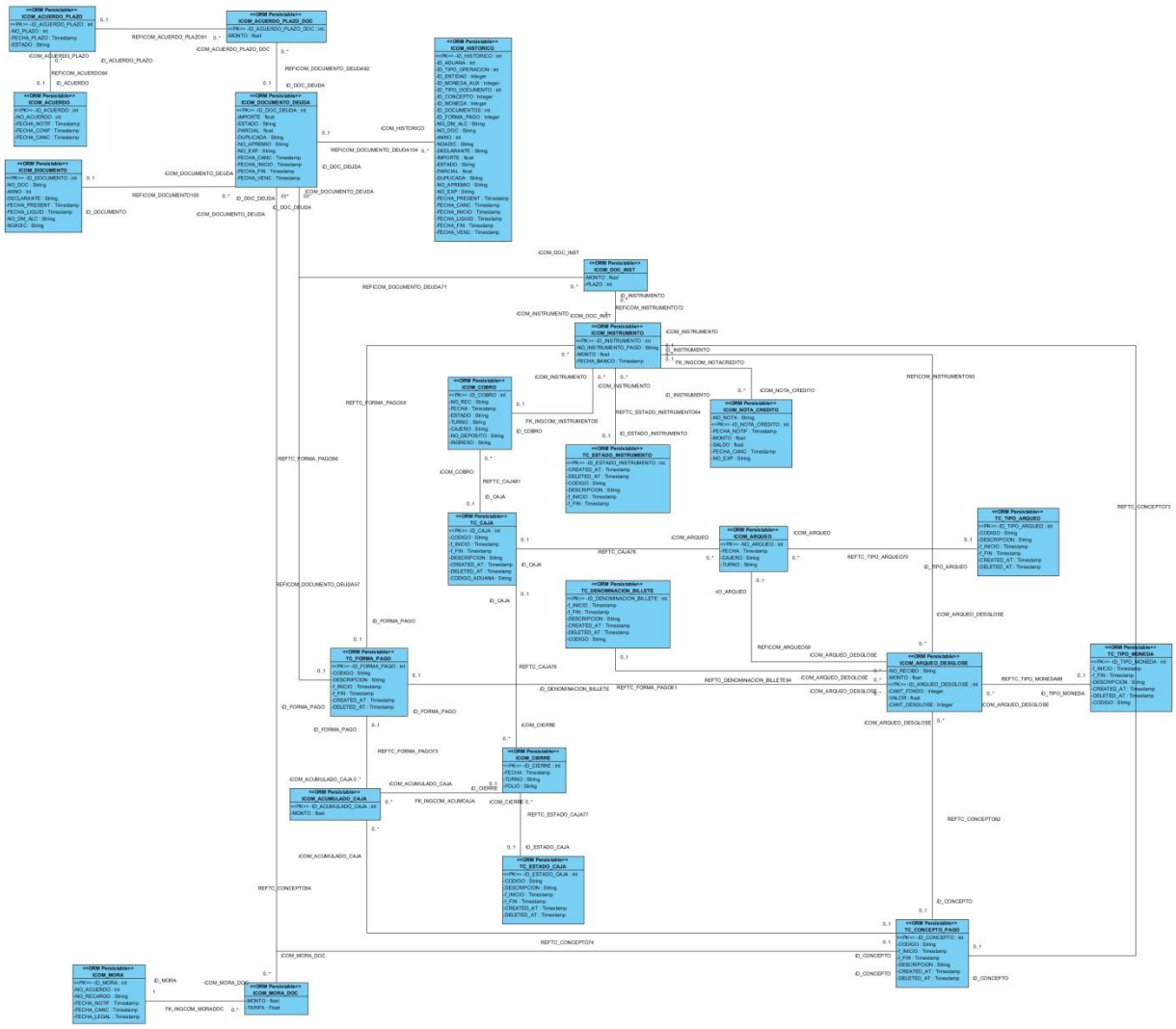


Figure 2- Diagrama de Clases Persistentes

2.5.2. Diagrama Entidad – Relación

El modelo entidad-relación **figura 3** generado para el sistema contiene las tablas (y las relaciones entre ellas) que guardarán la información de las formalidades de Ingresos por la vía Comercial.

Todas las clases en su conjunto conforman un modelo de entidad-relación, normalizado en Tercera Forma Normal (3FN), según la propuesta original de Codd (1972), que plantea que una relación está en tercera forma normal si todos los atributos de la relación dependen funcionalmente sólo de la clave, y no de ningún otro atributo. La *normalización de los datos* puede considerarse como un proceso durante el cual los esquemas de relación que no cumplen las condiciones se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que cumplen las condiciones establecidas. Un objetivo del proceso de normalización realizado para el modelo antes presentado, es garantizar que no ocurran anomalías de actualización.

Las tablas de color verde son las más importantes arquitectónicamente debido a que en ellas se almacenan los datos principales.

Se muestra el diagrama Entidad – Relación perteneciente a la aplicación.

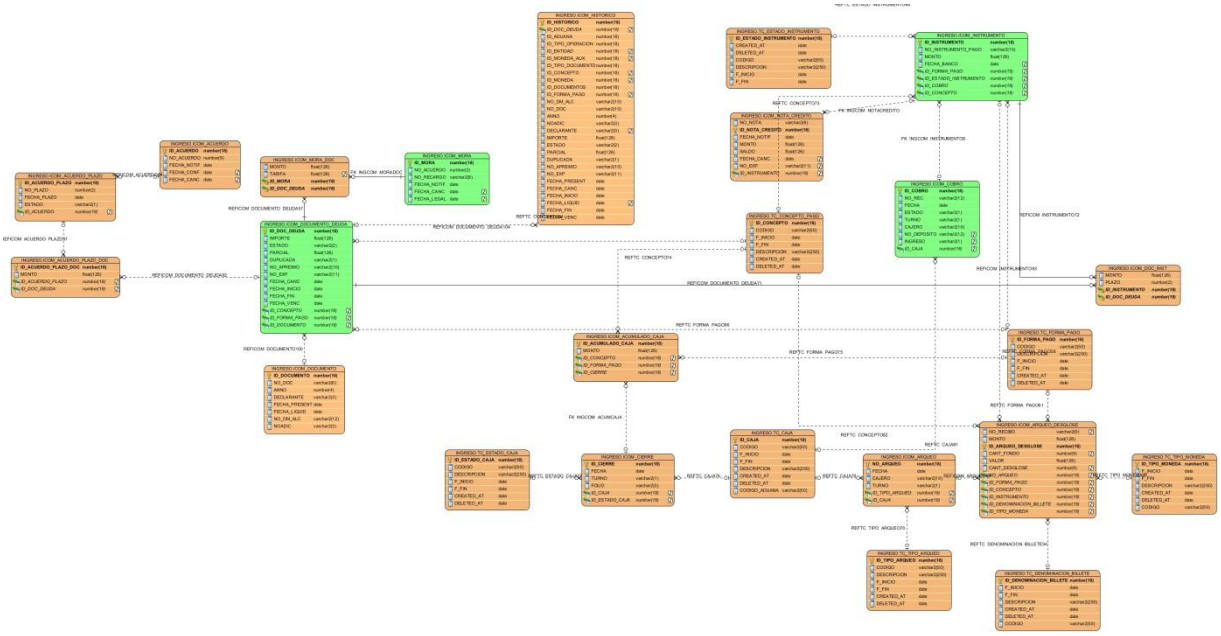


Figure 3- Diagrama Entidad - Relación

2.6. Métricas para la evaluación del diseño

Las **métricas** son la maduración de una disciplina, que, según Pressman van a ayudar a la (1) evaluación de los modelos de análisis y de diseño, (2) en donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y (3) ayudarán en el diseño de pruebas más efectivas.

Es por eso que propone un proceso de medición, el cual se puede caracterizar por cinco actividades:

(1) ***Formulación***: La obtención de medidas y **métricas del software** apropiadas para la representación de software en cuestión.

(2) ***Colección***: El mecanismo empleado para acumular datos necesarios para obtener las métricas formuladas.

(3) ***Análisis***: El cálculo de las métricas y la aplicación de herramientas matemáticas.

(4) ***Interpretación***: La evaluación de los resultados de las métricas en un esfuerzo por conseguir una visión interna de la calidad de la representación.

(5) ***Realimentación***: Recomendaciones obtenidas de la interpretación de métricas técnicas trasmitidas al equipo de software.

Una métrica es un instrumento que cuantifica un criterio. Una estructura de proyecto que sirve de guía al equipo de trabajo e involucra a los usuarios y a los puestos directivos en su desarrollo y en sus puntos decisivos, un conjunto de productos finales a desarrollar. En la mayoría de los desafíos técnicos, las métricas nos ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto. El producto se mide para intentar aumentar su calidad.

Las mediciones del mundo físico pueden englobarse en dos categorías: medidas directas y medidas indirectas.

Medidas Directas. En el proceso de ingeniería se encuentran el costo, y el esfuerzo aplicado, las líneas de código producidas, velocidad de ejecución, el tamaño de memoria y los defectos observados en un determinado periodo de tiempo.

Medidas Indirectas. Se encuentra la funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento, etc. (21).

Existen numerosas métricas para le medición del diseño desarrollado, dentro de las que se encuentran:

- Tamaño operacional de clase (TOC): métrica que se basa esencialmente en la cantidad de funcionalidades que presenta la clase.
- Relaciones entre Clases (RC): métrica que se basa en la cantidad de relaciones que presenta cada clase con el resto. Se establece relación entre dos clases cuando se tiene instancias, atributos y llamadas a funcionalidades de una clase en otra.
- Profundidad de herencia (PH): mide el máximo nivel en la jerarquía de herencia. Es la cuenta directa de los niveles en la jerarquía de herencia.
- Número de descendientes (ND): es el número de subclases subordinadas a una clase en la jerarquía, es decir, el número de subclases que pertenecen a una clase.
- Número de operaciones redefinidas para una clase hija (NOR): muestra en qué medida las subclases redefinen el comportamiento de sus superclases.

2.7. Métrica Tamaño Operacional de Clase

Se empleará la métrica Tamaño Operacional de Clase (TOC) la cual está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Responsabilidad: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.

Complejidad de implementación: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.

Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 1- Responsabilidad

	Categoría	Criterio
Responsabilidad	Baja	<= Prom
	Media	Entre Prom. Y 2* Prom.
	Alta	> 2* Prom.

