Universidad de las Ciencias Informáticas
Facultad 3

Título: Componente Fórmula para el Sistema de Gestión Integral de la Aduana (GINA).

Trabajo de Diploma para optar por el título de

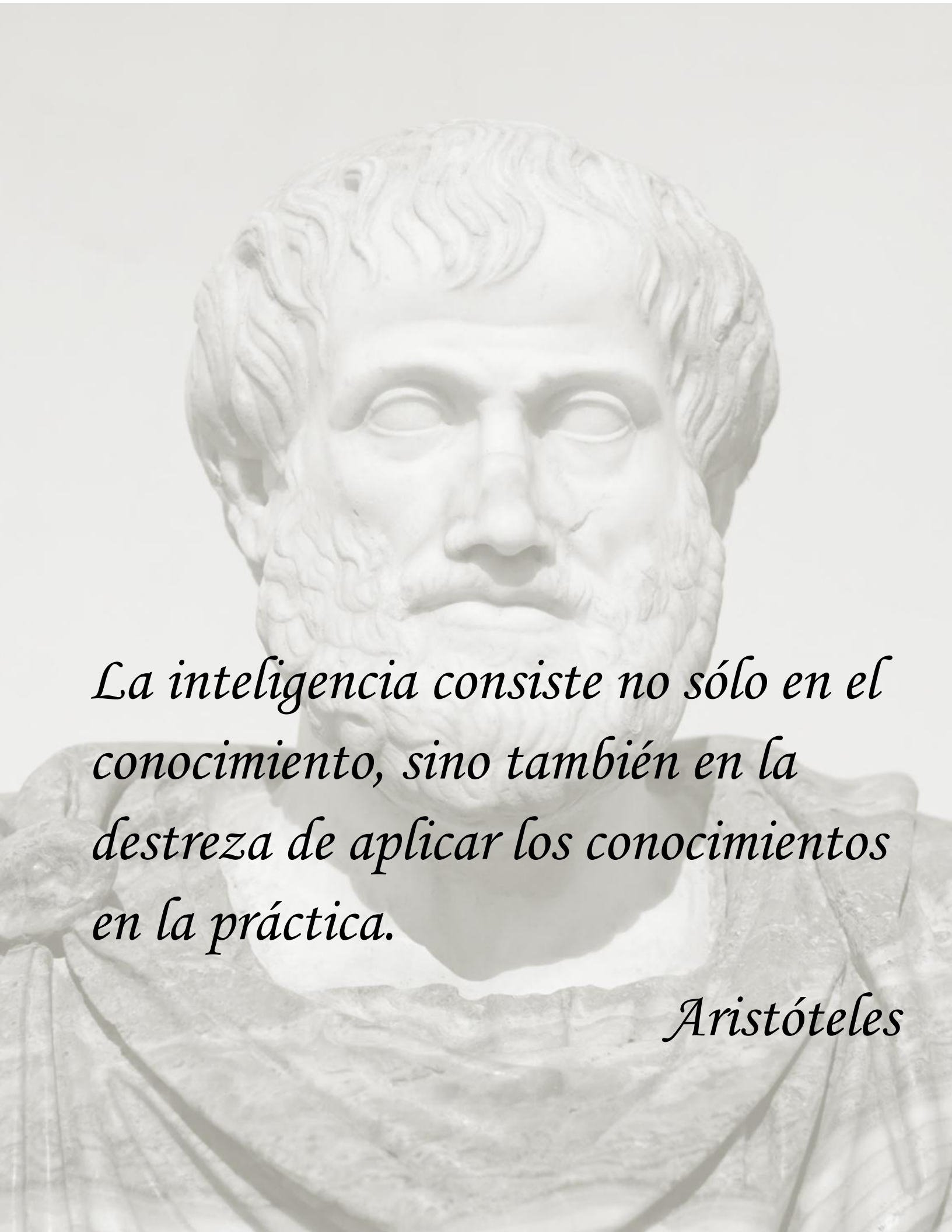
Ingeniero en Ciencias Informáticas

Autor(es): Saily de la Caridad Panuncia Ridel

Tutor(es): Ing. Boris Enrique Ruiz Guerra

Co-tutor: Ing. Julio César Bravo Rodríguez

Junio de 2012



La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.

Aristóteles

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Saily de la Caridad Panuncia Ridel

_____

Ing. Boris Enrique Ruiz Guerra

_____

DATOS DE CONTACTO

Ing. Boris Enrique Ruiz Guerra

Graduado de Ingeniero en Ciencias Informáticas por la UCI. Recién graduado.

AGRADECIMIENTOS

Quiero agradecer en primer lugar a mi madre por apoyarme en mi carrera en los momentos buenos y más aún en los malos, a mi familia en general, a los profesores y estudiantes del proyecto Aduana que me han brindado su ayuda y apoyo, en especial a mi tutor Boris, a Adrián, Ricardo, Lázaro, Daniel, Leodán, Maurice, Fernando que siempre confió en mí y me dijo que siguiera adelante. A la Revolución Cubana por darme la posibilidad de estar donde hoy estoy.

A todos muchas gracias

DEDICATORIA

Quisiera dedicar este trabajo diploma a todos los que han confiado en mí y en especial a mi mamá.

RESUMEN

La Aduana General de la República (AGR) pretende asegurar 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 nacional e internacional de viajeros, mercancías y medios de transporte. Esta, en colaboración con la Universidad de las Ciencias Informáticas (UCI), ha decidido informatizar todos sus procesos, lo cual se ve evidenciado en el hecho de que actualmente desarrollan el Sistema de Gestión Integral de Aduana (GINA).

El GINA actualmente presenta algunas dificultades con respecto al cálculo de aranceles que exige la aduana como impuestos a los productos extranjeros que son importados al país, ya que las fórmulas para realizar estos cálculos se encuentran definidas en el código fuente de los subsistemas que las necesitan, lo que dificulta la gestión de las mismas teniendo que ir directamente al código fuente de la aplicación, exigiendo mayor tiempo y conocimiento informático, por ello, el objetivo del presente trabajo diploma es desarrollar un componente para la centralización y gestión de las mismas.

Para el desarrollo de la solución se utilizaron las herramientas y tecnologías definidas en el Departamento de Soluciones para la Aduana, tales como: Symfony[1] y Ext js[2] como marcos de trabajo, PHP[3] como lenguaje del lado del servidor, NetBeans[4] como entorno de desarrollo, entre otras.

La solución proveerá al GINA de un componente que permita la centralización y gestión dinámica de las fórmulas empleadas para el cálculo de aranceles en la AGR, garantizando que no haya que ir al código a realizar modificaciones en las mismas.

PALABRAS CLAVE: GINA, fórmulas, aranceles

[1] Marco de trabajo o framework.
[2] Marco de trabajo o framework que permite crear interfaces y páginas web dinámicas.
[3] De sus siglas en inglés Hypertext Preprocessor, es un lenguaje de programación.
[4] Herramienta usada para escribir, compilar y ejecutar programas.

Contenido

INTRODUCCIÓN

La Organización Mundial de Aduanas (OMA) es el organismo rector a nivel internacional de algunas de las regulaciones aduaneras y emite los lineamientos generales bajo los cuales deben operar las autoridades de este ramo en todo el mundo. Esta organización agrupa a 173 Administraciones de Aduanas en todo el mundo y en el año 1974 estableció el convenio de Kyoto para la simplificación y armonización de los procedimientos aduaneros que fue revisado en el año 1999 y liberado para su firma en el año 2006. La Aduana cubana suscribió en 1995 este convenio y en 2009, ratificó el Convenio de Kyoto Revisado, siendo Cuba el único país signatario de Latinoamérica. (1)En dicho convenio se sugieren prácticas para facilitar el comercio y naturalización de mercancías sin afectar en la medida de lo posible la seguridad e integridad nacional de cada nación.

El 5 de febrero de 1963 se crea la Aduana General de la República de Cuba (AGR) con la misión de garantizar la seguridad y protección de la sociedad socialista y de la economía nacional. Además la AGR pretende asegurar 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 nacional e internacional de viajeros, mercancías y medios de transporte. Exige el pago de aranceles, es decir, de impuestos a los productos extranjeros que son importados al país, cálculo que se realiza a través de fórmulas que tienen la característica de ser gestionadas en cualquier momento y en dependencia de determinadas circunstancias, además de la reutilización de las mismas en los diferentes procesos que conforman el complejo mecanismo aduanero llevado a cabo en las fronteras de Cuba.

La importancia de la aduana se evidencia en que con una eficiente administración de esta, los gobiernos tendrían claramente definidos sus objetivos frente a la globalización y el continuo crecimiento del comercio internacional, ya que la interdependencia económica existente en las relaciones entre Estados necesita de políticas aduaneras, que para garantizar su efectividad debe contar con un personal altamente calificado y con una tecnología apropiada. (2) Es por esto que la AGR intenta elevar continuamente la excelencia de sus servicios y actualmente es uno de los organismos más importantes en la economía del país.

A partir de lo establecido en el convenio de Kyoto, existe en el entorno aduanero internacional una fuerte tendencia hacia la informatización de los procesos de despacho de mercancías y pasajeros, razón por la

cual un gran peso en la modernización de la AGR lo ha tenido la aplicación 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. Lo anterior se ve evidenciado en el hecho de que la Universidad de las Ciencias Informáticas en colaboración con el Centro de Automatización para la Dirección y la Información de la AGR (CADI) desarrollan el Sistema de Gestión Integral de Aduanas (GINA).

El GINA es un sistema de gestión de procesos de despacho aduanal de fabricación nacional y ajustado a las necesidades del país que incluye informatización de los procesos de Despacho de Mercancías, gestión de Medios de Transporte Internacional y contenedores, gestión de depósitos y almacenes bajo control aduanero. De igual manera esta solución soporta intercambio electrónico de información relevante para el funcionamiento de las aduanas.

GINA se divide en subsistemas como Medios de Transporte Internacional, Despacho No Comercial, Recursos Humanos, Configuración, Despacho Comercial, destacándose este último en el cálculo de aranceles debido a que se encarga de la recepción en formato electrónico de las Declaraciones de Mercancías y los documentos complementarios asociados al flujo del despacho de mercancías.

Actualmente en el GINA la implementación de las fórmulas se encuentran definidas en el código fuente de cada subsistema que las necesite, lo que trae como desventaja, que cada subsistema tenga que repetir el código que las implementa; además en caso de existir cambios en estas, el mantenimiento tendría que realizarse directamente en el código, provocando que el procedimiento exija mayor tiempo y conocimiento, así como la participación activa de personal calificado.

Por lo antes expuesto se plantea como **problema a resolver** del presente trabajo diploma: ¿Cómo centralizar y gestionar dinámicamente las fórmulas para el cálculo de aranceles en la Aduana General de la República?

Para dar solución al problema planteado se tiene como **objeto de estudio**: Sistemas Informáticos para la Gestión de Fórmulas Matemáticas.

Definiendo como **objetivo general**: Desarrollar el Componente Fórmula del Sistema de Gestión Integral de la Aduana para centralizar la gestión de las ecuaciones utilizadas en el cálculo de aranceles en la

Aduana General de la República, siendo el **campo de acción:** Componente Fórmula para el cálculo de aranceles en la Aduana General de la República.

Para cumplir con el objetivo y lograr una solución adecuada a la problemática especificada, se plantean las siguientes **tareas a cumplir**:

- Identificación de los requisitos de usuario.
- Estudio el estado del arte.
- Identificación de requisitos cumplimentados en otro software de intercambio.
- Desarrollo del modelo de especificación de requisitos.
- Evaluación de las herramientas a utilizar.
- Evaluación de las técnicas y patrones de diseño de aplicaciones.
- Evaluación de los estándares de codificación definidos en el proyecto.
- Elaboración del modelo de dato del Componente Fórmula.
- Elaboración del modelo de diseño de clases del Componente Fórmula.
- Elaboración de los diagramas de diseño de la solución del Componente Fórmula.
- Elaboración del diagrama de componentes para el Componente Fórmula.
- Implementación de la solución.
- Definición de las pruebas de validación.
- Realización de las pruebas de validación.

El presente trabajo está estructurado de la siguiente manera:

- Capítulo 1. Fundamentación Teórica: en este capítulo se realizará el estudio del estado del arte, se explicarán los conceptos esenciales vinculados al problema, los lenguajes, metodologías y herramientas que se emplearán en la construcción de la solución.

- Capítulo2. Descripción de la solución: en este capítulo se realizará el análisis y diseño de la solución propuesta y para ello se generarán todos los artefactos necesarios y definidos por el Departamento de Soluciones para la Aduana.

- Capítulo 3: Implementación y Prueba: en este capítulo se realizará el modelo de implementación definiendo los estándares de codificación, patrones arquitectónicos, entre otros, además de las pruebas realizadas al sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el siguiente capítulo se realiza una descripción de los elementos que fundamentan el contenido de la investigación y se realiza un estudio de los sistemas informáticos existentes para la gestión de fórmulas. Además se describen las metodologías, lenguajes, técnicas y herramientas que serán usadas para el desarrollo de la solución.

1.2 Conceptos

Aduana

Una aduana es una oficina pública, establecida generalmente en las costas y fronteras, para registrar, en el tráfico internacional, los géneros y mercaderías que se importan o exportan, y cobrar los derechos que adeudan. (3)

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 encomiendan. (4) Su objetivo es facilitar el comercio, hacerlo más práctico, expeditivo y funcional. (5)

Actualmente es posible afirmar que las aduanas constituyen el principal organismo ejecutor de la política de comercio internacional del Estado en cuanto al control y cumplimiento de las regulaciones económicas, administrativas, contractuales, restrictivas y tributarias que afectan los términos físicos del intercambio.