Tabla 2- Complejidad e Implementación

	Categoría	Criterio
Complejidad e Implementación	Baja	<= Prom
	Media	Entre Prom. Y 2* Prom.
	Alta	> 2* Prom.

Tabla 3- Reutilización

	Categoría	Criterio
Reutilización	Baja	<= Prom
	Media	Entre Prom. Y 2* Prom.
	Alta	> 2* Prom.

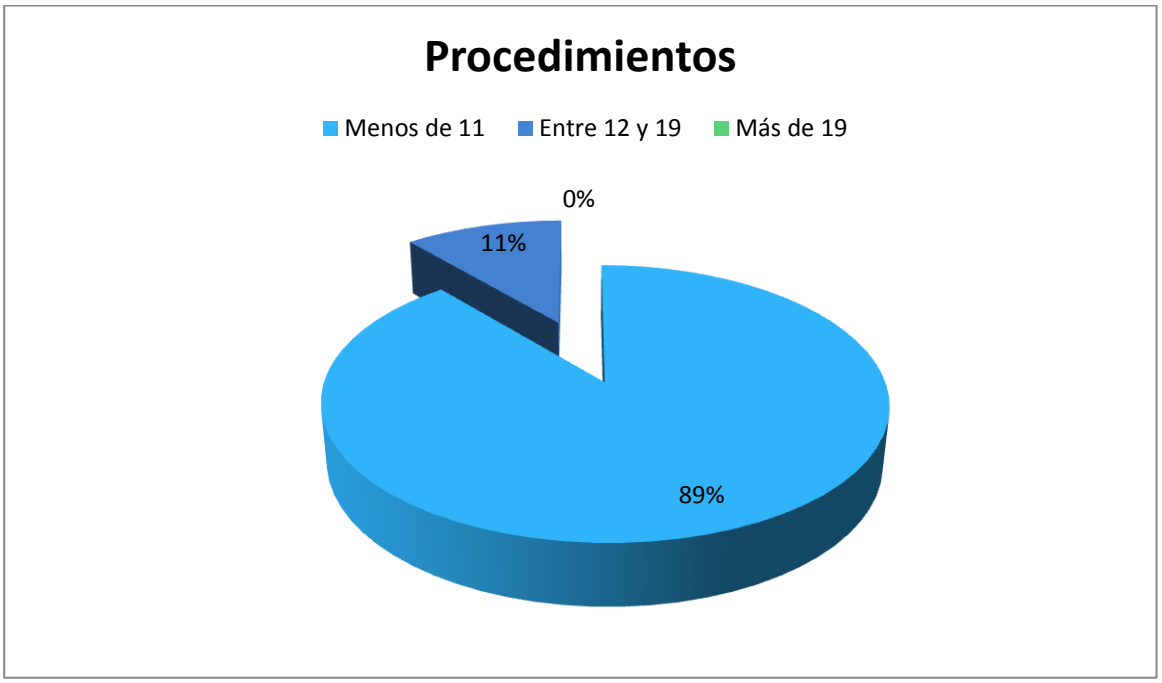


Figure 4- Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos según la métrica TOC.

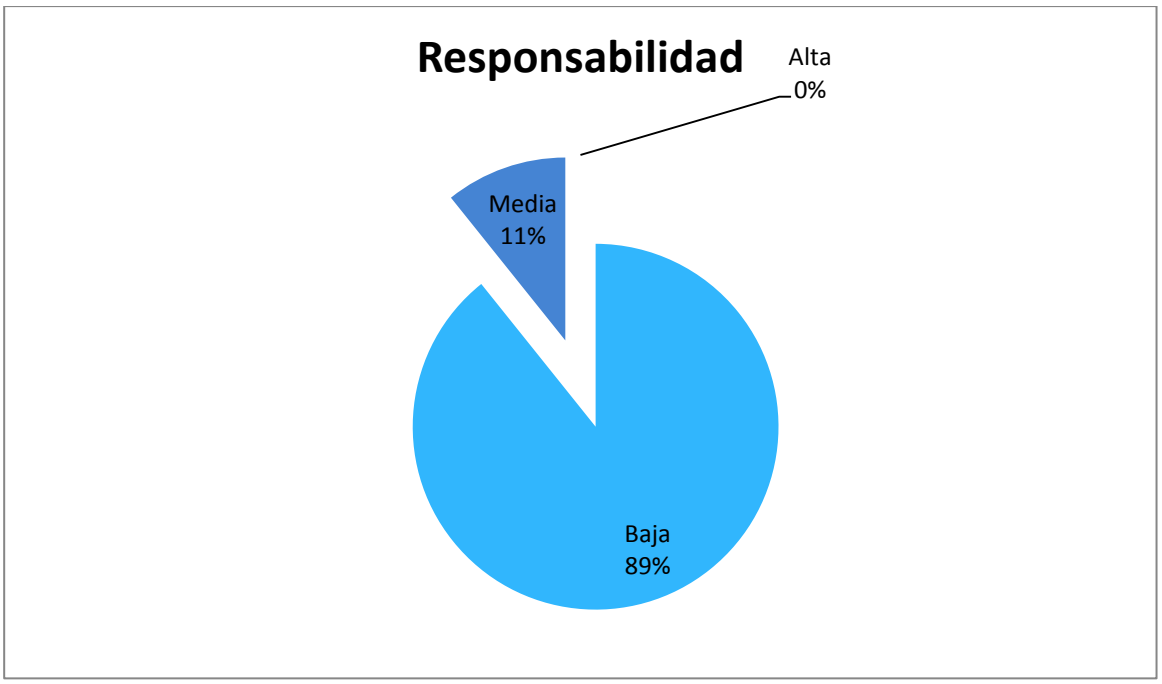


Figure 5- Responsabilidad por Clases

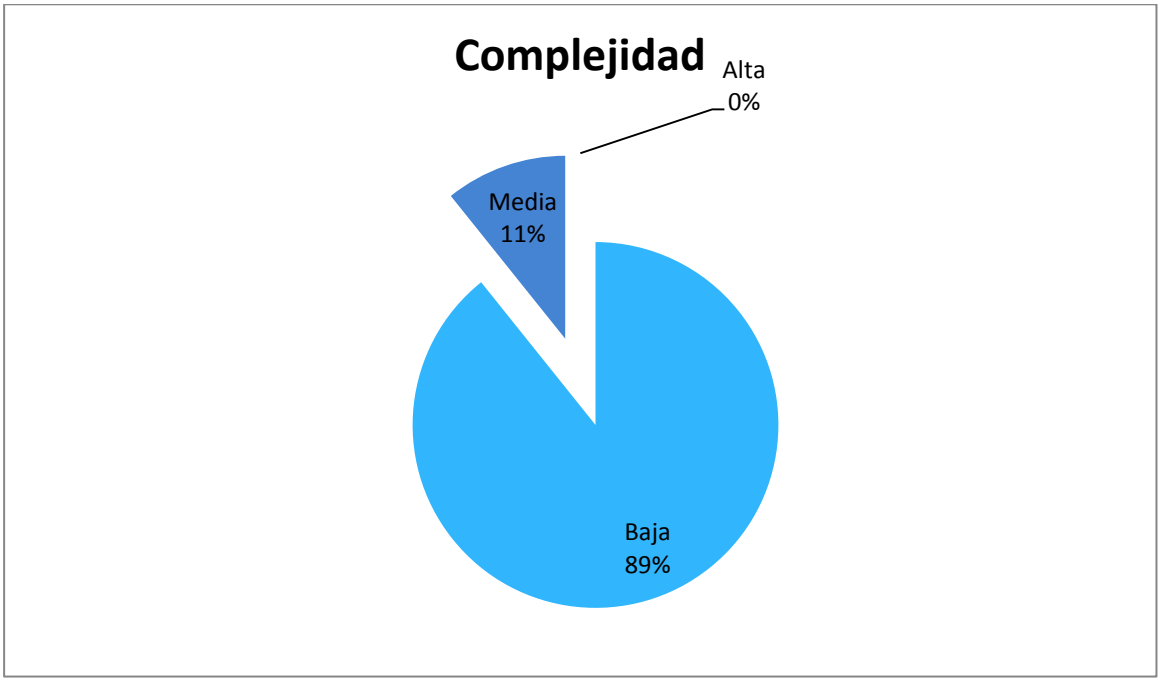


Figure 6- Complejidad por Clases

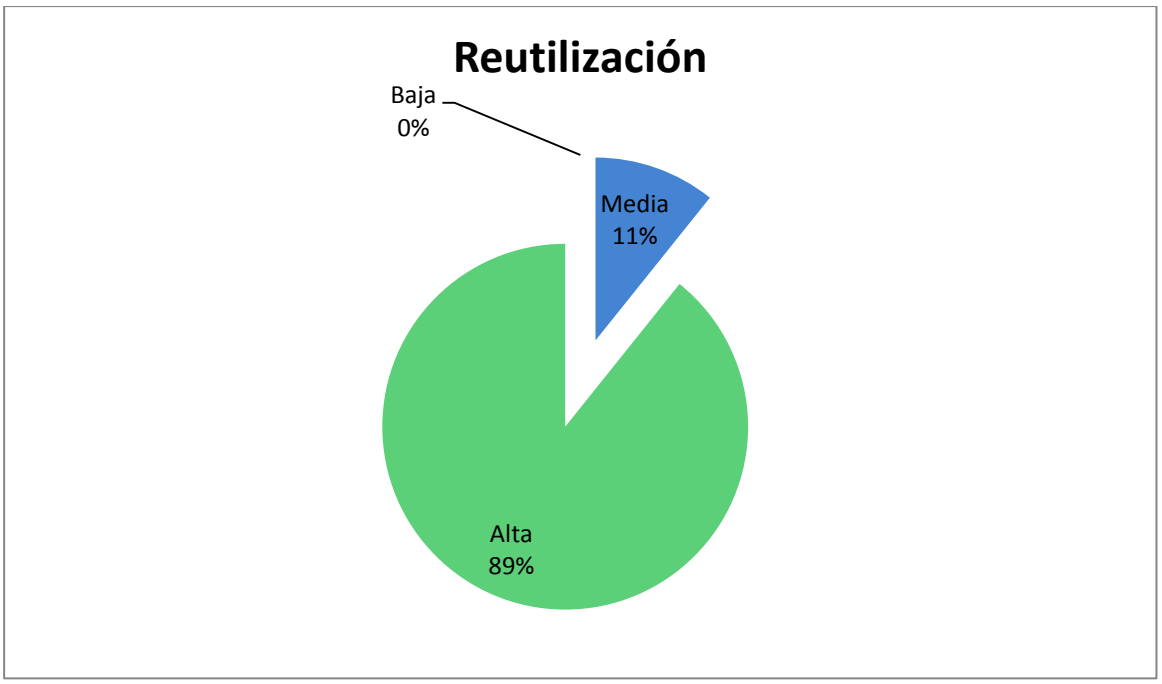


Figure 7- Reutilización por Clases

Durante la evaluación de la métrica TOC los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio (89%); de manera que se puede confirmar la elevada reutilización (elemento clave en el proceso de desarrollo de software) y cómo se reducen la responsabilidad y la complejidad de implementación. Se puede concluir que el diseño realizado es simple, además de que tiene una calidad aceptable.

2.8. Métrica Relaciones entre Clases

También se realizó la métrica de Relaciones entre Clases (RC), para definir el nivel de dependencia existente. Se realiza una valoración, sobre el nivel que presentan las propiedades de acoplamiento y cohesión. Debido que el modelo de clases del sistema no presenta herencias, no se desarrollaron las otras métricas definidas.Atributos de calidad que se abarcan en la métrica RC:

Acoplamiento. Consiste en el nivel de dependencia existente entre las clases.

Complejidad de Mantenimiento. Consiste en el grado de dificultad que se presenta a la hora de realizar el mantenimiento del sistema.

Tabla 4- Acoplamiento

	Categoría	Criterio
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	>2

Tabla 5- Mantenimiento

	Categoría	Criterio
Mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 6- Representación de las Dependencias por Clases

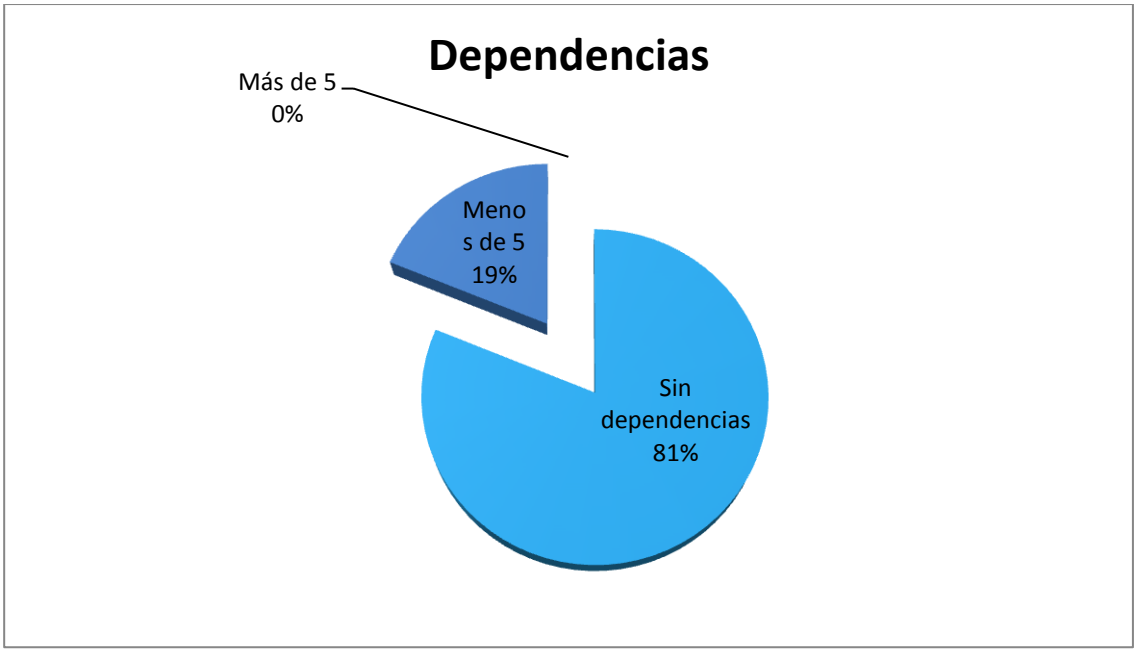


Tabla 7- Nivel de Acoplamiento por Clases

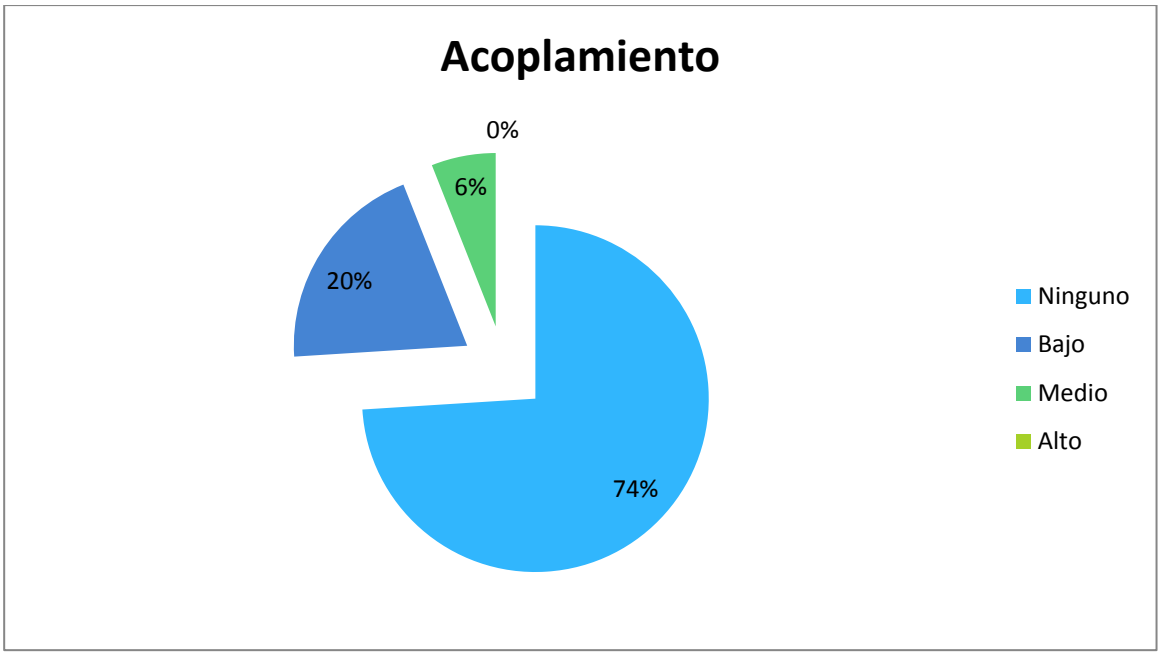
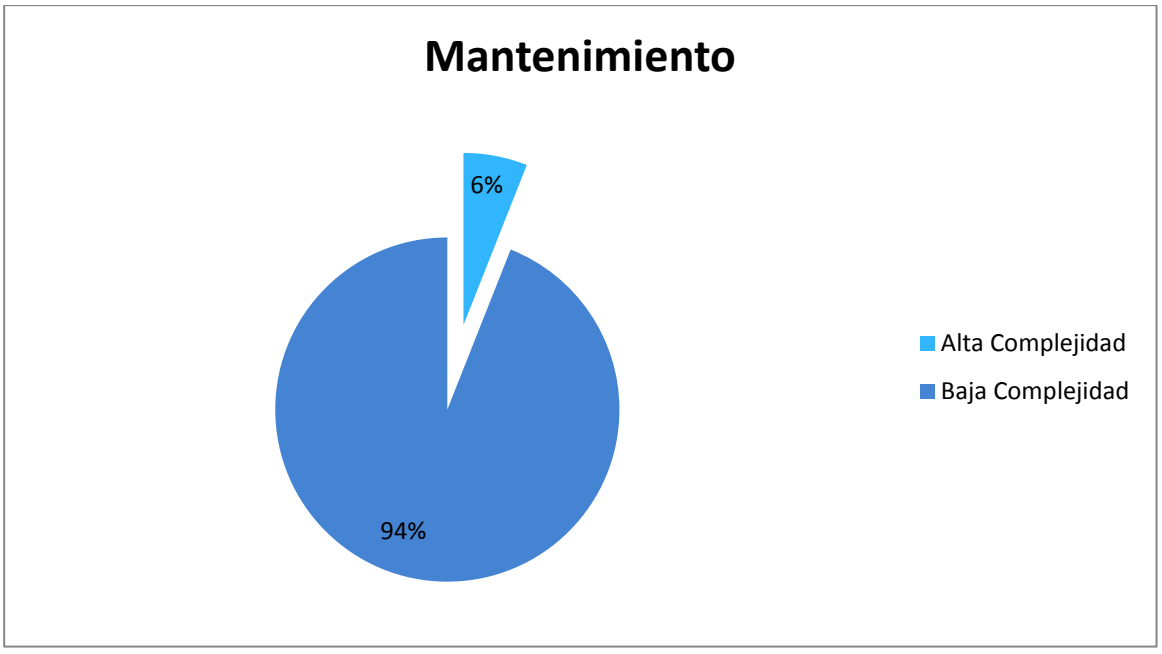


Tabla 8- Nivel de la Complejidad de Mantenimiento por Clases



La evaluación de la Métrica Relación entre Clases (RC), donde se definen los niveles de acoplamiento y complejidad de mantenimiento de las clases del sistema poseen un nuvel aceptable del 94%. Se han obtenido resultados que demuestran el bajo acoplamiento y el bajo nivel de complejidad que conlleva el mantenimiento del sistema. Se concluye que el diseño ha sido simple y tiene una calidad acceptable para realizar la implementación.

2.9. Implementación del sistema

El flujo de Implementación persigue como objetivo fundamental el desarrollo de la arquitectura y el sistema como un todo. El propósito de la implementación es definir la organización del código, en términos de los subsistemas de implementación organizados en capas, implementar los elementos de diseño en términos de los elementos de implementación, probar y desarrollar componentes como unidades y por último integrar los resultados producidos por los desarrolladores individuales (o equipos) en un sistema ejecutable.