En Cuba es cobrado el arancel de Aduanas, el cual es un instrumento que cumple una importante función en la realización e incremento de las relaciones económicas internacionales y, consecuentemente, coadyuva a la expansión del comercio exterior. Además es un medio de ejecución de la política comercial y de control y fiscalización de las importaciones de mercancías, que favorece el desarrollo de la economía nacional.

Arancel

Un arancel es un impuesto aplicable en el comercio exterior (6) que el gobierno exige a los productos extranjeros con el objeto de elevar su precio de venta en el mercado interno, y así proteger los productos nacionales para que no sufran la competencia de bienes más baratos. Un arancel tiende a elevar el precio, a reducir las cantidades consumidas e importadas y a incrementar la producción nacional. Los ingresos arancelarios suponen una transferencia al Estado por parte de los consumidores, ya que éstos no reciben nada a cambio de aquél, pero no representan un costo para la sociedad, ya que el Estado los utiliza en su presupuesto de gastos.

En la AGR los aranceles son calculados mediante el uso de fórmulas matemáticas.

Cálculo

El cálculo es un cómputo, cuenta o investigación que se hace de algo por medio de operaciones matemáticas. Existen distintos tipos de cálculo, como son: (7)

- Algebraico: se hace con letras que representan las cantidades, aunque también se empleen algunos números.
- Aritmético: se hace con números exclusivamente y algunos signos convencionales.
- Diferencial: parte de las matemáticas que opera con las diferencias infinitamente pequeñas de las cantidades variables.
- Integral: trata de obtener una función a partir de su derivada.
- Infinitesimal: conjunto de los cálculos diferencial e integral.
- Proposicional: estudia las estructuras deductivas de las implicaciones lógicas y sus relaciones axiomáticas.
- Prudencial: se hace con aproximación y sin buscar la exactitud.

Fórmula

Una fórmula es un método práctico de resolver un asunto, brindar instrucciones o expresar una operación en el ámbito científico.

En la matemática, la fórmula es la modalidad sintáctica formal que expresa una proposición de esta ciencia. Las fórmulas matemáticas más comunes son las que se componen de símbolos constantes, símbolos de funciones y símbolos de relación. (8)

1.3 Sistemas informáticos para la gestión de fórmulas

Internacionales

Software de liquidación de haberes que ofrece DDS (9)

Este software de liquidación de haberes es un sistema que realiza el cálculo de sueldos en base a datos previamente cargados en los legajos del personal (sueldo básico, cantidad de hijos, antigüedad, etc.), sumándole las novedades del mes / quincena que se liquida (llegadas tarde, horas extra, bonificaciones y premios, etc.). Una vez dada la orden, el software obtiene los totales y emite los recibos de sueldos además de los listados correspondientes para el pago a obras sociales, jubilaciones, sindicatos, seguros, etc. (9)

Software de gestión industrial Sistema Isis MRP Manager

Es una solución que le informa al usuario lo que se está gastando ahora en su fábrica, y le ayuda a planear la fabricación a futuro. (10)

Nacionales

Intérprete de ecuaciones del módulo de auditoría del Sistema de Gestión Integral CEDRUX

El editor de ecuaciones, o calculadora, es una herramienta de uso general, que está disponible para ingresar ecuaciones, crear fórmulas y expresiones dentro de la aplicación web.

La siguiente tabla muestra cada una de las características con que debe cumplir la solución y con cuáles de ellas cumplen los sistemas estudiados.

Características a cumplir	Software de DDS	Software del Sistema Isis	Software de CEDRUX
Gestionar Fórmula	X	X	X
Brindar Servicio			
Incorporar campos de la BD			X
Seleccionar Fórmula			
Centralización	X	X	X
Oracle			
Symfony			
Ext Js			X
PHP			X
Propel			
Visual Paradigm			X

Tabla 1.1 Cumplimiento de características por parte de los sistemas estudiados

Los sistemas anteriormente explicados no pueden ser usados para el cálculo de aranceles en la AGR ya que no cumplen con todas las características que debe tener el producto ni con las herramientas usadas para el desarrollo del GINA en el Departamento de Soluciones para la Aduana. A pesar de esto, el estudio de ellos ha brindado nuevas ideas tomadas específicamente del intérprete de ecuaciones de CEDRUX, como lo son el uso de la notación polaca para almacenar y calcular las fórmulas y la necesidad de un analizador para validar su sintaxis.

1.4 Lenguajes, metodologías y herramientas

El Departamento de Soluciones para la Aduana, donde se desarrolla el producto GINA tiene definido la tecnología a usar para su desarrollo. Por otro lado el componente Fórmula formaría parte del producto antes mencionado, por lo que el lenguaje, metodologías y herramientas a usar en el desarrollo de dicho componente están regidos por las usadas en dicho departamento. Seguidamente se realizará una breve fundamentación de cada una de estas tecnologías y herramientas. Para más información al respecto dirigirse al documento **CEIGE-ADUANA-Modelo de Desarrollo de Software** (11) y Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas. (12)

Lenguajes

- **Lenguaje Unificado de Modelado (UML) 2.0:** Prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos.

- **JavaScript**: Es un lenguaje de programación del lado del cliente que se utiliza principalmente para crear páginas web dinámicas.
- **XHTML[5]:** Se utiliza para generar documentos y contenidos de hipertexto generalmente publicados en la WEB y combina la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos.
- **CSS:** Es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma de separar los contenidos de su presentación y es imprescindible para crear páginas web complejas. (13)
- **PHP 5:** es un lenguaje de programación interpretado del lado del servidor, por lo que los script se ejecutan remotamente y el resultado aparece en la máquina cliente (local).

Metodología de Desarrollo

- **Modelo de Desarrollo de Software para el Departamento de Soluciones de la Aduana**

Los proyectos inscritos en el Departamento de Soluciones para la Aduana emplean un Modelo de Desarrollo de Software que se adapta a las necesidades de dicho departamento. Esta definición incluye los flujos de trabajo, encuentros y reuniones básicas, los roles y sus responsabilidades, así como la interacción entre ellos. En la figura 1.1 se puede ver las fases en las que se divide la producción del software en dicho departamento.

[5] Lenguaje de marcado de hipertexto extendido

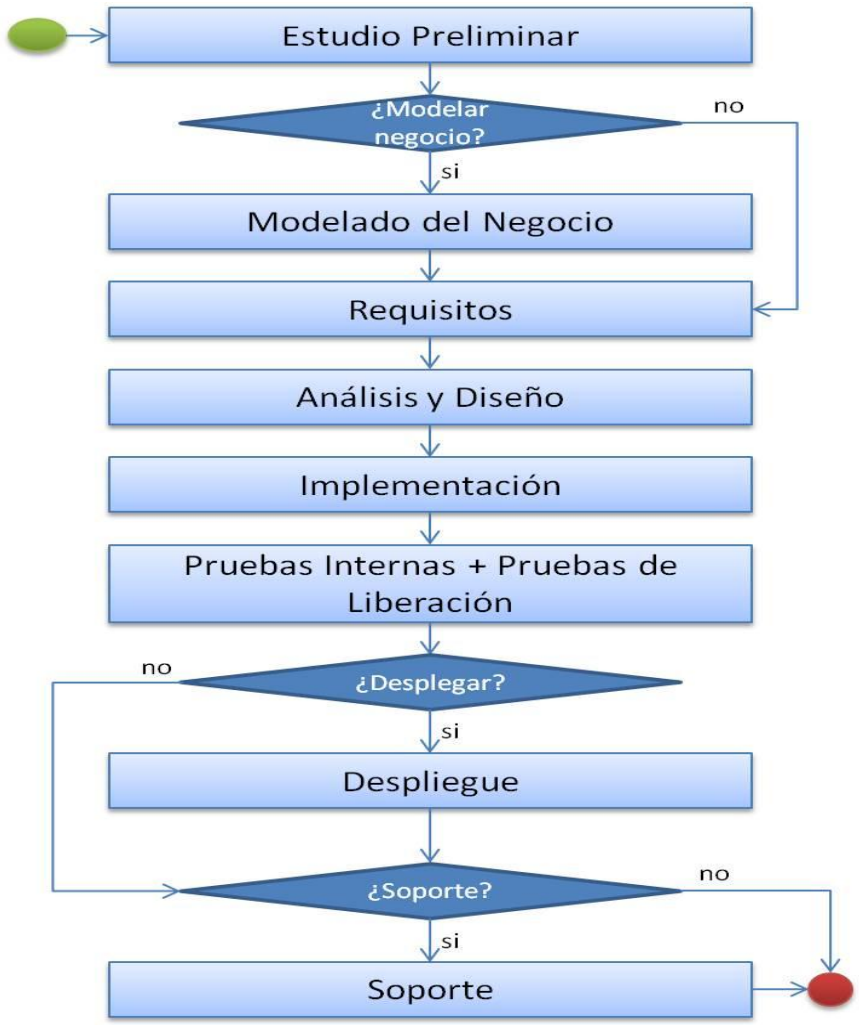


Figura 1.1 Ciclo de vida de proyectos desarrollados en el Departamento de Soluciones para la Aduana

Algunos de los artefactos definidos son:

- Modelo Conceptual
- Descripción de Procesos del Negocio
- Descripción de Requisitos
- Reglas del Negocio
- Modelo de Diseño
- Prototipos de Interfaz de Usuario
- Código Fuente

Herramientas

- **Visual Paradigm para UML 6.4:** Es una herramienta de modelado que permite el diseño de sistemas con todo tipo de tipos de diagramas UML. También es utilizado para diseñar diagramas de casos de uso, diagramas de requerimiento y diseño de bases de datos relacionales. (14) Es una herramienta multiplataforma y se integra con algunas herramientas realizadas en Java entre las cuales se puede citar el NetBeans IDE.

- **NetBeans IDE[6] 6.9:** Es un programa informático compuesto por una serie de herramientas utilizadas para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.

Notación

- **Notación para el Modelado de Procesos de Negocio (BPMN):** Es una notación gráfica que describe la lógica de los pasos de un proceso de Negocio, proporcionando un lenguaje común para que las partes involucradas en el desarrollo de software puedan comunicar los procesos de forma clara, completa y eficiente.

Marco de Trabajo (Framework)

- **Symfony 1.2.8:** 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. (15)

- **Ext JS 3.0:** Fue diseñado para la creación de páginas web dinámicas del lado del cliente basado en lenguaje JavaScript, permite una gran reutilización de componentes, personalización de los mismos, haciendo uso del manejo de tecnologías como AJAX[7] para el intercambio asincrónico de los datos entre el cliente y el servidor, además de lo ventajoso que puede representar la similitud de su aspecto con el de una aplicación de escritorio. (16)

[6] Por sus siglas en inglés Integrated Development Environment
[7] Por sus siglas en inglés Asynchronous JavaScript And XML

Gestor de Base de Datos

- **Oracle:** Es un sistema de gestión de base de datos objeto-relacional desarrollado por la empresa Oracle.

1.5 Conclusiones Parciales

En el presente capítulo se realizó un estudio de algunos de los sistemas informáticos para la gestión de fórmulas matemáticas existentes en nuestro país y a nivel internacional para analizar el posible uso de los mismos y así darle solución al problema u obtener experiencia y conocimiento para realizar una nueva solución. Del estudio de los sistemas se concluyó que la centralización de fórmulas es una característica común en ellos, lo que influye en que los procesos se realicen en muy poco tiempo. Teniendo en cuenta el problema y la conclusión del estudio realizado se evidencia la necesidad de realizar un componente para el Sistema de Gestión Integral de Aduanas (GINA) que se encargue de gestionar fórmulas mediante una interfaz visual, que estas estén centralizadas y que brinde la posibilidad a sus subsistemas de usar dichas fórmulas, debido a la no existencia de un sistema que cumpla con dichas especificidades. Además se concluyó que la solución puede ser realizada con las herramientas, tecnologías y herramientas definidas en el Departamento de Soluciones para la Aduana.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se describe una propuesta del sistema a desarrollar mediante las características principales del componente, describiendo las tareas que se llevarán a cabo en las fases de modelado del negocio, requisitos y análisis y diseño definidas en el modelo de desarrollo descrito en el capítulo anterior.

2.2 Propuesta de solución

El Centro de Informatización de Entidades (CEIGE), en específico, el Departamento de Soluciones Aduaneras en conjunto con el CADI están interesados en realizar un componente para gestionar de manera centralizada las fórmulas utilizadas para el cálculo de aranceles en la AGR llevadas a cabo en el sistema GINA. Dicho componente proporcionaría facilidades para la modificación e inserción de las mismas a través de interfaces orientadas al uso y fácil de entender por los usuarios, evitando de esta manera tener que introducirse en el código para realizar mantenimiento; además brindará servicios que harán posible la utilización de las fórmulas por todos los subsistemas que las necesiten, así como la implementación de condiciones que posibilitarán la asignación de manera automática de estas.

2.3 Descripción de los Procesos del Negocio

Para tener una mayor comprensión de los procesos que desarrollará la solución, se mencionan, a continuación, cada uno de ellos. Luego se describirán algunos de estos procesos mediante Diagramas de Procesos del Negocio y sus especificaciones.