2.10. Estándar de Codificación

En la actualidad existen diferentes estándares para cada uno de los lenguajes, su utilización permite una comunicación fluida y directa entre los programadores de manera que se favorece la reutilización y mantenimiento de los sistemas. El Departamento de Soluciones para la Aduana de CEIGE presenta su propia propuesta de estándar de codificación para todas las aplicaciones a desarrollar. Comprende todo el código generado bajo la tecnología y el lenguaje PHP y que utilicen la arquitectura regida por la utilización del Framework Arquitectónico Symfony. A continuación citamos algunos de los aspectos relevantes que presenta dicho documento:

Acciones (métodos o funcionalidades de la clase nombreDelModuloAction.class.php):

- Dentro de las especificaciones del Framework está que cada una de estas acciones debe comenzar con la palabra ***execute***.
- Todos los nombres de acciones deben estar en la nomenclatura "**Lower*CamelCase***" comenzando por la palabra ***execute***.
- En caso de ser acciones referentes a un módulo de CRUD de una tabla deben ser nombres específicos como ***executeNuevo***, ***executeEditar***, entre otros.
- Los nombres de las acciones deben especificar con la menor cantidad de palabras cual es el objetivo de la acción, de ser posible estar en infinitivo. Debe especificar bien claro cuál es la acción que se pretende ejecutar con la acción pero sin especificar los parámetros que recibe.

Nombre de las clases:

- Los nombres de las clases deben estar expresados en notación ***LowerCamelCase***.
- No se deben utilizar guiones bajos en su nombre "_".
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.
- En los nombres compuestos por más de tres palabras se debe revisarse el diseño, no sea que se le estén dando a la clase más responsabilidades de las que realmente tiene.

Nombres de las funciones:

- Todas las funciones definidas por los desarrolladores deben seguir la nomenclatura LowerCamelCase, a no ser que para cierto ámbito se especifiquen características específicas.

Además deben cumplir con las siguientes disposiciones de forma general:

- Los nombres de las funciones deben dejar reflejado claramente cuál es la acción que realiza el mismo.
- Se debe apoyar en la utilización de sufijos que ayuden a identificar el resultado final de la ejecución de un método.
- Se debe apoyar en los prefijos para expresar la acción que realiza sobre un elemento determinado.

Variables:

- Los nombres de las variables deben expresar claramente el contenido de la misma.
- Pueden estar referidas en singular o plural.
- Se definen al principio de las estructuras donde son utilizadas.
- En caso de que no se le asigne un valor inicial se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.
- De esta nomenclatura se exceptúan las variables generadas por el Framework. (22)

2.11. Tratamiento de Errores

El tratamiento de errores es un aspecto importante a la hora de desarrollar un software, debido a que son muchos los errores cometidos por los usuarios, ya sean accidentes o no. Para garantizar el correcto funcionamiento del sistema es fundamental identificar y controlar los posibles problemas que se pueden presentar a la hora de interactuar con el software. En el módulo Ingresos Comerciales se realizan varios tipos de validaciones de errores. Las validaciones del negocio se encargan de

controlar el flujo de los datos recibidos en el controlador para evitar consecuencias alarmantes. Se llevan a cabo en las diferentes clases con el uso de funciones propias del lenguaje. En caso de que exista algún error, es notificado al usuario para que pueda este corregirlos. Las validaciones realizadas en las vistas son muy importantes en la entrada correcta de los datos a la aplicación. Tienen como función principal evitar que se introduzcan datos incorrectos en los diferentes campos de los formularios para impedir ataques contra el sistema. Otro de los aspectos importantes utilizado en el tratamiento de errores del sistema desarrollado es la utilización de los formularios para insertar, eliminar o modificar valores de la base de datos, utilizando cada uno de los validadores implementados de manera que se garantiza una mayor coherencia de los mismos (22).

2.12. Comunicación entre Capas

El marco de trabajo que se utiliza es Symfony y éste utiliza el patrón arquitectónico MVC que separa el sistema en tres capas diferentes, por lo que se hace necesario realizar la comunicación entre las mismas.

La capa de la Vista es la encargada de recoger la información deseada y enviarla a través de sus formularios hacia la capa del controlador, por medio en algunos casos de peticiones AJAX para lograr mayor rapidez y a través de objetos JSON y en otras ocasiones por XML logrando el intercambio de información y objetos completos entre las capas de presentación y la del controlador.

- La capa del controlador, que contiene el código que liga la lógica de negocio con la presentación, está dividida en varios componentes que se utilizan para diversos propósitos y se encargan de obtener los datos y según la petición, realizar la acción que le corresponde, la cual devolverá una respuesta determinada:
- El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse.
- Las acciones contienen la lógica de la aplicación. Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación.
- Los objetos request, response y session dan acceso a los parámetros de la petición, las cabeceras de las respuestas y a los datos persistentes del usuario. Se utilizan muy a menudo en la capa del controlador.
- Los filtros son trozos de código ejecutados para cada petición, antes o después de una acción. Por ejemplo, los filtros de seguridad y validación son comúnmente utilizados en aplicaciones web. Puedes extender el Framework creando tus propios filtros.

Symfony utiliza Propel que es un ORM para PHP, transformando las tablas de la base de datos en objetos y realizando el tratamiento de la información mediante ellos, con los que se puede recuperar, insertar y modificar datos, proporcionando la posibilidad que no sea necesario preocuparse por las conexiones, ni de escribir SQL.

2.12.1. Ejemplo de Comunicación entre Capas

En este ejemplo se plasmará la solución del requisito funcional Arquear Caja.

Primeramente el usuario se encuentra en la ventana principal que presenta un menú que contiene todas las funcionalidades referentes a Ingresos Comerciales. Selecciona en el menú “Caja” el submenu “Arquear Caja”.

Figure 8.-Página Principal

Se cargan los formularios necesarios para realizar el Arqueo de Caja.

Figure 9-Formularios: Arquear Caja

Pantalla lista para Insertar lo datos necesarios para el Arqueo de Caja.

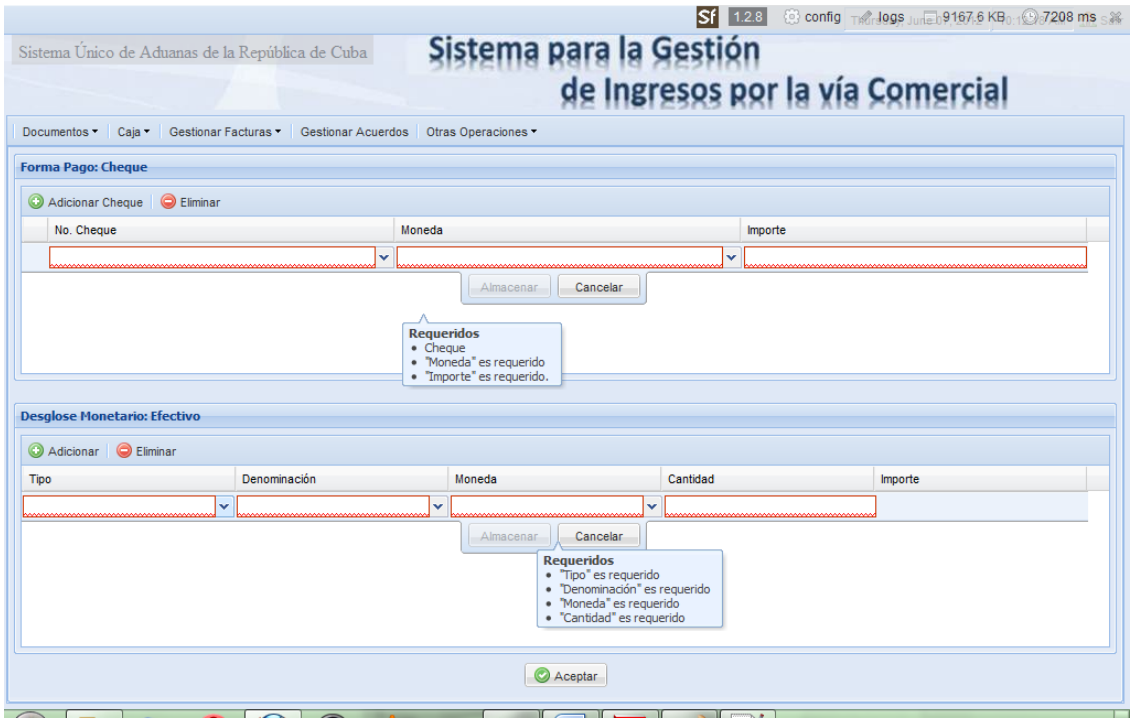


Figure 10.- Adicionar Cheque

Figure 11-Arqueo Caja

A través de peticiones Ajax se envían los datos a la clase Actions en formato json como se muestra en la siguiente figura:

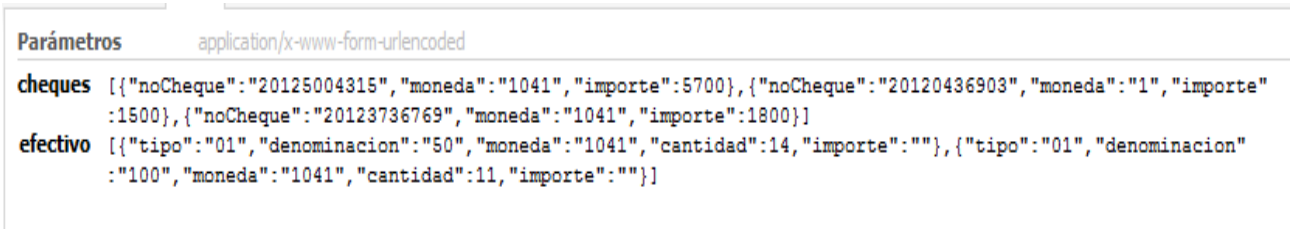


Figure 12-Formato datos.

Los parámetros se decodifican:

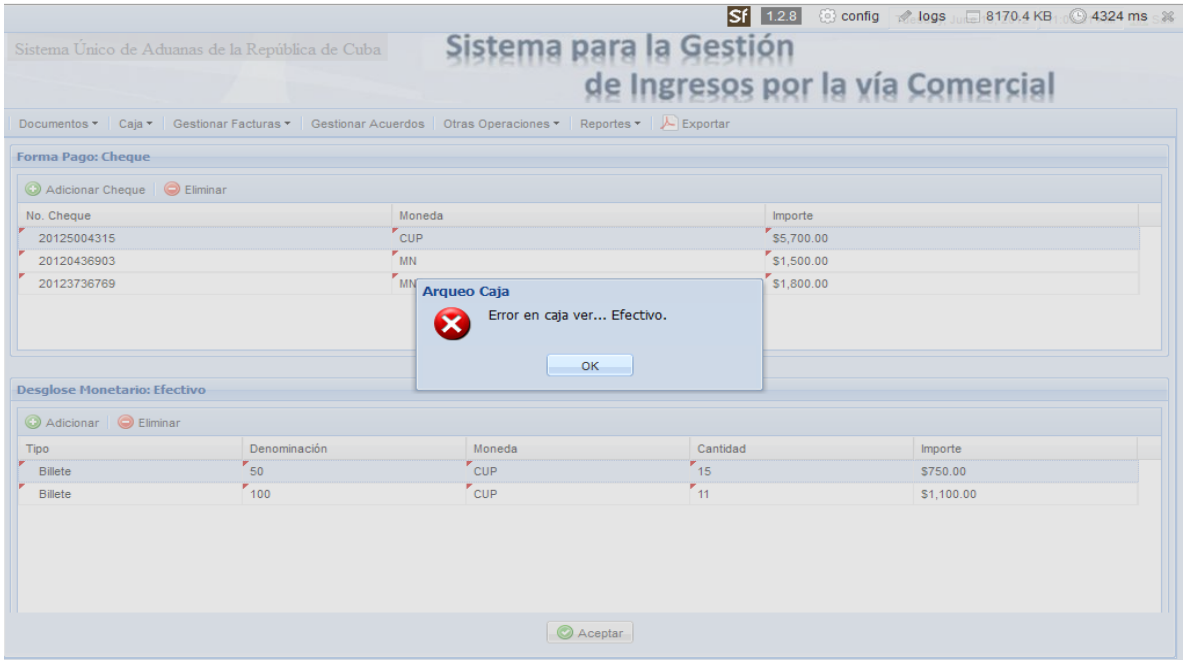
```
Array
(
    [0] => stdClass Object
        (
            [tipo] => 01
            [denominacion] => 50
            [moneda] => 1041
            [cantidad] => 14
            [importe] =>
        )

    [1] => stdClass Object
        (
            [tipo] => 01
            [denominacion] => 100
            [moneda] => 1041
            [cantidad] => 11
            [importe] =>
        )

)
```

Figure18. Datos Efectivo

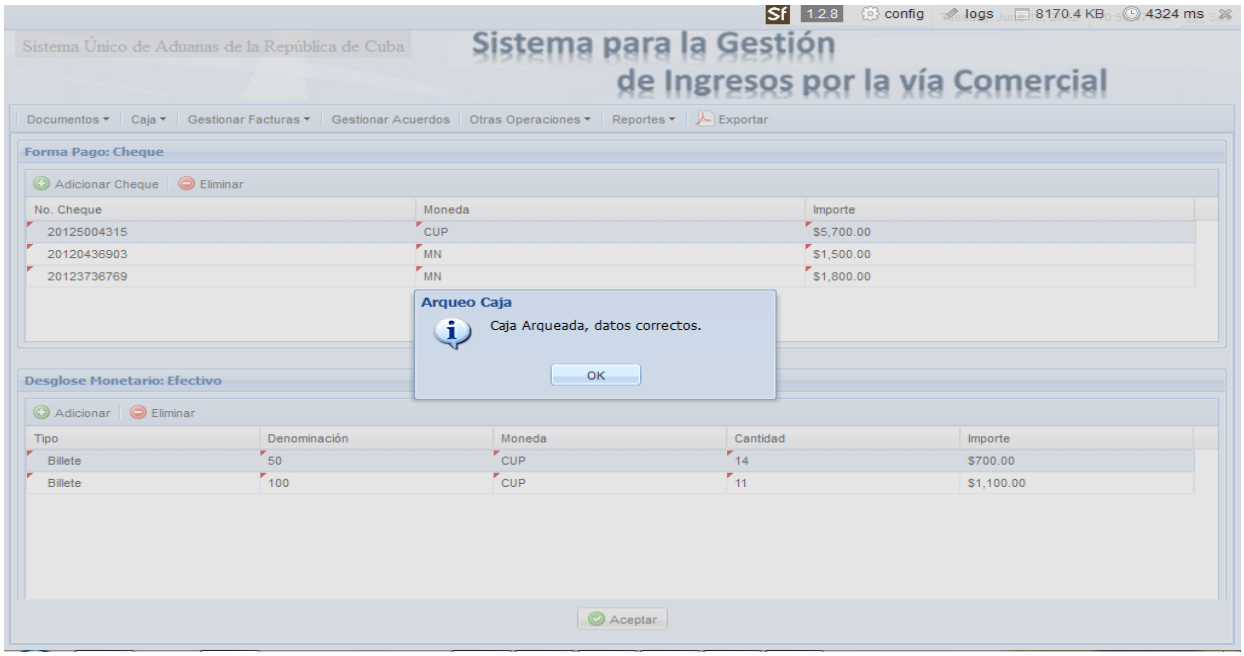
```
Array
(
    [0] => stdClass Object
        (
            [noCheque] => 20125004315
            [moneda] => 1041
            [importe] => 5700
        )

    [1] => stdClass Object
        (
            [noCheque] => 20120436903
            [moneda] => 1
            [importe] => 1500
        )

    [2] => stdClass Object
        (
            [noCheque] => 20123736769
            [moneda] => 1041
            [importe] => 1800
        )

)
```

figure19. Datos Cheque

En caso de que se produzca un error se notifica al usuario mediante un objeto json, por ejemplo.



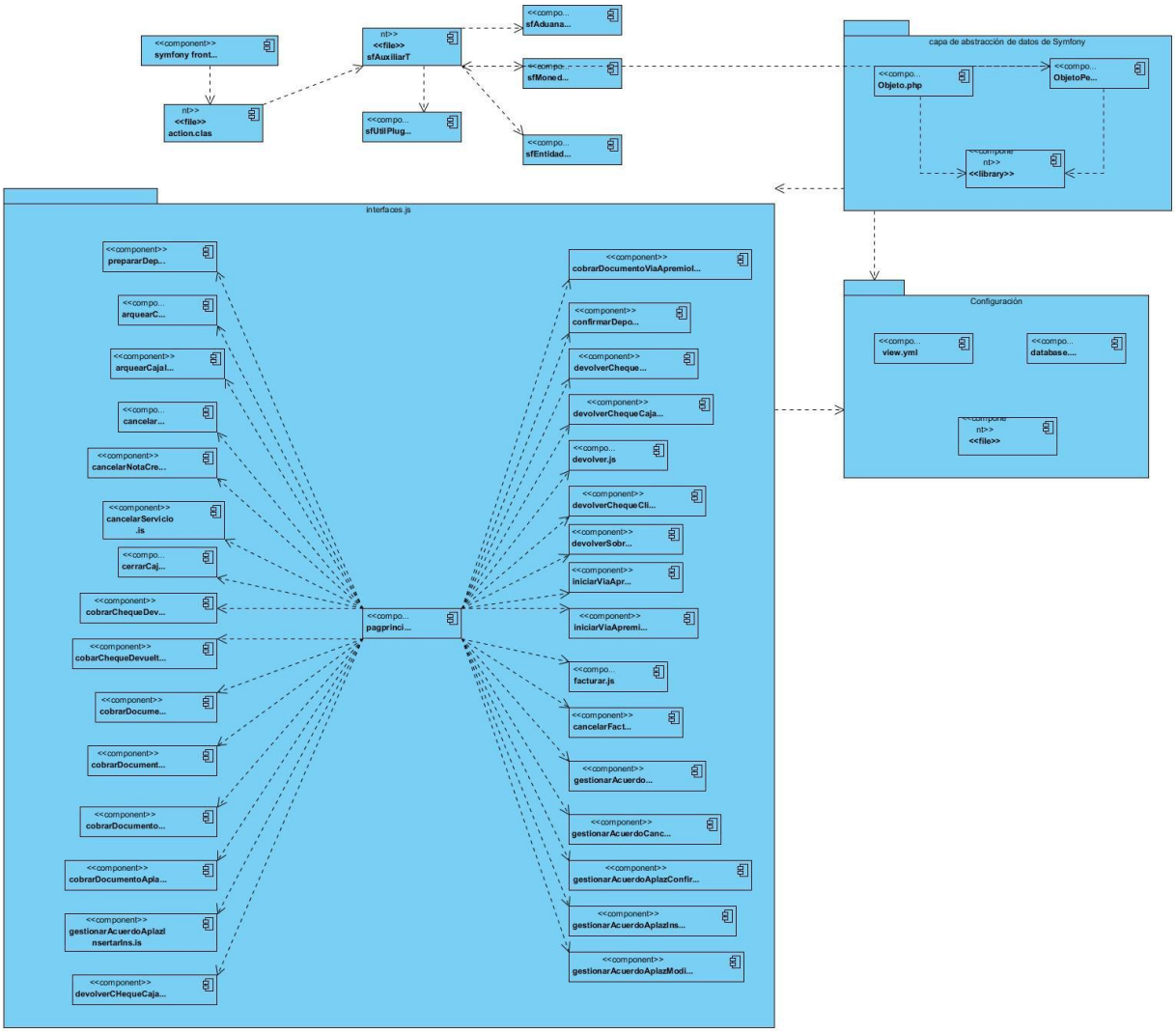
Si no se encuentran errores se realiza la operación de “Arquear Caja” y se notifica al usuario.



Asimismo se permite la realización de reportes en formato PDF a una serie de procesos que los requieren, a continuación se muestra un reporte realizado al caso de uso “Realizar Depósito”

2.13. Diagrama de Componentes

En el diagrama que muestra la Figura se presentan los principales componentes que se utilizan en el módulo. Se accede al sistema por medio del controlador frontal el cual según la petición realizada se comunica con la clase actions; esta clase interactúa con las interfaces que utilizan las librerías del Framework ExtJS y que son las encargadas de vincularse directamente con el usuario; también con las clases del negocio creadas por Propel y la clase sfAuxiliarTC que por medio de eventos es capaz de obtener información de los diferentes plugins y subsistemas. Los componentes de configuración que están presentes en todas las acciones realizadas en el módulo.



2.14. Diagrama de Despliegue

El diagrame de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos nodos que son representados (20). Se definirá una arquitectura cliente-servidor detallada de la siguiente manera. **Ver Figure 11**.

Cliente: Computador, la aplicación se ejecuta a través de un navegador internet instalado, en este caso se debe usar el Mozilla Firefox sobre cualquier sistema operativo (se sugiere GNU/Linux, en específico la distribución de Ubuntu para estaciones donde se conectarán dispositivos externos: Impresoras).

Servidor Web: Radica la lógica de negocio de la aplicación. Servidor Web Apache2. Utilizando el lenguaje PHP.