- Administrar Fórmula
    - Gestionar Fórmula
        - Insertar Fórmula
        - Modificar Fórmula
    - Evaluar Fórmula
    - Mostrar Reporte

2.3.1 Descripción del proceso Administrar Fórmula

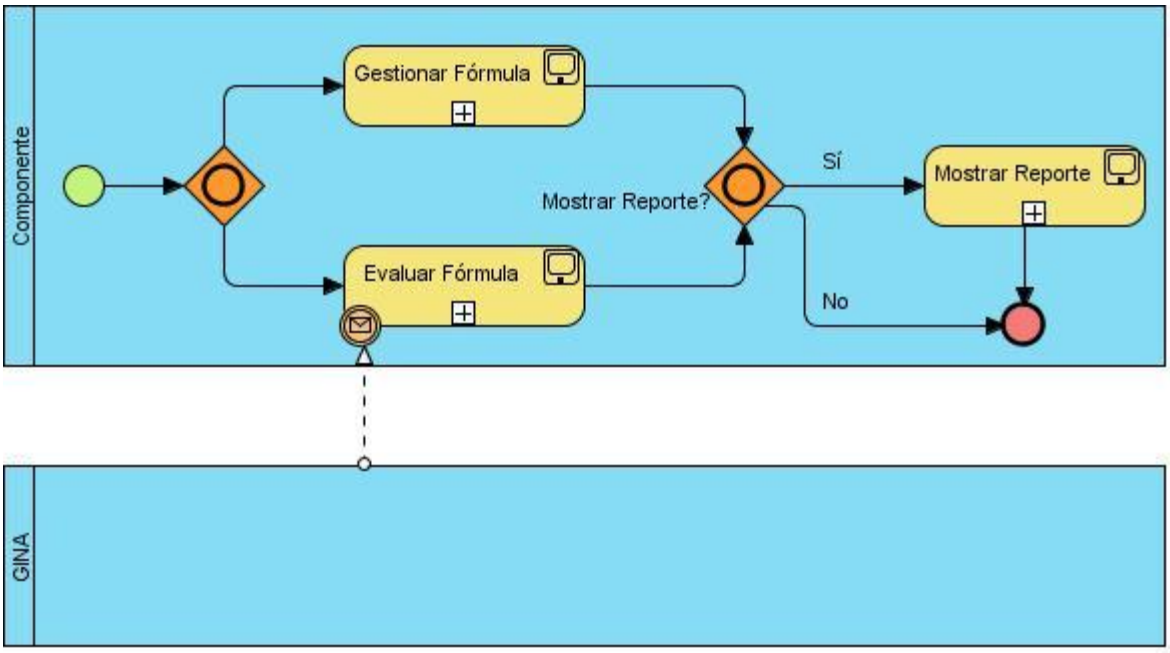


Figura 2.1 Diagrama de procesos Administrar Fórmula

Especificación del proceso Administrar Fórmula

Nombre:	Administrar Fórmula
Misión:	Este proceso se encarga de administrar las fórmulas empleadas en la AGR para el cálculo de aranceles brindando la posibilidad de gestionar y evaluar las mismas al mismo tiempo, además de mostrar Reportes.
Alcance:	Este proceso está dirigido hacia todos los subsistemas del sistema GINA que necesiten administrar fórmulas.
Subprocesos:	• Gestionar Fórmula • Evaluar Fórmula • Mostrar Reporte

2.3.2 Descripción del subproceso Gestionar Fórmula

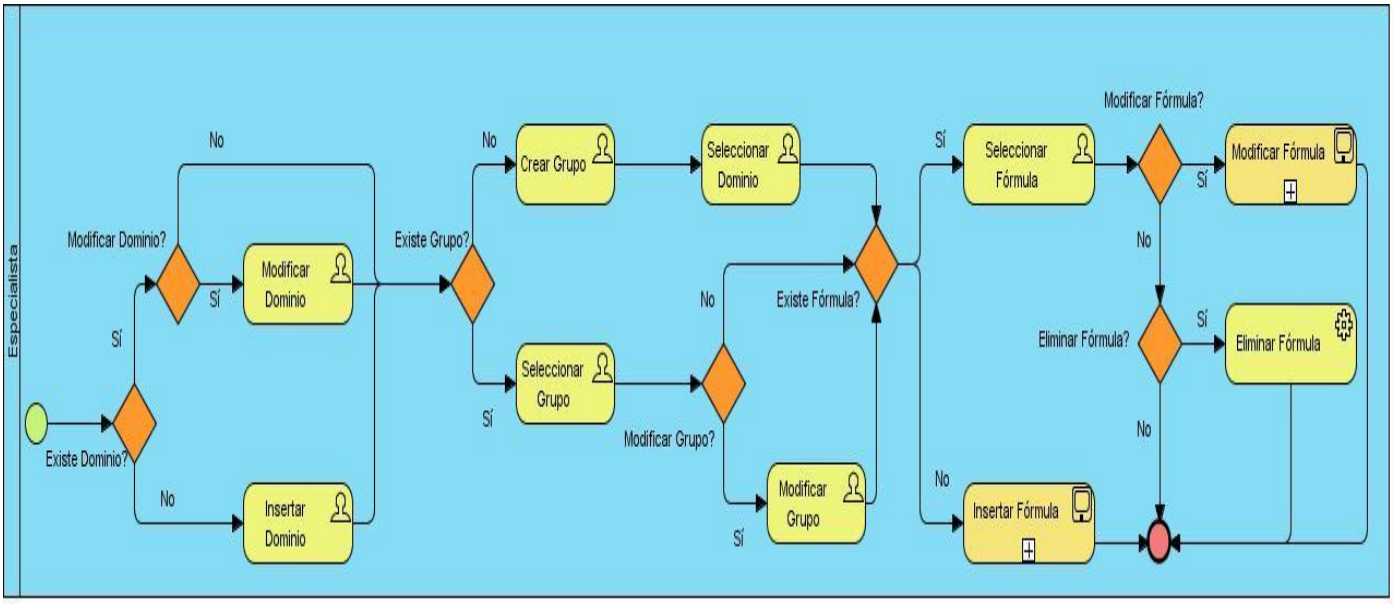


Figura 2.2 Diagrama del subproceso Gestionar Fórmula

Especificación del proceso Gestionar Fórmula

Nombre:	Gestionar Fórmula
Objetivos:	Gestionar las Fórmulas empleadas en la AGR.
Precondiciones:	No aplica
Poscondiciones:	Se han gestionado fórmulas para su posterior uso.
Reglas de Negocio:	No aplica
Responsables:	Especialista en el negocio
Clientes internos:	No aplica
Clientes externos:	No aplica
Entradas:	No aplica
Salidas:	No aplica
Subprocesos	• Insertar Fórmula • Modificar Fórmula

Actividades	**Reglas de Negocio**
1- Insertar Dominio: el actor inserta un nuevo dominio al cual van a	No aplica

estar asociados los grupos de fórmulas.	
2- Modificar Dominio: el actor modifica los datos y/o elementos de un dominio ya existente.	RN_Modificar Dominio
3- Crear Grupo: el actor crea un nuevo grupo, para agrupar una o más fórmulas.	RN_Crear Grupo RN_Fórmulas Grupo
4- Seleccionar Dominio: el actor selecciona un dominio para asociarlo con el grupo.	RN_Seleccionar Dominio
5- Seleccionar Grupo: el actor selecciona un grupo para su posterior modificación.	RN_Seleccionar Grupo
6- Modificar Grupo: el actor modifica los datos y/o fórmulas de un grupo.	RN_Modificar Grupo
7- Seleccionar Fórmula: el actor selecciona una fórmula para luego eliminarla o modificarla.	RN_Seleccionar Fórmula
8- Eliminar Fórmula: el especialista decide eliminar una fórmula de la lista de fórmulas existentes y luego el sistema la elimina completamente.	RN_Eliminar Fórmula

2.4 Requerimientos del Sistema

2.4.1 Técnicas para la captura de requerimientos

Como técnica para la captura de requerimientos se hace uso de la entrevista.

Entrevista: permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Su estructura abarca cuatro pasos:

- identificación de los entrevistados
- preparación de la entrevista
- realización de la entrevista
- documentación de los resultados (protocolo de la entrevista).

Esta técnica requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado. (16)

Como entrevistados se identificaron a algunos trabajadores de los subsistemas del GINA que usan fórmulas sin importar su rol, que pudieran tener conocimiento de cuáles serían los requisitos con los que debería cumplir la solución propuesta. Luego se les informó a los trabajadores antes mencionados la necesidad de su participación en la realización de esta técnica, se realizó la entrevista y se documentaron los resultados obtenidos, quedando así los requisitos funcionales del sistema definidos.

2.4.2 Descripción de los requerimientos funcionales

Los requisitos funcionales expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento). Estos definen qué debe hacer un sistema. (17)

Los requisitos definidos para el sistema se encuentran a continuación:

1 Gestionar Fórmula
   - Insertar Fórmula
   - Modificar Fórmula
   - Eliminar Fórmula
   - Insertar Dominio
   - Modificar Dominio
   - Insertar Grupo
   - Modificar Grupo
   - Analizar sintácticamente

2 Mostrar Reporte
   - Mostrar Reporte de Grupos
   - Mostrar Reporte de Dominios
   - Exportar a PDF

3 Brindar Servicio

- Seleccionar Fórmula
- Evaluar Fórmula

Descripción textual del requisito Insertar Fórmula

Precondiciones	Debe existir al menos un dominio y debe crearse el grupo que contendrá a la fórmula	
Flujo de eventos		
Flujo básico Insertar Fórmula		
No	**Actor**	**Sistema**
1	Selecciona la opción “Fórmulas” en la Interfaz de usuario "Crear Grupo".	
2	Selecciona la opción "Adicionar" para insertar una nueva fórmula	
3		Muestra la interfaz de usuario para adicionar una fórmula al grupo.
4	Selecciona la opción "Número", "Operador", "Paréntesis" o "Elemento”.	
5		En caso de seleccionar la opción “Operador” ir al paso 10 del flujo básico de eventos, en caso de seleccionar la opción “Paréntesis” ir al paso 12 del flujo básico de eventos y en caso de seleccionar la opción “Elemento” ir al paso 14 del flujo básico de eventos.
6		Muestra la interfaz de usuario para adicionar un número a la fórmula
7	Introduce un número para la fórmula.	
8	Selecciona el botón "Aceptar".	
9		Ir al paso 17del flujo básico de eventos
10	Selecciona el operador que desea incluir a la fórmula.	
11		Ir al paso 17del flujo básico de eventos
12	Selecciona el paréntesis que desea incluir a la fórmula.	
13		Ir al paso 17del flujo básico de eventos
14		Muestra la interfaz de usuario para adicionar un elemento a la fórmula con el campo.

15	Selecciona el elemento que desea añadir a la fórmula.	
16	Selecciona el botón "Aceptar"	
17		Muestra las expresiones que se han insertado para la nueva fórmula.
18	Puede eliminar una de las expresiones insertadas, cambiarlas de lugar y/o puede insertar condiciones para la fórmula.	
19		En caso de desear eliminar una expresión ir al paso 21 del flujo básico de eventos, en caso de desear mover las expresiones de lugar ir al paso 25 del flujo básico de eventos y en caso de desear insertar condiciones para la fórmula ir al paso 28 del flujo básico de eventos.
20	Ir al paso 35 del flujo básico de eventos.	
21	Selecciona la expresión que desea eliminar.	
22		Muestra un mensaje de confirmación para eliminar la expresión.
23	Selecciona el botón "Aceptar".	
24		Elimina la expresión visualmente
25	Selecciona la expresión que desea mover de lugar	
26	Selecciona el botón "Arriba" o "Abajo".	
27		Cambia la expresión seleccionada de lugar según la selección.
28	Selecciona la opción “Reglas”	
29	Da doble clic sobre el elemento del dominio que determinará la condición de la fórmula	
30		Muestra la interfaz de usuario para conformar condición de la fórmula
31	Selecciona el operador relacional que poseerá la fórmula.	
32	Selecciona o inserta el valor, dependiendo de si es un campo foráneo o no.	

33	Selecciona la opción "Aceptar" para introducir la regla.	
34		Muestra las reglas que se han insertado a la fórmula.
35	Presiona el botón “Aceptar” de la interfaz “Insertar Fórmula”	
36		Muestra la fórmula que se ha insertado
37	Presiona el botón “Aceptar” de la interfaz de usuario “Crear Grupo”.	
38		Verifica la sintaxis de las nuevas fórmulas. En caso de existir errores ver Flujo Alterno 38.a Sintaxis Incorrecta
39		Inserta las fórmulas y el grupo creado. En caso de existir errores ver Flujo Alterno 39.a Existencia.
40		Concluye el requisito

Poscondiciones
No aplica
Flujos alternativos
Flujo alternativo 38.a Sintaxis Incorrecta

No	**Actor**	**Sistema**
1		Muestra un mensaje indicando que la sintaxis de la nueva fórmula insertada es incorrecta.
2		Concluye el requisito

Poscondiciones
No aplica
Flujo alternativo 39.a Existencia.

No	**Actor**	**Sistema**
1		Muestra un mensaje indicando que la nueva fórmula insertada ya existe.
2		Concluye el requisito

Poscondiciones
No aplica
Validaciones
Se validan los datos según lo establecido en el Modelo conceptual.