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

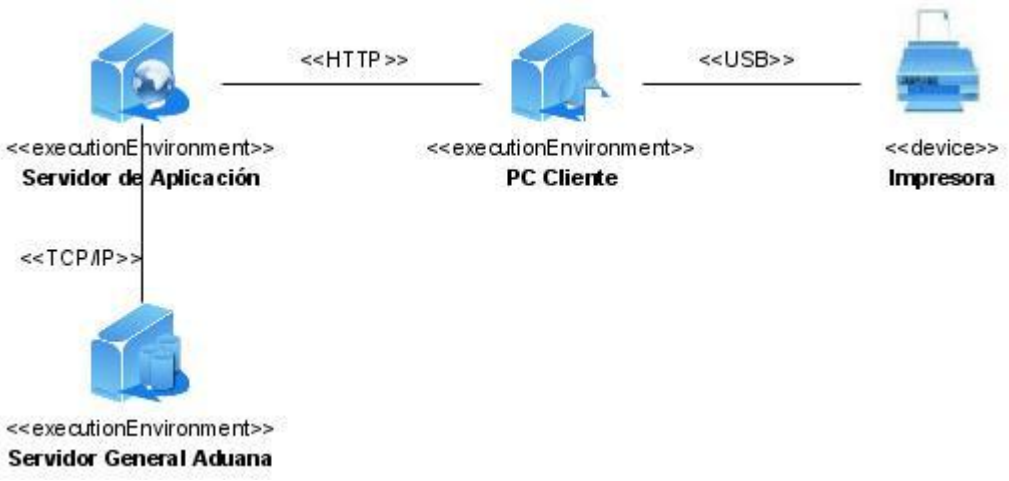


Figure 13-Diagrama de Despliegue

2.15. Conclusiones Parciales

Se obtuvieron como resultado, una serie de diagramas que fueron la base utilizada para dar cumplimiento de manera satisfactoria a la implementación del sistema. Se logró incorporar a la solución obtenida todos los requerimientos previstos con un bajo nivel de funcionalidades, minimizando la complejidad de implementación de las clases y atribuyendo la responsabilidad equitativamente, logrando también un alto por ciento de reutilización. Se aprovechó al máximo las posibilidades brindadas por los Frameworks utilizados, logrando un correcto tratamiento de errores y comunicación entre las capas previstas por el patrón arquitectónico MVC.

3. Capítulo III

3.1. Introducción

En el presente capítulo se realizaran una serie de pruebas para la validación de la solución obtenida. A través de una serie de casos de pruebas realizados, se comprobará si todos los requerimientos fueron cumplidos satisfactoriamente. Los avances tecnológicos conjuntamente con la competencia en el ámbito informático a nivel mundial exigen software de calidad y es por esto que la aplicación de pruebas de este tipo durante todas las etapas de desarrollo de un proyecto evita la ocurrencia de errores y defectos en el producto final.

3.2. Pruebas de Software

Las pruebas son la actividad en la cual se somete a un sistema o uno de sus componentes a una evaluación de los resultados que arroja en base a la ejecución de éste en condiciones específicas. Los Casos de Pruebas son un conjunto de entradas y condiciones que arrojan resultados esperados desarrollados con un objetivo en particular. Se conoce como error a la acción humana que produce un resultado incorrecto. El defecto es la manifestación de un error en el software. Es encontrado porque causa una falla, la cual es una desviación del servicio o resultado esperado. Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas (23).

Las fallas de software ocasionan graves pérdidas económicas y son más costosas de encontrar y reparar después de la construcción. También para evitar plazos y presupuestos incumplidos, insatisfacción del usuario, escasa productividad y mala calidad en el software producido y finalmente la pérdida de clientes.

3.3. Pruebas Funcionales o de Caja Negra

Se denominan pruebas funcionales o Functional Testing, a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de

pruebas con apoyo de algunos usuarios finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta el paso siguiente es el pase a producción. (23)

A este tipo de pruebas se les denomina también pruebas de comportamiento o pruebas de caja negra, ya que los testers o analistas de pruebas, no enfocan su atención a como se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas. (23) En esencia permite encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Dentro de la prueba de caja negra se incluyen las Técnicas de Pruebas que serán descritas a continuación:

Partición equivalente: Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. (24)

Análisis de valores límite: Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el análisis de valores límite lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el análisis de valores límite obtiene casos de prueba también para el campo de salida (24).

Para cada uno de los requisitos funcionales fueron definidos casos de pruebas, los cuales son los siguientes:

A continuación se muestra ejemplo de un caso de pruebas realizado para el requisito funcional Cancelar Servicio. Durante la puesta en práctica de los casos de pruebas realizados para cada requisito funcional, incluyendo el ejemplo a presentar, se identificaron varias no conformidades. También se evidenciaron en menor grado, inconformidades con el envío de peticiones AJAX, recomendaciones sobre notificaciones que se necesitaban realizar cuando se insertaban datos. Todas las inconformidades detectadas fueron resueltas en un cien por ciento, contribuyendo a la mejora del funcionamiento.

	Dominio	Particiones Válidas	Particiones no Válidas
Tipo Doc (entrada)	Texto	PV1: Lista desplegable	PN1: Empty
Operación (entrada)	Texto	PV1: Lista desplegable	PN1: Empty
Concepto (entrada)	Texto	PV1: Lista desplegable	PN1: Empty
Expediente (entrada)	Número de Expediente	PV1: numero.length() <= 11	PN1: Empty PN2: numero.lenght() >11
No. Doc (entrada)	Número de Documento	PV1: numero.length() <= 15	PN1: Empty PN2: numero.lenght() >15
Año (entrada)	Numérico	PV1: 1990 < año < 2050	PN1: 1990 > año > 2050
Monto (entrada)	Numérico	PV1: 0 < monto	PN1: 0 >= monto

	Particiones	Representante
Tipo Doc (entrada)	PV1: Lista desplegable PN1: Empty	Informe Recepción Null
Operación (entrada)	PV1: Lista desplegable PN1: Empty	Transferencia Null
Concepto (entrada)	PV1: Lista desplegable PN1: Empty	Aranceles Null
Expediente (entrada)	PV1: numero.length() <= 11 PN1: Empty PN2: numero.lenght() >11	K3104 Null A88113022547
No. Doc (entrada)	PV1: numero.length() <= 6 PN1: Empty PN2: numero.lenght() > 6	D3 Null Al1z3387
Año (entrada)	PV1: 1990 < año < 2050 PN1: 1990 > año > 2050	2012 1850
Monto (entrada)	PV1: 0 < monto PN1: Empty PN1: 0 >= monto	9600 Null 0

	Valore Válidos	Valore no Válidos
Tipo Doc (entrada)	Informe Recepción	Null
Operación (entrada)	Transferencia	Null
Concepto (entrada)	Aranceles	Null
Expediente (entrada)	K3104	Null, A88113022547
No. Doc (entrada)	D3	Null, Al1z3387
Año (entrada)	2012	1850
Monto (entrada)	9600	Null, 0

	C1	C2	C3	C4
Tipo Doc (entrada)	Informe Recepción	Informe Recepción	Null	Informe Recepción
Operación (entrada)	Transferencia	Null	Transferencia	Transferencia
Concepto (entrada)	Aranceles	Aranceles	Null	Aranceles
Expediente (entrada)	K3104	A88113022547	K3104	Null
No. Doc (entrada)	D3	Al1z3387	D3	Null
Año (entrada)	2012	2012	1850	2012
Monto (entrada)	9600	9600	0	9600
Salida Esperada	*"Deuda Cancelada"*	*-El campo "Operación" es requerido.* *-El "No. Doc" solo permite hasta 6 caracteres.* *-El "Expediente" solo permite hasta 11 caracteres.*	*-El campo "Tipo Doc" es requerido.* *-El campo "Concepto" es requerido.* *-El valor mínimo permitido es 1990.*	*-El campo "Expediente" es requerido.* *-El campo "No. Doc" es requerido.*

			-El "monto" debe ser > 1.	

Los demás Casos de Pruebas que se realizaron se encuentran especificados en el repositorio del proyecto.

Con la realización y puesta en marcha de las validaciones realizadas anteriormente se obtuvieron resultados positivos, se identificaron un número importante de no conformidades que al ser resueltas en un su totalidad ayudaron en gran medida al mejor funcionamiento de la solución obtenida.

3.4. Pruebas de caja blanca

La Prueba de Caja Blanca también se conoce como Prueba de Caja Transparente o de Cristal. Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento (24). Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación:

- **Prueba del Camino Básico:** Permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos. (24)

- **Prueba de Condición:** Ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método (24).

- **Prueba de Flujo de Datos:** Se seleccionan caminos de prueba de algún programa de acuerdo con la ubicación de las definiciones y los usos de las variables de programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez (24).

La prueba de caja blanca empleada en la solución desarrollada fue la prueba del camino básico, la cual permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

```
1719    public function executeCobrarViaApremio(sfWebRequest $request)
1720    {
1721        $parametros = $request->getPostParameters(); //1
1722        $documentos = json_decode($parametros['documento']); //1
1723        $instrumentos = json_decode($parametros['instrumento']); //1
1724        $idDoc = IcomDocumentoDeudaPeer::obtenerIdDocNumero($documentos[0]->numero); //1
1725
1726        $montoI = 0; //1
1727        $idInst = 0; //1
1728        try //2
1729        {
1730            $con = Propel::getConnection('ingreso'); //3
1731            $con->beginTransaction(); //3
1732
1733            $noRec = date('Yis'); //3
1734            $arrayIcomCobroForm = array(
1735                'idCaja' => 2,
1736                'cajero' => 'Alizee',
1737                'noDeposito' => '',
1738                'estado' => 'T',
1739                'noRec' => $noRec,
1740                'idAduana' => 1,
1741                'turno' => 'B',
1742                'fecha' => date('d-m-Y H:i:s'),
1743                'ingreso' => 'S',
1744                'idEntidad' => 1,
1745                ); //3
1746
1747            $IcomCobroForm = new IcomCobroForm(); //3
1748            $IcomCobroForm->updateObject($arrayIcomCobroForm); //3
1749            $IcomCobroForm->bind($arrayIcomCobroForm); //3
1750
1751            if ($IcomCobroForm->isValid()) //4
```

```
1752        {
1753            $IcomCobroForm->getObject()->save($con); //5
1754            $idCobro = $IcomCobroForm->getObject()->getIdCobro(); //5
1755
1756            foreach ($instrumentos as $i) //6
1757            {
1758                $montoI += $i->monto; //7
1759                $noInst = $i->numeroInstrumento; //7
1760
1761                if ($i->formaPago == '1') //8
1762
1763                    if ($this->executeComprobarCheque($i->numeroInstrumento, $i->moneda, $i->monto))
1764                        //9
1765
1766                        throw new Exception(' El instrumento ' . $i->numeroInstrumento .
1767                            ' se ha utilizado anteriormente.'); //10
1768
1769                $arrayInstrumento = array(
1770                    'fechaBanco' => $i->fechaBanco,
1771                    'pagadoPor' => '',
1772                    'idCobro' => $idCobro,
1773                    'idMoneda' => $i->moneda,
1774                    'idEstadoInstrumento' => 1,
1775                    'idConcepto' => $i->concepto,
1776                    'idFormaPago' => $i->formaPago,
1777                    'noInstrumentoPago' => $i->numeroInstrumento,
1778                    'monto' => $i->monto,
1779                    ); //11
1780
1781                $IcomInstrumentoForm = new IcomInstrumentoForm(); //11
1782                $IcomInstrumentoForm->updateObject($arrayInstrumento); //11
1783                $IcomInstrumentoForm->bind($arrayInstrumento); //11
1785                if ($IcomInstrumentoForm->isValid()) //12
1786                {
1787                    $IcomInstrumentoForm->getObject()->save($con); //13
1788                    $idInst = $IcomInstrumentoForm->getObject()->getIdInstrumento(); //13
1789
1790                    $arrayDocInstForm = array(
1791                        'monto' => $documentos[0]->importe,
1792                        'plazo' => 0,
1793                        'idDocDeuda' => $idDoc,
1794                        'idInstrumento' => $idInst,
1795                        ); //13
1796
1797                    $IcomDocInstForm = new IcomDocInstForm(); //13
1798                    $IcomDocInstForm->updateObject($arrayDocInstForm); //13
1799                    $IcomDocInstForm->bind($arrayDocInstForm); //13
1800
1801                    if ($IcomDocInstForm->isValid()) //14
1802
1803                        $IcomDocInstForm->save($con); //15
1804
1805                } else //16
1806                {
1807                    throw new Exception('No se Inserta Instrumento.'); //17
1808                }
1809            }
1810            $difMonto = $documentos[0]->importe - $montoI; //18
1811            if (($difMonto * -1) > 1) //19
1812
1813                if (!IcomNotaCreditoPeer::existeNotaCredito($noInst)) //20
1814                {
1815                    $difMonto *= -1; //21
1816                    $noNota = $instrumentos[0]->numeroInstrumento; //21
1817                    $idAduanaDoc = IcomDocumentoPeer::obtenerDocNumero($documentos[0]->numero); //21
```

```
1818
1819                        $this->executeGenerarNotaCredito($noNota, date('d-m-Y'), $difMonto, 0, $idInst,
1820                            $idAduanaDoc[0]->getIdAduana(), $difMonto); //22
1821                    } else //23
1822                    {
1823                        IcomNotaCreditoPeer::actualizarNotaCredito($instrumentos[0]->numeroInstrumento,
1824                            $difMonto); //24
1825                    }
1826
1827                    if (!IcomDocumentoDeudaPeer::cancelarDeudaViaApremio($documentos[0]->numero))
1828                        //25
1829                    {
1830                        throw new Exception('No se puede cancelar la deuda.'); //26
1831                    } else //27
1832                    {
1833                        $con->commit(); //28
1834                        return $this->renderText("{'success' : 'true', 'mensaje' : 'Deuda cancelada.'}"); //28
1835                    }
1836            }
1837
1838            throw new Exception('No se puede cancelar las deuda, error encontrado'); //29
1839
1840        }
1841        catch (exception $ex) //30
1842        {
1843            $con->rollBack(); //31
1844            throw new Exception($ex->getMessage()); //31
1845
1846        }
1847    }
```

Figure 14- Código del método executeCobrarViaApremio (sfWebRequest $request)

A continuación se muestra el grafo de flujo asociado a la funcionalidad.

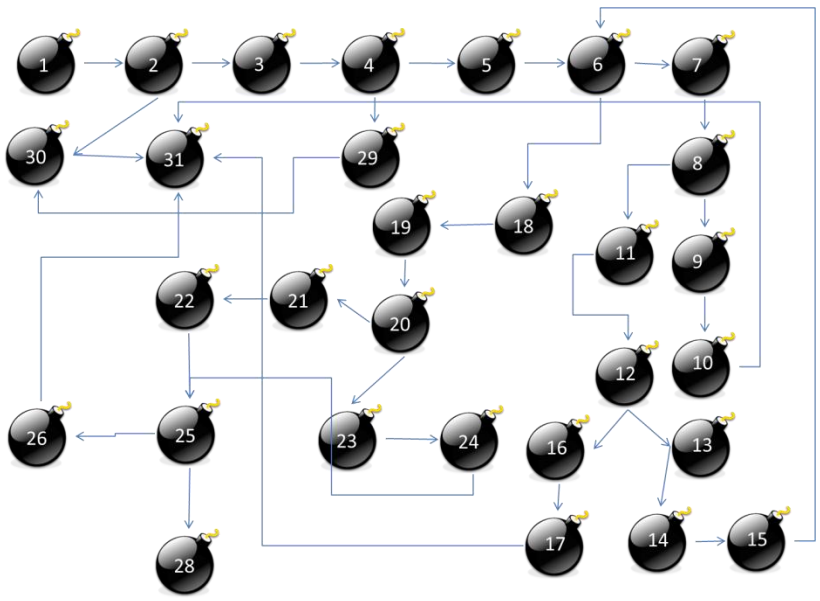


Figure 15- Grafo asociado al algoritmo executeCobrarViaApremio (sfWebRequest $request)

Cálculo de la complejidad ciclomática a partir de un segmento de código

Luego de haber construido el grafo, se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas utilizadas para el análisis de complejidad del algoritmo, las cuales tienen que arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto. Fórmulas para calcular complejidad ciclomática:

1. V (G) = (A - N) + 2.
   Donde “A” es la cantidad de Aristas y “N” la cantidad de Nodos.
   V (G) = (35 - 30) + 2.
   V (G) = 7.
2. V (G) = P + 1.
   Siendo “P” la cantidad de Nodos predicados (son los nodos de los cuales parten dos o más aristas).
   V (G) = 6 + 1 V (G) = 7.
3. V (G) = R.
   Donde **“R”** representa la cantidad de regiones en el grafo.
   V (G) = 7.

A partir del valor de la complejidad ciclomática obtenemos el número de caminos independientes, que nos dan un valor límite para el número de pruebas que tenemos que diseñar. En el ejemplo, el número de caminos independientes es 7, y los caminos independientes son:

- 1-2-30-31.
- 1-2-3-4-29-30-31.
- 1-2-3-4-5-6-7-8-9-10-30-31.
- 1-2-3-4-5-6-7-8-11-12-16-17-30-31.
- 1-2-3-4-5-6-7-8-11-12--13-14-15-6-18-19-20-21-22-25-28.
- 1-2-3-4-5-6-7-8-11-12-13-14-15-6-18-19-20-23-24-25-26-30-31.
- 1-2-3-4-5-6-7-8-11-12-13-14-15-6-18-19-20-21-22-25-26-30-31.

Escenario	Descripción	No. Instrumento	Monto	Fecha Banco	Moneda	Forma de Pago	Concepto	Número	Año	Tipo Doc	Operación	Aduana	Respuesta del sistema	Flujo central
EC 1 Cobrar Vía Apremio	Permitirá el pago de documentos que se encuentren en vía de apremio	V	V	V	V	V	V	V	V	V	V	V	Muestra una notificación "Cobro Realizado".	1- Se selecciona la opción "Cobrar Vía Apremio" del submenú "Documentos" en la interfaz principal. 2- Se inserta el instrumento de pago que se utilizará para saldar la deuda adquirida por los diferentes conceptos. 3- Se busca el documento que se desea pagar, especificando en el filtro de búsqueda (número, año, tipo, operación, aduana). 4- Se envía la solicitud de pago de deuda. 5- Se recibe una respuesta del servidor con el resultado obtenido por las acciones previas.
		K3104	1800	01/06/2012	MN	Cheque	Arancelario	46	2012	InformeRecepcion	TRANSFERENCIA	ADUANA GENERAL DE LA		
		I	V	V	V	V	V	V	V	V	V	V	Muestra un error "El instrumento 'K3104' ya ha utilizado anteriormente."	
		K3104	1800	01/06/2012	MN	Cheque	Arancelario	44	2012	InformeRecepcion	TRANSFERENCIA	ADUANA GENERAL DE LA		
EC 1.1 Insertar Instrumentar de Pago.	Permitirá insertar instrumentar de pago para saldar la deuda.	V	V	V	V	V	V	N/A	N/A	N/A	N/A	N/A	Agrega a la grid "Instrumento de Pago" los datos del Instrumento deseado.	
		K3104	1800	01/06/2012	MN	Cheque	Arancelario							
		I	V	V	V	V	V	N/A	N/A	N/A	N/A	N/A	Muestra un error "El "No. Instrumento" solo permite hasta 15 caracteres	
		AI123881130225 47	5000	*******	MN	Cheque	Arancelario							
EC 1.2 Buscar Documento	Permitirá realizar búsquedas de personas a partir de criterios de búsqueda más avanzados.	N/A	N/A	N/A	N/A	N/A	N/A	V	V	V	V	V	Se carga en el grid el Documento que cumple con el filtro.	
								46	2012	InformeRecepcion	TRANSFERENCIA	ADUANA GENERAL DE LA REPUBLICA		
		N/A	N/A	V	N/A	N/A	N/A	I	V	V	V	V	Nuevo muestra ningún documento ya que no existe un documento que cumpla con el filtro y además pertenece a una deuda.	
								48	2012	InformeRecepcion	TRANSFERENCIA	ADUANA GENERAL DE LA REPUBLICA		
		N/A	N/A	N/A	V	N/A	N/A	V	I	V	V	V	Nuevo muestra ningún documento ya que no existe un documento que cumpla con el filtro y además pertenece a una deuda.	
								46	2013	InformeRecepcion	TRANSFERENCIA	ADUANA GENERAL DE LA REPUBLICA		

Los demás Casos de Pruebas que se realizaron se encuentran especificados en el repositorio del proyecto.