Relaciones	**Requisitos Incluidos**	Analizar sintácticamente, en la agrupación Analizar sintácticamente.
	Extensiones	Modificar Fórmula, en la agrupación Gestionar Fórmula. Eliminar Fórmula, en la agrupación Gestionar Fórmula. Mostrar reporte de Grupos de Fórmulas, en la agrupación Mostrar Reportes.
Requisitos	No aplica	

especiales	
Asuntos pendientes	No aplica

Prototipo de interfaz gráfica de usuario del requisito Insertar Fórmula.

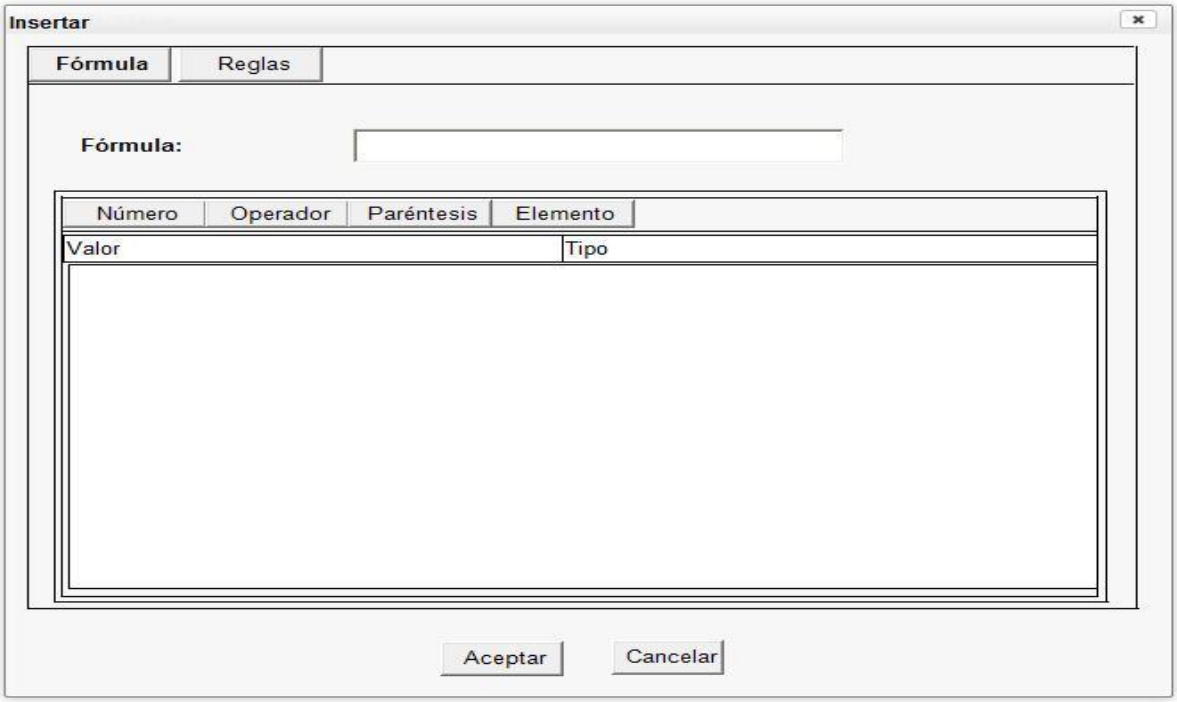


Figura 2.3 Prototipo de interfaz gráfica de usuario del requisito Insertar Fórmula

2.4.3 Técnicas de validación de requerimientos

De las técnicas de validación existentes se usarán para el desarrollo de la solución las revisiones de requerimientos y prototipos.

Revisiones de requerimientos

Consiste en que un equipo de revisores analice sistemáticamente los requerimientos. Esta técnica se realiza manualmente e involucra a usuarios y expertos. Además estas pueden ser informales o formales.

(18) En la revisión formal el equipo de desarrollo conduce al cliente a través de los requerimientos, explicándole las implicaciones de cada uno.

La revisión de requisitos es uno de los mejores métodos de validación de requisitos. Generalmente hablando, las revisiones de requisitos permiten: (19)

- Descubrir una gran cantidad de defectos en los requisitos.
- Reducir los costos de desarrollo entre un 25% y un 30%.
- Reducir el tiempo de pruebas entre un 50% y un 90%.

Las revisiones de requisitos consisten en una o varias reuniones planificadas, donde se intenta confirmar que los requisitos poseen los atributos de calidad deseados. Estas reuniones son llevadas a cabo por el analista encargado del proyecto y un conjunto de colegas[8] que, preferiblemente, no están relacionados con el proyecto y, además, son competentes en la actividad de requisitos. El resultado final de las reuniones de revisión es un documento que contiene la lista de defectos localizados y una lista de acciones recomendadas. Las reuniones de revisión se realizan habitualmente siguiendo un procedimiento compuesto por los siguientes pasos: (19)

1. Preparar el plan de la revisión. Este plan debería incluir al menos lo siguiente: Las tareas a realizar (esto es, las que se indican a continuación), la planificación temporal y las personas que participarán en la revisión.
2. Distribuir los documentos a revisar. Habitualmente, el único documento a revisar será el documento de especificación. Junto con este documento, también es necesario remitir a los participantes en la revisión todos aquellos documentos que ayuden a comprender adecuadamente el documento de especificación. Típicamente, tales documentos son los documentos referenciados y los anexos a la especificación.
3. Preparar la reunión. Esta tarea es muy distinta, dependiendo de quien la realice:

[8] Los cuales pueden ser voluntarios, seleccionados por el analista o asignados a la revisión por cualesquiera otros medios.

- Para el analista promotor de la reunión, esta tarea es fundamentalmente logística. Básicamente consiste en reservar una sala donde realizar la revisión, solicitar los materiales que sean necesarios (desde papel y lápiz a cañones de proyección), preparar justificantes del tiempo empleado por el personal participante (en caso de que sea preciso), etc.

- Para los analistas participantes, la preparación de la reunión es, probablemente, la más importante de las tareas de revisión. Durante esta tarea se deben leer cuidadosamente los documentos recibidos y anotar todas aquellos defectos (o cosas que parezcan defectos) con la finalidad de ponerlos de manifiesto durante la reunión posterior.

4. Realizar la reunión de revisión. Básicamente, el formato de esta reunión puede ser muy diverso: puede oscilar desde una total falta de control hasta grupos muy formalizados y sujetos a protocolos de actuación.

5. Identificar los defectos y acciones a realizar. La lista de defectos y acciones recomendadas es el documento final obtenido en las revisiones de requisitos.

6. Realizar las correcciones que sean precisas a los documentos revisados. El analista promotor de la reunión (esto es, el analista encargado del documento de especificación) debe evaluar y, si lo estima conveniente, llevar a cabo, las acciones recomendadas que han surgido de la reunión de revisión.

7. Informar de las modificaciones realizadas a los participantes en la reunión. Una vez que los defectos en la especificación han sido subsanados, debería enviarse un breve informe de las tareas realizadas, y una copia corregida de los documentos de especificación, a los participantes en la reunión para su visto bueno.

Prototipos

Consiste en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Es muy importante que el usuario entienda que lo que está viendo no es el sistema final. (16)

Las tareas indicadas para esta técnica son las siguientes: (19)

• Seleccionar quién evaluará el prototipo. Para que la evaluación del prototipo sea lo más efectiva posible, deben seleccionarse adecuadamente los usuarios que participarán en la evaluación. Siempre que sea posible, debe seleccionarse un conjunto de usuarios representativos de los distintos perfiles, de tal modo que sea posible validar el software en todos sus modos de utilización. Por ejemplo: Si un software es utilizado por administrativos, ingenieros y contables, al menos un usuario de cada uno de estos tipos de usuarios debería evaluar el prototipo del software.

• Desarrollar escenarios de validación. Para evaluar correctamente el prototipo, deben identificarse una serie de escenarios o tareas, los cuales deben ser llevados a cabo por los usuarios utilizando el prototipo. En este caso se realizaron escenarios para cada interfaz de usuario que presenta el sistema. En la mayoría de los casos, dichos escenarios pueden identificarse directamente a partir del detalle de la especificación. Otra alternativa para generar los escenarios es intentar confeccionar un manual de usuario para el sistema. Si este manual está correctamente confeccionado, deberá contener las alternativas de utilización del sistema, lo cual es exactamente lo mismo que un escenario.

• Ejecutar los escenarios. Esta tarea consiste en que el usuario realice, o ejecute, cada uno de los escenarios previstos. El analista deberá orientar al usuario acerca de la diferencia entre el producto final y el prototipo. Asimismo, el analista deberá paliar las carencias del prototipo, proporcionando al usuario la información que el prototipo no ofrezca y que el usuario necesite para ejecutar los escenarios.

• Documentar los problemas. Esta tarea consiste, simplemente, en confeccionar la lista de problemas encontrados.

2.5 Metodologías de diseño

Desarrollo de un sistema o método para una situación única. Este término se aplica a los campos tecnológicos en referencia al diseño web, software o diseño de sistemas de información.

De las metodologías de diseño existentes se hace uso de los diseños orientado al uso y centrado en el usuario. Esto se debe a que el trabajo que se desarrolla en el Departamento de Soluciones para la Aduana se basa en las necesidades del cliente.

Diseño Orientado al Uso: el diseño se centra en las tareas asociadas al uso y sus objetivos. La usabilidad se refiere a la capacidad del software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

Diseño Centrado en el Usuario: es la metodología de diseño más habitual a nivel mundial. Coloca todas las necesidades, deseos y limitaciones del usuario como núcleo del proceso de diseño, razón por la cual, es necesario investigar y analizar al usuario.

El uso de estas metodologías pudo lograrse a través de la técnica de validación de prototipos donde se hizo lo posible porque los mismos se centraran en ser comprendidos y fáciles de usar por el usuario, teniendo como uno de los objetivos cumplir con las necesidades del usuario.

2.6 Patrones de diseño

Para la implementación de Symfony se utilizan varios patrones, situándolos en las capas de Modelo y Control que plantea el patrón arquitectónico MVC. (20) Dichos patrones son:

Patrones GRASP[9] (20)

- Experto: Este patrón permite asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. La responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo o implementarlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantendrá encapsulada. (21) 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 para trabajar con la tabla que representan.

[9] De sus siglas en inglés General Responsibility Assignment Software Patterns, describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

- Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos (21). En la clase ”actions.class.php” 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.class.php” es creador de dichas entidades. Ejemplos de algunas funciones utilizadas en la clase “actions.class.php” para el caso de las clases de abstracción a datos son: “doSelect()”, “retrieveByPK()”, “doSelectOne()”.
- Alta Cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase (21). 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 properties.
- Bajo Acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas (21). 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.
- 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 en este caso haciendo uso del controlador frontal para acceder a la aplicación.

Patrones GoF[10] (20)

- Patrón Command: Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma (22). 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

[10] De sus siglas en inglés Gang of Four

del administrador del sistema donde se aplique el marco de trabajo, 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.

- Patrón Decorator: Añade funcionalidad a una clase dinámicamente (22). 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 la plantilla 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. Este procedimiento es una implementación de este patrón.

2.7 Modelo del Diseño

El diseño de sistemas define 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. En esta fase han de tenerse en cuenta los procedimientos relativos a la actividad Gabinete de Programación. (23)

2.7.1 Diagrama de paquetes

Los Diagramas de Paquetes se usan para reflejar la organización de los paquetes y sus elementos, y para proveer una visualización de sus correspondientes nombres de espacio. (24)

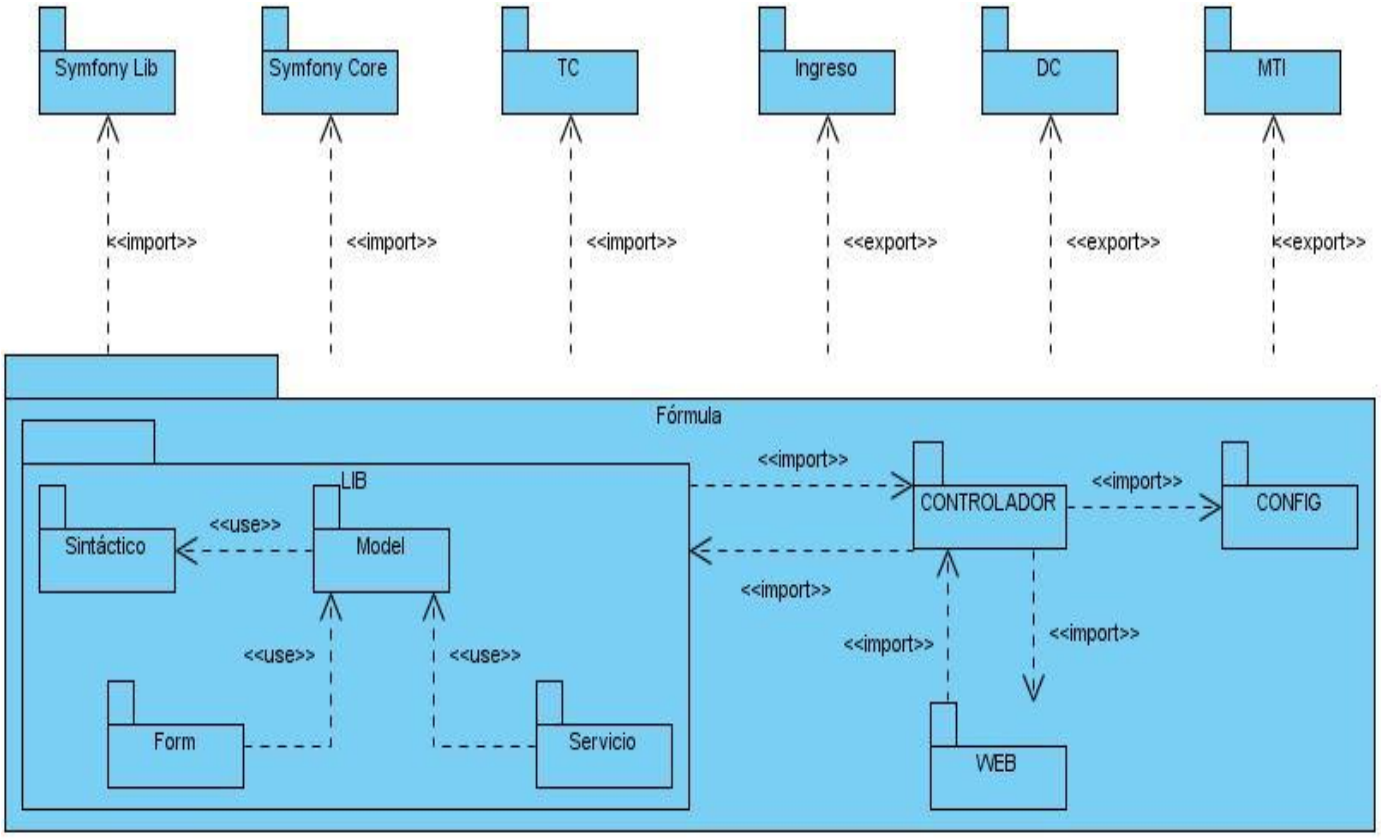


Figura 2.6 Diagrama de Paquetes del Componente Fórmula

En la figura 2.6 se muestra el diagrama de paquetes del componente Fórmula, compuesto por los paquetes que se describen a continuación:

Paquetes que son necesarios para el funcionamiento del componente:

- Symfony Lib: almacena las librerías de symfony.
- Symfony Core: representa el núcleo de symfony.
- TC: representa el esquema Tablas de Control, es decir, las tablas nomencladoras del GINA y otras del propio componente.

Paquetes que usarán los servicios del componente:

- Ingreso: subsistema del GINA.
- DC (Despacho Comercial): subsistema del GINA.

- MTI (Medios de Transporte Internacional): subsistema del GINA.

Paquetes propios del componente:

- LIB: contiene las librerías propias del componente y está compuesto por los paquetes:
  - Model: contiene las clases del modelo de datos, hace uso del paquete Sintáctico y es empleado por los paquetes Form y Servicio.
  - Sintáctico: contiene las clases referentes al análisis sintáctico que es realizado a las fórmulas antes de insertarlas.
  - Form: contiene los formularios necesarios para validar los datos antes de almacenarlos.
  - Servicio: contiene las clases donde se implementa el servicio que será brindado a otros subsistemas.
- Controlador: emplea el paquete Config para dirigir el flujo de actividades de la aplicación, esto lo hace modificando el modelo y abriendo y cerrando vistas en dependencia de los mensajes o datos introducidos en las mismas por el usuario.
- Config: almacena la configuración del componente.
- WEB: encargado de agrupar los ficheros (imágenes, estilos y clases JavaScript) que contienen y decoran las vistas de la aplicación.

2.7.2 Diagrama de clases del Sistema

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. (25)

Un diagrama de Clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Nos sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso. (25)

El diagrama de clases del sistema cuenta con 21 clases, para más información sobre el objetivo de las mismas dirigirse al documento Modelo de Diseño del Componente Fórmula.

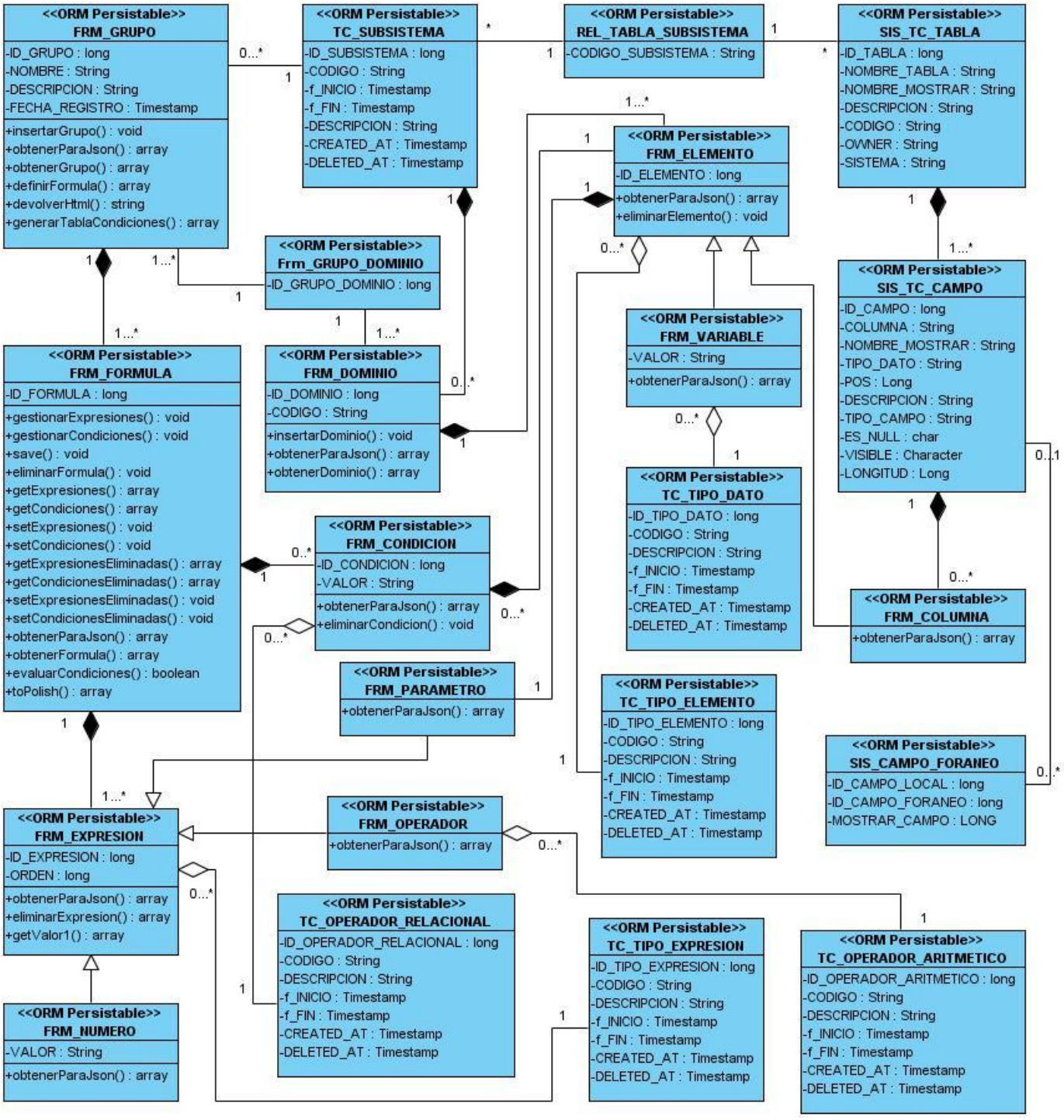


Figura 2.7 Diagrama de clases del componente Fórmula

2.7.3 Diagrama de Interacción del Diseño

El diagrama de interacción, representa la forma en como un Cliente (Actor) u Objetos (Clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente. (26)

Diagrama de Secuencia

En un diagrama de secuencia se indicarán los módulos o clases que forman parte del programa y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada. (27)

Diagrama de Secuencia Orientado a Actividades del Negocio

El objetivo principal de este diagrama es definir la interacción entre las clases y los usuarios 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. Pueden existir uno o varios diagramas asociados a cada requisito funcional. Se representan por calles, actividades y la relación entre estas actividades incluyendo también comentarios para mayor comprensión. (14)

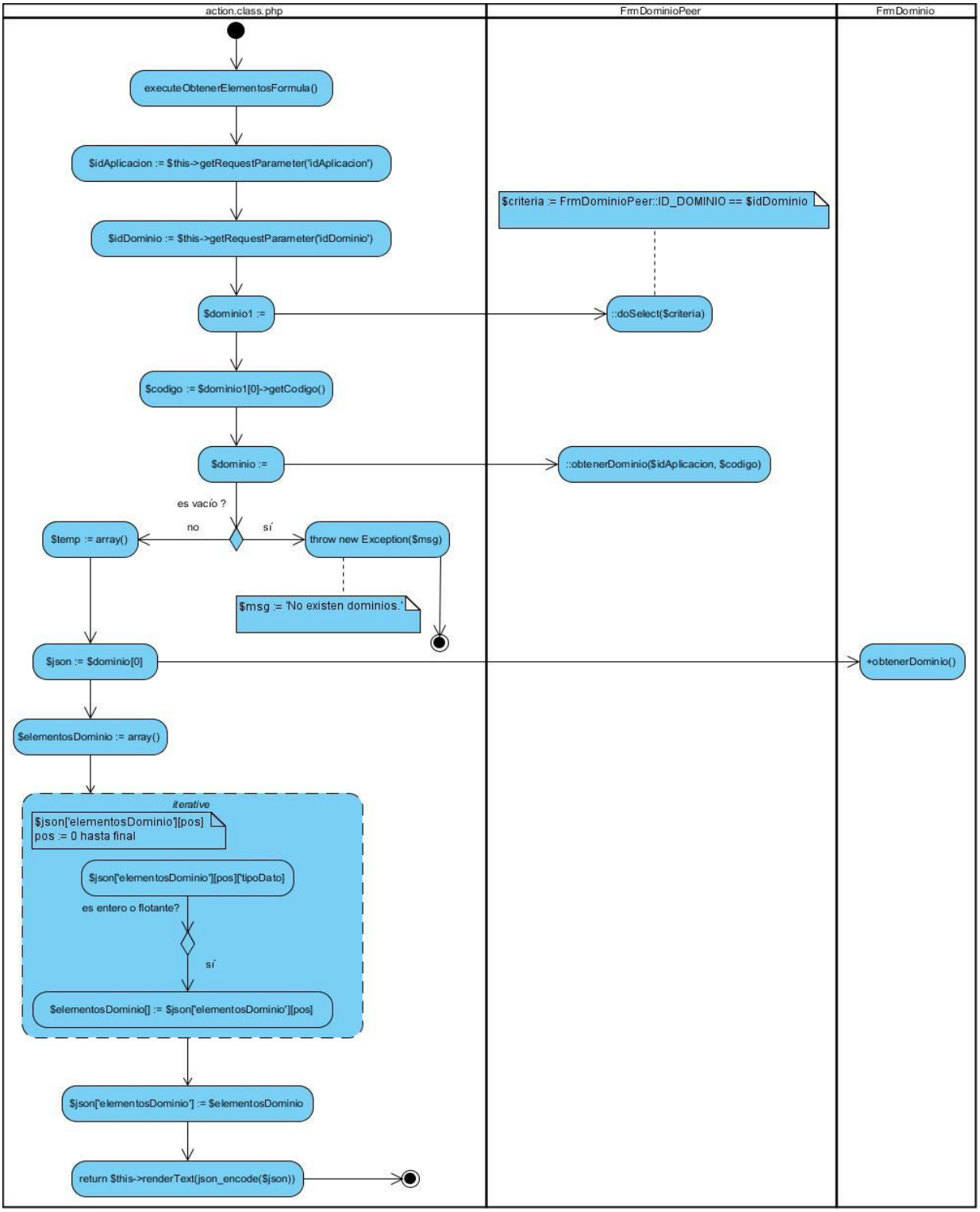


Figura 2.8 Diagrama de Secuencia Orientado a Actividades del Negocio

El diagrama de la figura 2.8 se refiere a la funcionalidad “executeObtenerElementosFormula()” de la clase actions.class.php. Esta funcionalidad es la encargada de obtener los elementos de tipo entero o flotante para adicionar como expresiones a la fórmula, estos elementos pertenecen al dominio asociado al grupo donde se encuentra la fórmula. En la figura, el diagrama consta de 3 calles que constituyen la clase donde se encuentra la funcionalidad y otras dos clases participantes en la ejecución de la misma, incluye además actividades que identifican cada paso seguido, así como consultas a la Base de Datos, lanzamiento de excepciones, bifurcaciones y recorridos a los arreglos.

Diagrama de Secuencia Orientado a Actividades de Componentes Visuales

Este diagrama presenta las mismas políticas del orientado a negocio. Está dirigido principalmente a los programadores de las interfaces de usuario para que tengan una base por donde guiarse a la hora de brindar las funcionalidades y la forma de comportamiento de las pantallas. Se evidencian en ellos una serie de comentarios y acciones que incluyen nombre de las funcionalidades a llamar, así como de los valores a enviar y mensajes que mostrar. (14)

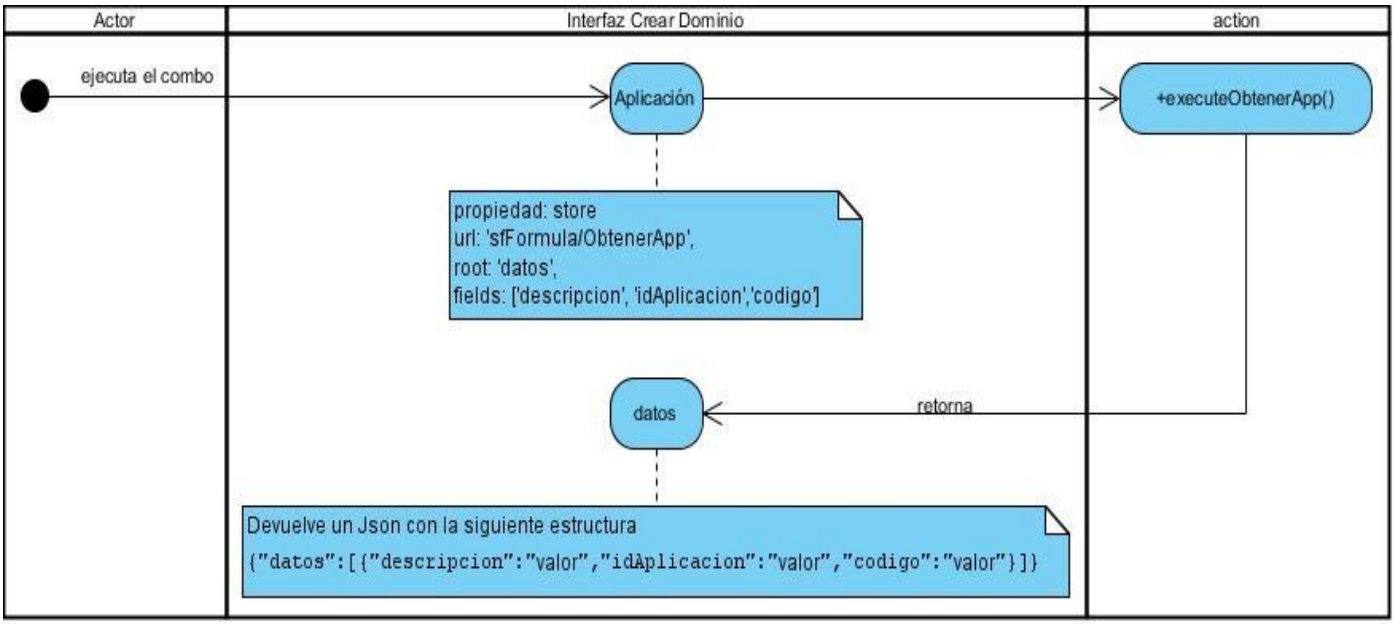


Figura 2.9 Diagrama de Secuencia Orientado a Actividades de Componentes Visuales

El diagrama de la figura 2.9 se refiere al combobox Aplicación, de la interfaz de usuario Insertar Dominio.

El actor ejecuta el combobox llamado “Aplicación”, el cual carga sus datos de una dirección (url) donde se indica el nombre de la funcionalidad de la clase “actions.class.php” encargada de retornar todos los subsistemas existentes en el GINA, en este caso “executeObtenerApp()”. Además es necesario indicar los fields[11] y el root[12]. Luego la funcionalidad antes mencionada envía los datos a la vista para ser mostrados al usuario.

Para más información sobre los diagramas de secuencia que presenta el sistema dirigirse al Modelo de Diseño (28) del Componente Fórmula.

2.7.4 Diseño de la Base de Datos

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener la base de datos de un sistema de información determinado. En el caso relacional, esta estructura será un conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias, claves foráneas, etc. (29)

Modelo Entidad-Relación

A continuación se muestra el modelo Entidad – Relación del componente que contiene las entidades empleadas para almacenar la información necesaria y las relaciones entre ellas. Las entidades de color verde son las más importantes arquitectónicamente ya que almacenan los datos más significativos para le gestión y centralización de las fórmulas, ejemplo de ellas son FrmGrupo, FrmFormula y FrmExpresion; las de color gris pertenecen al esquema TC (tablas de control) de la base de datos, permitiendo el acceso de todos los subsistemas que conforman el GINA y las restantes contribuyen al completamiento de información necesaria para el funcionamiento del componente. Todas estas entidades conforman un modelo de entidad-relación en Tercera Forma Normal (3FN) ya que no existen atributos compuestos, multievaluados o compuestos (1FN[13]), los mismos no tienen dependencias parciales (2FN[14]), ni existen dependencias transitivas (3FN).

[11] Nombres de los campos que serán enviados a la vista.
[12] Elemento raíz del cual se van a obtener los datos.
[13] Primera Forma Normal
[14] Segunda Forma Normal

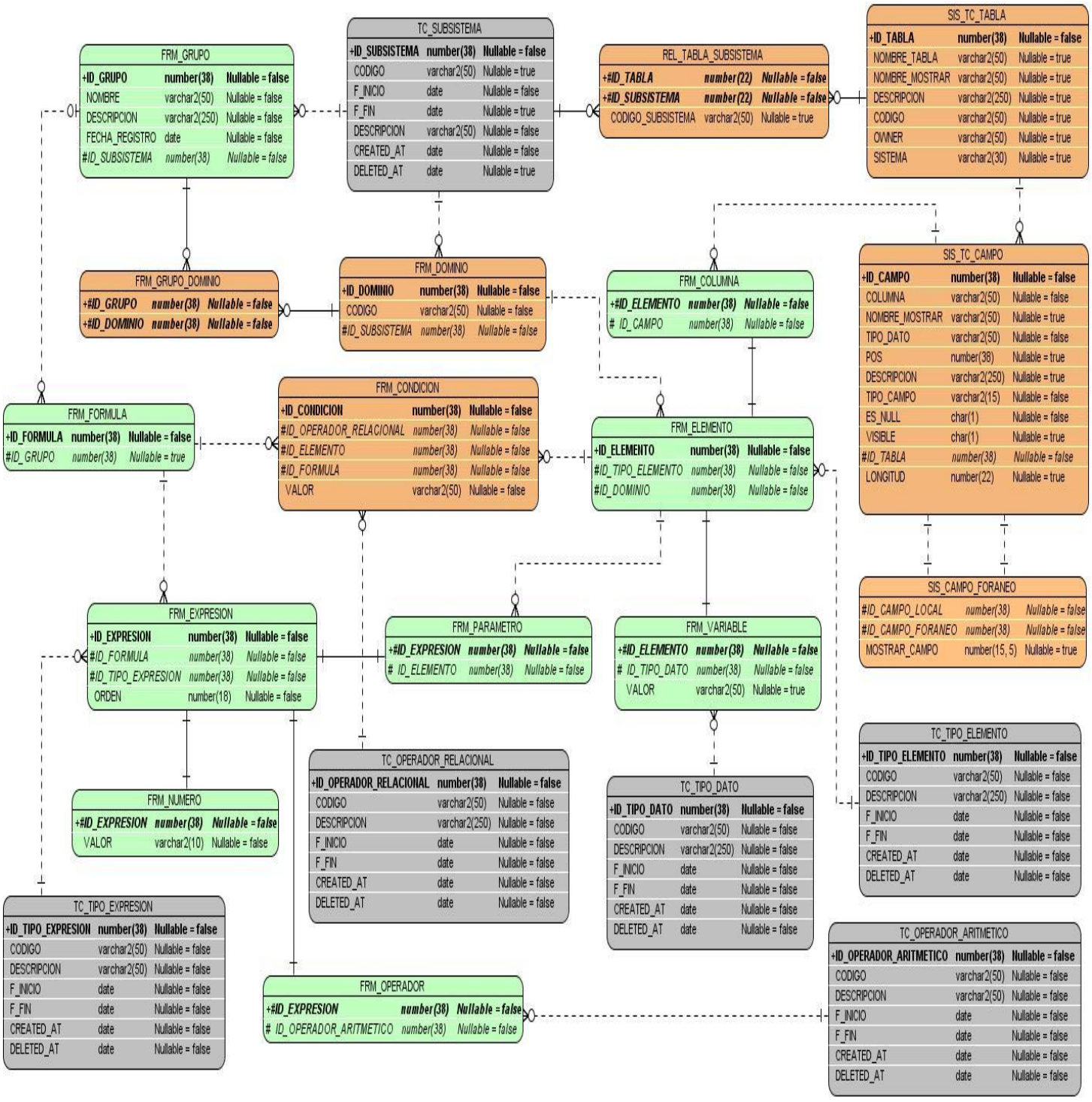


Figura 2.10 Modelo Entidad Relación del componente Fórmula

2.8 Conclusiones parciales

En el presente capítulo se detallaron las características principales de la propuesta de solución específicamente el análisis y diseño de la misma, obteniendo 13 requisitos, de los cuales los más significativos son insertar, modificar, y eliminar fórmula, además de analizar sintácticamente ya que son los que específicamente permiten gestionar las fórmulas y tener las mismas centralizadas. Además se realizó el diagrama entidad-relación donde se obtuvieron 22 entidades, de las cuales 9 fueron las más significativas ya que almacenarán los grupos de fórmulas y las expresiones de las mismas. Además se mencionaron los patrones de diseño a utilizar propios del marco de trabajo empleado en el desarrollo de la solución y se realizaron los diagramas de secuencia orientado a actividades, los cuales conforman las entradas necesarias para la implementación de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

Luego de haber culminado la fase de análisis y diseño de la solución corresponde darle paso a la fase de implementación de la misma para luego comenzar la fase de Pruebas. En el presente capítulo se especificarán algunos de los estándares de codificación y patrones de arquitectura utilizados para llevar a cabo la implementación de la aplicación. Además se describirán clases importantes para el entendimiento de su funcionalidad, así como el proceso seguido para el tratamiento de las excepciones. Además se realizarán las pruebas necesarias para la validación de la solución obtenida.

3.2 Modelo de implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (30)

Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre éstos. (30)

Para representar los diagramas del Modelo de Implementación se puede emplear el diagrama de UML de Componentes. (30)

3.2.1 Patrones de Arquitectura

Los patrones de arquitectura están relacionados fundamentalmente con la estructura de un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Describen un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (31)

Patrón Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres capas o componentes distintos. El patrón MVC se utiliza frecuentemente en aplicaciones web, donde la vista es la página HTML (del inglés, Hypertext Markup Language) que se visualiza en el navegador y el código que proporciona datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista, enviarlos al modelo y manejar también las respuestas del modelo y transmitirlas hacia la vista. (32)

En este caso se hace uso de este patrón de la siguiente forma:

- Modelo: mediante el uso de Propel se generan las clases con todas las funcionalidades comunes de las entidades y las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan. Ejemplo: las clases “FrmFormula.php”, “FrmGrupo.php” y “FrmExpresion.php”, cada una de ellas con su respectiva clase de abstracción de datos.
- Vista: en este caso se crean los archivos de JavaScript (JS), cada uno de los cuales representa una interfaz de usuario. Ejemplo: los archivos “Formula.js”, “ModificarGrupo.js” y “WInsertarRegla.js”
- Controlador: el controlador en este caso es la clase “actions.class.php”.

3.2.2 Estándar de Codificación

El objetivo de crear un estándar de codificación es que los programadores se rijan por el mismo a la hora de implementar. Este estándar debe servir para el personal del proyecto para identificar de forma sencilla cual es el objetivo y las funcionalidades que brinda cada una de las clases, funciones y demás componentes de Software dada su nomenclatura, es necesario que esto se pueda identificar a simple vista. Además debe servir de guía para los implementadores que tienen que continuar el desarrollo de las aplicaciones luego. (33)

La implementación realizada para el desarrollo del componente está regida por el documento Propuesta de un estándar de codificación (34) del Departamento de Soluciones para la Aduana ya que el mismo comprende todas las aplicaciones desarrolladas bajo dicho Departamento.

Algunas de los estándares más significativos que se encuentran el documento antes mencionado son:

- Todos los nombres de acciones y las clases deben estar en la nomenclatura UpperCamelCase[15] comenzando por la palabra execute en el primer caso.
- En los nombres de las clases no se deben utilizar guiones bajos en su "_".
- Los nombres de los archivos para la realización de pruebas deben coincidir con el nombre de la clase o la funcionalidad que están probando terminada en el sufijo "Test.php".
- Los archivos de pruebas se colocarán en el directorio raíz del proyecto, dentro del directorio "test/unit".
- Los nombres de los plugins deben iniciar con el prefijo **sf**.

3.2.3 Tratamiento de Errores

Un aspecto importante a la hora de desarrollar un software es identificar y controlar los posibles errores que pueden ocurrir cuando se interactúa con el mismo, para asegurar su calidad, es por ello que es indispensable tratarlos. A continuación se mencionan algunos posibles errores que se identificaron en la solución propuesta:

- Inserción de datos inválidos o repetidos en la base de datos.
- Búsqueda o intento de eliminación de elementos que no existen en la base de datos.

Para evitar la ocurrencia de los posibles errores detectados se realizaron validaciones en las vistas para lograr que el usuario no insertara valores inválidos al sistema, es decir, la aplicación no permite que el usuario introduzca textos en campos numéricos, ni viceversa y se muestran mensajes de error en caso de que el usuario intente insertar un elemento ya existente o con campos vacíos en casos que no puedan serlo.

En el modelo se realizan validaciones mediante el uso de formularios de symfony, los cuales poseen funcionalidades para validar los datos de la clase que representan antes de ser insertados, haciendo uso del lanzamiento de excepciones en caso de la ocurrencia de errores.