Se presentaron además, otra serie de casos de pruebas para los demás requisitos. A continuación se enumeran algunos de los mismos:

1. **Requisito Cobrar Acuerdo de Aplazamiento**:
   - EC 1.1 Insertar instrumento de pago.
   - EC 1.2 Buscar documentos aplazados.
2. **Requisito Cobrar Vía Apremio:**
   - EC 2.1 Insertar instrumento de pago.
   - EC 2.2 Buscar documento en vía de apremio.
3. **Iniciar Vía Apremio**:
   - EC 3.1 Documentos en Vía de Apremio.
4. **Cancelar Mora**:
   - EC 4.1 Cancelar mora.

5. **Arquear Caja**:
   - EC 5.1 Adicionar cheques.
   - EC 5.2 Adicionar efectivo.
6. **Devolver Cheque Caja**:
   - EC 6.1 Devolver cheque caja.
7. **Devolver Cheque Cliente:**
   - EC 7.1 Devolver cheque cliente.
8. **Facturar:**
   -EC 8.1 Facturar.
9. **Gestionar Acuerdo:**
   -EC 9.1 Insertar Acuerdo.
   -EC 9.2 Modificar Acuerdo.
   -EC 9.3 Cancelar Acuerdo.
   -EC 9.4 Confirmar Acuerdo.
10. **Cancelar Nota Crédito:**
    -EC 10.1 Nota Crédito.
11. **Cancelar Servicio:**
    -EC 11.1 Cancelar Servicio.
12. **Cobrar Cheque Devuelto:**
    -EC Datos Cheque Devuelto.
    -EC Datos Cheque Nuevo.
13. **Preparar Depósito:**
    -EC Recibos Pendientes a Depositar.

Con la aplicación de los casos de prueba expuestos anteriormente se comprobó que el flujo de trabajo de las funcionalidades es correcto, ya que se probó que cada sentencia es ejecutada al menos una vez, cumpliéndose así las condiciones de la prueba.

3.5. Conclusiones Parciales

En el capítulo se realizó la constatación del sistema construido mediante pruebas funcionales, fundamentalmente a través de las pruebas de caja blanca y caja negra; incluyendo valoraciones obtenidas por la Aduana General de la República de Cuba. Se construyeron Casos de Pruebas, y sus escenarios correspondientes, que brindaron resultados insatisfactorios, dando paso a la supresión de los problemas, favoreciendo de esta manera a la mejora de la solución.

3.6. Conclusiones Generales

En el presente trabajo se trataron una serie de problemas que presenta la Aduana General de la República de Cuba, referente a los procedimientos de gestión de Ingresos por la Vía Comercial que se llevan a cabo por el SUA.

Luego de realizado un estudio del estado del arte y una profunda investigación sobre las buenas prácticas llevadas a cabo en el mundo, se logró dar cumplimiento a los objetivos planteados, logrando contribuir a la mejora del rendimiento, mantenimiento y funcionamiento de los procesos para la gestión de ingresos en la AGR.

Como parte de la solución obtenida se lograron generar una serie de artefactos, que fueron la base para la posterior implementación del sistema, destacando el diagrama de secuencia orientado a actividades, solución obtenida por el proyecto, para agilizar la implementación del módulo.

Se utilizó, en la creación del sistema, el patrón arquitectónico MVC, trayendo como resultado, que quedara separada la arquitectura en tres niveles distintos e independientes, logrando facilitar el mantenimiento en caso que sea necesario. También se incorporó a la solución, una serie de nuevas funcionalidades que no se tenían concebidas por la AGR hasta el momento, contribuyendo de esta manera la agilización de los procedimientos de Ingresos por la vía Comercial, dando paso, a la mejora del rendimiento y funcionamiento, según pruebas y valoraciones que se llevaron a cabo por la AGR.

Se logró realizar pruebas funcionales al sistema, a través de casos de pruebas realizados por los especialistas del proyecto, brindando inconformidades que propiciaron la modificación de actividades, que aportaron como resultado la culminación de un producto cien por ciento utilizable.

Bibliografía

1. **LANSA.** Lansa. *Lansa.* [En línea] enero de 2012. [Citado el: 21 de febrero de 2012.] http://www.lansa.com/es/products/integration.htm.

2. Agenda Magna. *Agenda Magna.* [En línea] 27 de febrero de 2009. [Citado el: 15 de abril de 2012.] http://agendamagna.wordpress.com/2009/02/26/glosario-de-terminos-aduaneros/.

3. **Aduana General de la República de Cuba.** Aduana General de la República de Cuba. *Aduana General de la República de Cuba.* [En línea] 2011. [Citado el: 20 de enero de 2012.] http://www.aduana.co.cu/.

4. **Sybase Iberia, S.L.** sybase. *sybase.* [En línea] 2012. [Citado el: 21 de febrero de 2012.] http://www.sybase.es/products/dataintegration.

5. **Arellano, Ana Alejandra Febres.** GestioPolis.com. *GestioPolis.com.* [En línea] marzo de 2004. [Citado el: 15 de enero de 2012.] http://www.gestiopolis.com/canales2/economia/sidunea.htm.

6. **Billy, Carlos Reynoso.** *Introducción a la Arquitectura de Software.* Buenos Aires : Universidad de Buenos Aires, 2004, Vol. I., 2004.

7. **Barry, Chris y Lang, Michael.** *A Survey of Multimedia and Web Development Techniques and Methodology Usage.* April-June 2001. s.l. : IEEE Multimedia, 2001. págs. 52-60.

8. **Melisa.** Buenas tareas. *Buenas tareas.* [En línea] 2010. [Citado el: 25 de enero de 2012.] http://www.buenastareas.com/ensayos/Arquitectura-Del-Software/922875.html.

9. **Potencier, Fabien y Zaninotto, François.** *Symfony 1.1, la guía definitiva.* s.l. : librosweb.es, 2009. http://www.librosweb.es/symfony_1_1.

10. **Corzo, Giancarlo.** [En línea] 22 de 10 de 2008. [Citado el: 14 de 3 de 2012.] Socio y Gerente de desarrollo de nuevos productos en Antartec. Con experiencia en desarrollo de interfaces para Web, Móvil (en especial para iPhone). Me desempeño además como Profesor en la universidad PUCP asesorando a los futuros tesistas.. http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

11. **Visconti, Marcello y Astudillo, Hernán.** *Fundamentos de Ingeniería de Software.* Santa María : Universidad Técnica Federico Santa María.

12. **Angelfire.** http://geektheplanet.net. *http://geektheplanet.net.* [En línea] 11 de 5 de 2011. [Citado el: 17 de 2 de 2012.] http://geektheplanet.net/5462/patrones-gof.xhtml.

13. **Symfony.es.** Symfony.es. *Symfony.es.* [En línea] 13 de marzo de 2011. [Citado el: 14 de marzo de 2012.] http://www.symfony.es/categoria/propel/page/2/.

14. **Copyright © 2001-2011 The PHP Group.** PHP. *PHP.* [En línea] 2011. [Citado el: 13 de Diciembre de 2011.] http://www.php.net/.

15. Cascading Style Sheets. *Cascading Style Sheets.* [En línea] 20 de abril de 2012. [Citado el: 20 de abril de 2012.] http://www.w3.org/Style/CSS/.

16. **Pérez, Javier Eguíluz.** *Introducción a JavaScript.* s.l. : www.librosweb.es, 2008.

17. **Visual Paradigm.** Visual Paradigm. *Visual Paradigm.* [En línea] [Citado el: 13 de Diciembre de 2011.] http://www.visual-paradigm.com/.

18. **Martínez, Jenni Manso.** *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* Ciudad de La Habana : s.n., 2010. tesis.

19. **Oracle Corporate.** Oracle. [En línea] 2011. [Citado el: 13 de Diciembre de 2011.] http://www.oracle.com/index.html.

20. **EcuRed.** Enciclopedia Cubana. [En línea] 2012. [Citado el: 25 de febrero de 2012.] http://www.ecured.cu/index.php/Diagrama_de_despliegue..

21. **mitecnologico.com.** mitecnologico.com. *mitecnologico.com.* [En línea] [Citado el: 15 de 2 de 2012.] http://www.mitecnologico.com/Main/AplicacionMetricasParaEvaluacionDise%F1o.

22. **Departamento de Soluciones para la Aduana, CEIGE.** *Propuesta de un Estándar de Codificación.* 2011.

23. **B., Ing. Alexander Oré.** Calidad y Software. [En línea] NazcaSoft.com, 2009. [Citado el: 25 de 2 de 2012.] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

24. **Pressman, Roger S.** *Ingeniería de Software, Un enfoque Práctico.* 2005.

25. **García, Gonzalo Eduardo Castillo.** monografias. *monografias.* [En línea] 2 de marzo de 2008. [Citado el: 29 de noviembre de 2011.] http://www.monografias.com.

26. **PORTILLA, Ing. MSc. Esp. YEZID ORLANDO GARCÍA.** slideshare.net. *slideshare.net.* [En línea] 14 de Mayo de 2010. [Citado el: 1 de Diciembre de 2011.] http://www.slideshare.net.

27. **Eguiluz, Javier.** Symfony.es. *Symfony.es.* [En línea] 2011. [Citado el: 13 de Diciembre de 2011.] http://www.symfony.es/.

28. **Integración de aplicaciones empresariales (EAI).** http://es.kioskea.net. *http://es.kioskea.net.* [En línea] 16 de octubre de 2008. [Citado el: 21 de febrero de 2012.] http://es.kioskea.net.

29. **Sæther Bakken, Stig, y otros, y otros.** *Manual de PHP.* 2001.

30. **Tedeschi, Nicolás.** http://msdn.microsoft.com. *http://msdn.microsoft.com.* [En línea] [Citado el: 21 de febrero de 2012.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

31. **Sencha.** Sencha.com. *Sencha.com.* [En línea] 2011. [Citado el: 15 de febrero de 2012.] http://www.sencha.com/products/extjs.

32. **http://javascript.about.com.** http://javascript.about.com. *http://javascript.about.com.* [En línea] 2012. [Citado el: 21 de febrero de 2012.] http://javascript.about.com.

33. **Arias, Alain Rodríguez.** *Modelo de Desarrollo Dpto Soluciones Aduana.* Habana : s.n., 2012.

34. **Oracle Corporation.** Portal del IDE Java de Código Abierto. *Portal del IDE Java de Código Abierto.* [En línea] 2010. [Citado el: 6 de febrero de 2012.] http://netbeans.org/index_es.html.