Además en las diferentes clases del modelo, en caso de que existan errores, se lanzan excepciones que luego son capturadas y mostradas notificando al usuario para que pueda corregirlos. Es necesario aclarar que estas excepciones mostradas al usuario en forma de mensajes son muy precisas y brindan solo la información necesaria para el usuario, con el objetivo de evitar el uso de esta información para vulnerar la aplicación.

[15] UpperCamelCase: estilo de escritura que se aplica a frases o palabras compuestas, donde cada palabra va unida a la otra, con la peculiaridad de que la primera letra de cada término se encuentra en mayúscula.

En el anexo1 se muestran los posibles errores que son mostrados en la vista.

3.2.4 Diagrama de Componentes

Los diagramas de componentes se usan para describir los elementos físicos del sistema y sus relaciones. En la figura 3.1 se muestran los principales componentes que presenta el sistema.

El controlador frontal constituye el único punto de entrada a la aplicación, carga la configuración y en dependencia de las peticiones realizadas en las interfaces (archivos **js**), hace una llamada a la acción a ejecutarse en la clase “actions.class.php”. Por otra parte las interfaces de usuario usan las librerías de Ext JS que contiene los archivos “ext-all.css”, el cual contiene todos los estilos para visualizar correctamente los componentes, “ext-base.js” que contiene solo lo que necesita Ext JS para su correcto funcionamiento y “ext-all.js” que contiene todos los componentes y características disponibles en el marco de trabajo.
La clase “actions.class.php”, a su vez, interactúa con las clases del negocio creadas por Propel, las cuales mediante servicios obtienen información de las tablas de control y son usadas por el el paquete Servicio donde se encuentra la clase “IntegracionComun” que es la encargada de brindar servicios a los subsistemas MTI, DC e Ingreso.

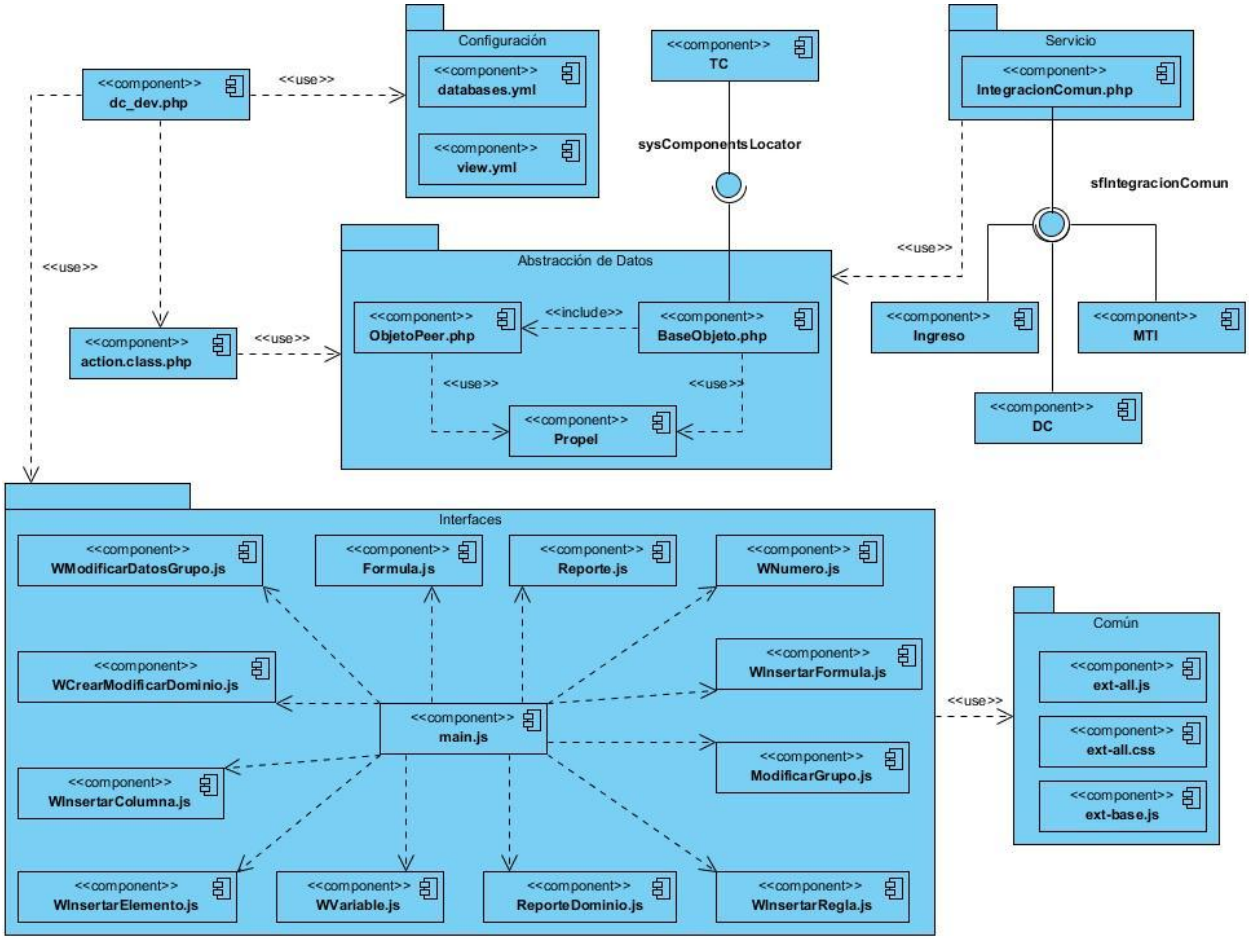


Figura 3.1 Diagrama de Componentes del sistema.

3.3 Validación de la Solución

Una de las validaciones realizadas a la solución fue la relacionada con las variables definidas en el problema propuesto: gestión dinámica y centralización.

Después de una revisión bibliográfica se encontró que algunos de los conceptos que se usan para definir ***dinamismo*** son:

- Energía activa y propulsora. Actividad, presteza, diligencias grandes. Sistema que considera el mundo corpóreo como formado por agrupaciones de elementos simples, realmente inextensos, y cuyo fondo esencial es la fuerza; de suerte que los fenómenos corpóreos resultan del choque de fuerzas elementales, y se reducen en definitiva a modos del movimiento. (35)

- Se trata de la habilidad para trabajar arduamente en situaciones cambiantes o alternativas, que cambian es cortos espacios de tiempo, en jornadas de trabajo prolongadas sin que por esto se vea afectado su nivel de actividad. (36)

Teniendo en cuenta cada uno de los conceptos antes mencionados y enfocándolos al trabajo que se ha llevado a cabo, se relaciona el dinamismo con la posibilidad que brinda el componente a los usuarios de realizar actividades con las fórmulas de manera sencilla y delimitada ante situaciones cambiantes.

La ***centralización*** es la acción o iniciativa para reunir distintas cosas en un centro común. (37) Además la Real Academia de la Lengua Española define la centralización como: hacer que varias cosas dependan de un poder central (38).

Sobre la base de los conceptos definidos se fijaron indicadores para operacionalizar las variables centralización y dinamismo de manera conjunta, debido a la vinculación que existe entre ellas durante la ejecución de la aplicación. Esta vinculación se manifiesta en el cómo a través del componente se logra reunir en un mismo lugar los procedimientos definidos, con la capacidad de realizarlos de manera sencilla, con claridad y elegancia, previendo además una serie de acciones para lograr la validez de las operaciones realizadas. Los indicadores definidos fueron los siguientes:

- **Dependencia**: todos los subsistemas dependen del componente realizado para acceder a las fórmulas que necesiten. La manifestación de esta indicador se ve en el hecho de que el componente es quien almacenará todas las fórmulas trayendo consigo que en caso de que uno de los subsistemas necesiten usar una de ellas tengan que pedir los servicios brindados por el componente, dependiendo totalmente del mismo.
- **Reutilización**: se define como la facilidad de los subsistemas de poder utilizar las mismas fórmulas sin necesidad de repetirlas. En este caso si muchos subsistemas usan las mismas fórmulas solo necesitarán re-implementar el código que las define, sino que solo tendrán que acceder al componente y reutilizarlas.
- **Actualización**: se define como la capacidad de realizar modificaciones a las fórmulas en el componente logrando que estas queden disponibles para cada uno de los subsistemas que las usen. En caso de que sea necesario realizar modificaciones a alguna de las fórmulas empleadas solo tendremos que hacer este procedimiento en el componente y no en cada subsistema que emplee dichas fórmulas.

- **Comunicación**: se refiere a la existencia de reglas con el objetivo de lograr el entendimiento entre los subsistemas y el componente. Entre el componente realizado y los subsistemas del GINA la comunicación se realiza cuando estos piden usar el servicio que este brinda, para ello deben especificar el nombre del grupo de fórmulas que desean emplear, los valores de los elementos del dominio pertenecientes a dicho grupo y la aplicación en la cual se encuentran el grupo y el dominio antes mencionado. De esta forma el componente podrá brindar el servicio satisfactoriamente.

Para demostrar la validez de cada uno de estos indicadores se realizó como estrategia una prueba unitaria al método “evaluarFormula()” de la clase “IntegracionComun”, para ello el marco de trabajo usado, symfony, proporciona herramientas y utilidades para automatizar las mismas.

Esta estrategia de prueba asegura que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto. (15)

En la figura 3.2 se muestra cómo un subsistema usaría el servicio brindado por la solución propuesta, mediante el uso de la prueba unitaria antes mencionada.

```
27  $elementosDominio = array(
28      'var1' => '2',
29      'var2' => '3',
30      TcProveedorPeer::DESCRIPCION => 'Pepe',
31      TcProveedorPeer::CODIGO => 'pepe'
32  );
33
34  try {
35      $result = IntegracionComun::evaluarFormula('PRUEBA', $elementosDominio, 8);
36
37      $t->is($result, 51, 'EXITO');
38  } catch (Exception $exc) {
39      echo $exc->getMessage();
40  }
```

```
Output - GINA (test:unit PruebasServicioFormula)
1..1
ok 1 - EXITO
 Looks like everything went fine.
```

Figura 3.2 Prueba Unitaria realizada a la funcionalidad “evaluarFormula()” de la clase “IntegracionComun”.

En la figura anterior se pueden demostrar la dependencia y la reutilización en el hecho de que varios subsistemas del GINA que usen las mismas fórmulas podrán y tendrán que usar el servicio brindado usando la línea 35, donde se hace un llamado al mismo, sin tener que repetir la implementación del código de definición de las mismas.

La comunicación se evidencia ya que para que los subsistemas puedan usar el servicio necesitan enviar datos o parámetros como son los elementos del dominio (línea 27 hasta línea 32), nombre del grupo e id de la aplicación donde se encuentran este dominio y grupo (línea 35).

Por otra parte la figura 3.3 demuestra el indicador actualización ya que se evidencian claramente las modificaciones que se pueden realizar a las fórmulas en el componente que quedarán disponibles para todos los subsistemas que las necesiten.

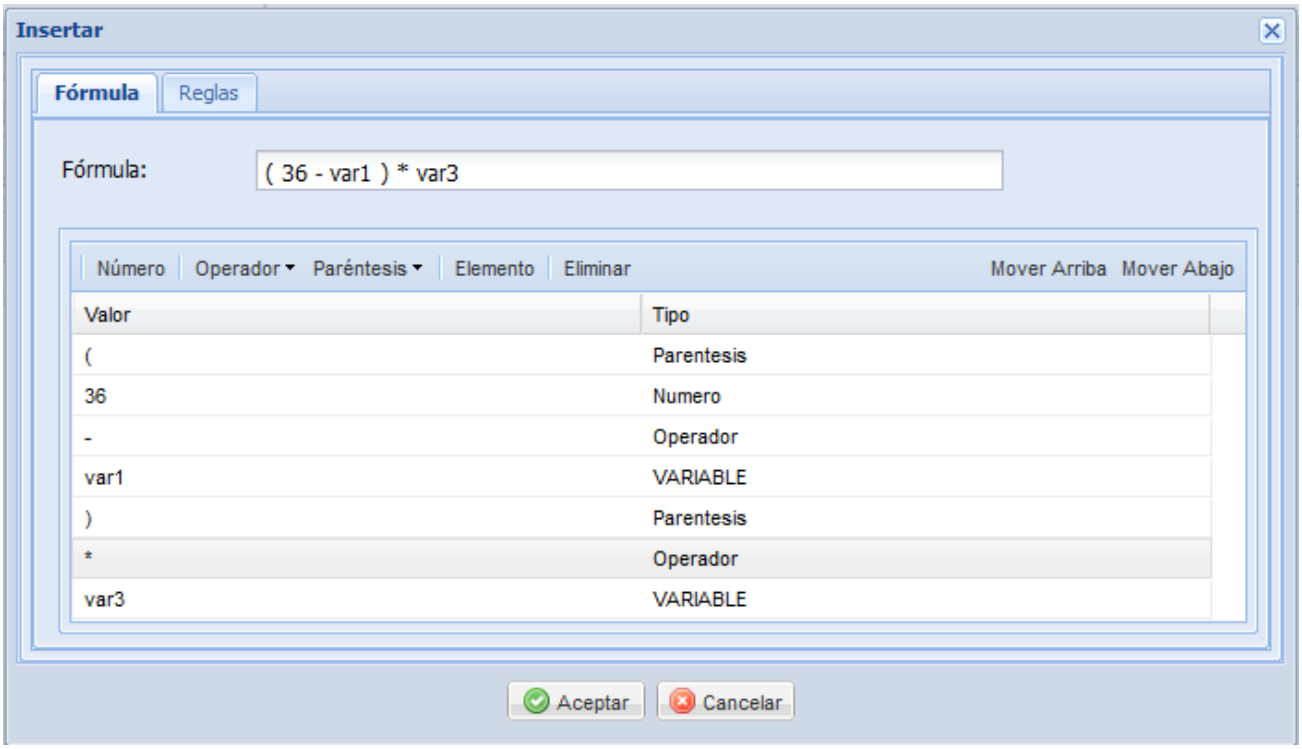


Figura 3.3 Inserción y modificación de las fórmulas pertenecientes a un grupo.

En la tabla 3.1 se muestra el diseño de caso de prueba unitaria realizada. La funcionalidad probada se encarga de verificar que se cumplan cada una de las condiciones de cada una de las fórmulas que se encuentran en el grupo que recibe como parámetro para seleccionar la correcta, evaluarla y devolver el resultado al subsistema que realizó la petición del servicio.

Para esta prueba se realizó una única iteración arrojando resultados satisfactorios.

Prueba: Unitaria		
Nombre Prueba: evaluarFormulaTest		
Estado: Satisfactoria	**Tipo**: Regresión	**Última ejecución**: 5/06/2012
Ejecutado por: Saily de la Caridad Panuncia Ridel	**Verificado por**: Saily de la Caridad Panuncia Ridel	
Descripción: Se verifica que la funcionalidad devuelva los datos correctos, realizando las validaciones correspondientes, seleccionando la fórmula correcta según el cumplimiento de sus condiciones y devolviendo como resultado el resultado de evaluar la misma según los datos entrados.		
Entrada: Arreglo con los elementos del dominio, y sus valores, asociado al grupo donde se encuentra la fórmula que deseamos, nombre del grupo e id de la aplicación donde se encuentran el grupo y el dominio.		
Criterio de aceptación: Devuelve el resultado de evaluar los valores entrados en la fórmula que se necesita usar perteneciente a un grupo.		
Resultado: Retorna el valor correcto de la evaluación de la fórmula correcta para este caso.		

Tabla 3.1 Prueba unitaria a la funcionalidad “evaluarFormula()” de la clase “IntegracionComun”

Técnicas de prueba: (39)

Las técnicas de evaluación dinámica o prueba proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Estas técnicas se agrupan en:

- Técnicas de caja blanca o estructurales, que se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- Técnicas de caja negra o funcionales, que realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

Otras de las pruebas realizadas fueron las técnicas de pruebas de caja negra o funcionales.

Pruebas de caja negra o funcionales:

El componente se ve como una “Caja Negra” cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable se selecciona un conjunto de ellas sobre las que se realizan las pruebas. Para seleccionar el conjunto de entradas y salidas sobre las que trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que

causan un comportamiento erróneo en nuestro sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Entonces, dado que la prueba exhaustiva es imposible, el objetivo final es pues, encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible. (39)

Se realizaron pruebas de caja negra para comprobar que la aplicación cumple con los requisitos funcionales definidos. Para la realización de estas pruebas se realizaron los siguientes diseños de casos de prueba (DCP):

- DCP Insertar Dominio.
- DCP Insertar Formula.
- DCP Insertar Grupo.
- DCP Modificar Dominio.
- DCP Modificar Formula.
- DCP Modificar Grupo.
- DCP Mostrar Reporte de Dominios.
- DCP Mostrar Reporte de Grupos.

La tabla 3.2 muestra el diseño de caso de prueba para la interfaz insertar grupo, el cual consta de un solo escenario y la tabla 3.3 muestra las variables usadas en el mismo.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Insertar Datos Grupo	Permite insertar los datos propios del grupo.	Inserta correctamente el grupo creado.	1 - El actor selecciona la opción “Crear" en el Menú "Grupo". 2 - El sistema muestra la interfaz de usuario para insertar un nuevo grupo con los datos: nombre, dominio, descripción y elementos del dominio. 3 - El actor selecciona la aplicación donde desea insertar el nuevo grupo. 4 - El actor introduce el nombre y la descripción para el nuevo grupo.

			5 - El actor selecciona el dominio que desea asociar al grupo. 6 - El sistema muestra los elementos del dominio seleccionado. 7 - El actor selecciona la pestaña "Fórmulas" para insertar fórmulas al grupo, ver Diseño de Casos de Prueba Insertar Fórmula. 8- El actor selecciona el botón "Aceptar". Si el grupo no posee al menos una fórmula el sistema muestra un mensaje de error indicando que el grupo debe tener fórmulas y si existe algún campo inválido el sistema muestra un mensaje indicándolo. 9 - El sistema inserta el nuevo grupo. 10 - Termina el requisito.

Tabla 3.2 Diseño de caso de prueba de la interfaz Insertar Grupo

Descripción de las variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Aplicación	Lista desplegable	No	Aplicación del GINA en la cual se va a crear el nuevo grupo.
2	Nombre	Campo de Texto	No	Nombre que tendrá el nuevo grupo.
3	Dominio	Lista desplegable	No	Dominio que será asociado al nuevo grupo
4	Descripción	Campo de Texto	No	Descripción que tendrá el nuevo grupo.
5	Datos del Dominio	Campo de	No	Datos del dominio asociado al

		Selección		grupo.

Tabla 3.3 Descripción de variables para el diseño de caso de prueba de la interfaz Insertar Grupo

Se contó con dos iteraciones. La primera iteración arrojó como resultado 19 no conformidades, de las cuales 17 fueron significativas y en la segunda iteración todas las no conformidades fueron corregidas.

3.4 Conclusiones Parciales

En el presente capítulo se mencionaron algunos de los estándares de codificación por los que se rigió la implementación del sistema asegurando así, un mejor entendimiento del código fuente del sistema. Además se definió como se tratarían los errores que pudieran ocurrir en la aplicación con el objetivo de eliminar problemas ocurridos en tiempo de ejecución y se mostraron las pruebas necesarias realizadas para evaluar el software, que dio la posibilidad de llegar a la conclusión de que el componente realizado cumple con los requisitos definidos.

CONCLUSIONES

Para la realización del presente trabajo se realizó un estudio del estado del arte, el cual arrojó el resultado de que ninguno de los sistemas informáticos estudiados daría solución al problema planteado pero sí sirvieron de ejemplo ya que se tomaron sus principales ventajas para el desarrollo de la aplicación. Se realizó el análisis y diseño de la propuesta de solución teniendo como resultado la especificación, descripción y validación de los requisitos funcionales del sistema, además de diagramas de clases, modelo entidad relación, diagramas de secuencia, entre otros artefactos definidos en el Departamento de Soluciones para la Aduana, necesarios para la posterior implementación del sistema.

El componente obtenido permitió la centralización de las fórmulas usadas en la AGR y la gestión dinámica de las mismas mediante interfaz de usuario.

Por todo lo anterior expuesto se concluye que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente ya que la aplicación da solución a la situación polémica que le dió origen, por lo que la misma está en condiciones de ser desplegada en la AGR.

RECOMENDACIONES

Una vez cumplido con los objetivos de la investigación, teniendo en cuenta las experiencias obtenidas en la misma, se recomienda:

- Desplegar la solución en la AGR.
- Integrar la solución con el resto de los componentes del GINA.
- Continuar el estudio referente a la centralización y gestión dinámica de fórmulas para la posible inserción de nuevas funcionalidades para el mejoramiento de la solución.

BIBLIOGRAFÍA

1. Sitio Web de la Aduana Cubana. *CONTROL ADUANERO SOBRE LAS CARGAS Y LOS MEDIOS DE TRANSPORTE INTERNACIONAL* . [En línea] [Citado el: 14 de diciembre de 2011.] http://www.aduana.co.cu/comercio/comercio.htm.

2. **D, Lic. Luis Ml Sánchez.** *Aduana Siglo XXI.* Santo Domingo : s.n., 2007.

3. **Española, Real Academia.** Real Academia Española. *Diccionario de la Lengua Española.* [En línea] [Citado el: 5 de diciembre de 2011.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=aduana.

4. **Magna, Agenda.** Agenda Magna. *Glosario de términos aduaneros.* [En línea] [Citado el: 5 de diciembre de 2011.] http://agendamagna.wordpress.com/2009/02/26/glosario-de-terminos-aduaneros/.

5. **Rodríguez, Gorge I. Arce.** *La Importancia de las Aduanas en el Comercio Exterior.* San Marcos : s.n., 2008.

6. **Alvarez, Yersy Soto.** Slideshare. *Slideshare.* [En línea] [Citado el: 2 de diciembre de 2011.] http://www.slideshare.net/wilpica/clasificacion-arancelaria.

7. **Española, Real Academia.** Real Academia Española. *Diccionario de la Lengua Española.* [En línea] [Citado el: 5 de diciembre de 2011.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=c%C3%A1lculo.

8. **ABC, Definición.** Definición ABC. *Definición ABC.* [En línea] [Citado el: 5 de diciembre de 2011.] http://www.definicionabc.com/ciencia/formula.php.

9. **Informáticos, DDS Sistemas.** DDS Sistemas Informáticos. *DDS Sistemas Informáticos.* [En línea] [Citado el: 5 de diciembre de 2011.] http://www.ddsoftware.com.ar/software-sueldos.html.

10. **ISIS, Sistema.** Sistema ISIS. *Sistema ISIS.* [En línea] [Citado el: 5 de diciembre de 2011.] http://www.sistemaisis.com/erp-produccion.htm.

11. **Aduana Repositorio, Dpto Soluciones para la.** [En línea] 2012. [Citado el: 20 de junio de 2012.] http://10.52.17.252:5700/Repo/AGR.

12. **Rodríguez, José Antonio Cobo.** Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas. 2008.

13. librosweb.es. *librosweb.es.* [En línea] [Citado el: 5 de junio de 2012.] http://www.librosweb.es/css/capitulo1.html.

14. **Guerra, Boris Enrique Ruiz.** *Tránsito y Transferencia para el Despacho Comercial en la Aduana General de la República de Cuba.* Habana, Cuba : s.n., 2011.

15. **Fabien Potencier, François Zaninotto.** *Symfony, la guía definitiva.* 2008.

16. **Moreno, Juan Javier Dans.** *Modelación e implementación del módulo Documentos del subsistema Depósitos de Aduana.* Habana : s.n., 2011.

17. **I.A, Departamento de Ciencias de la Computación e.** [En línea] [Citado el: 20 de Febrero de 2012.] http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf.

18. **Sommerville, Ian.** *Ingeniería del Software.* Madrid España : s.n., 2005.

19. [En línea] [Citado el: 20 de Junio de 2012.] http://is.ls.fi.upm.es/docencia/masterTI/ARS/docs/Manual_M2C1U11.pdf.

20. EcuRed. [En línea] [Citado el: 20 de junio de 2012.] http://www.ecured.cu/index.php/Patrones_en_Symfony.

21. **Visconti, Marcello y Astudillo, Hernán.** Fundamentos de Ingeniería de Software. *Patrones de Diseño.* [En línea] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

22. EcuRed. *EcuRed.* [En línea] [Citado el: 25 de junio de 2012.] http://www.ecured.cu/index.php/Patrones_Gof.

23. *Gestión y Diseño de Sistemas.*

24. [En línea] [Citado el: 20 de abril de 2012.] http://analisisydiseoiii.blogspot.com/2011/05/diagrama-de-paquetes.html.
25. [En línea] 22 de mayo de 2009. [Citado el: 20 de abril de 2012.] http://egdamar877.blogspot.com/2009/05/expocicion.html.
26. [En línea] [Citado el: 20 de abril de 2012.] http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html.
27. **Tello, Jesús Cáceres.** DIAGRAMAS DE SECUENCIA. [En línea] [Citado el: 20 de abril de 2012.] http://www2.uah.es/jcaceres/capsulas/DiagramaSecuencia.pdf.
28. **Ridel, Saily de la Caridad Panuncia.** *Modelo de Diseño.* 2012.
29. **Costa, Dolors Costal.** Introducción al diseño de bases de datos. *Introducción al diseño de bases de datos.*
30. MeRinde. *MeRinde.* [En línea] [Citado el: 20 de abril de 2012.] http://merinde.net/index.php?option=com_content&task=view&id=495&Itemid=291.
31. **Hernández, Pedro Veloso.** *Uso de Patrones de Arquitectura.* 2009.
32. **Pantoja, Ernesto Bascón.** *El patrón de diseño modelo -vista - controlador (MVC) y su implementación el Java Swing.* 2004.
33. **Aduana, Departamento de Soluciones para la.** *Propuesta de un Estándar de Codificación.* Habana : s.n., 2011.
34. **Aduana, Departamento de.** *Propuesta de un estandar de codificación.* 2012.
35. **Española, Real Academia de la Lengua.** Real Academia Española. *Real Academia Española.* [En línea] [Citado el: 25 de junio de 2012.] http://lema.rae.es/drae/?val=dinamisma.
36. CEP de Alcalá de Guadaíra - Aula Virtual: Redes Profesionales. *Diccionario de Competencias Genericas y Especificas.* [En línea] [Citado el: 25 de junio de 2012.] http://www.redes-cepalcala.org/inspector/DOCUMENTOS%20Y%20LIBROS/COMPETENCIAS/DICCIONARIO%20DE%20COMPETENCIAS.pdf.
37. Definicion.de. [En línea] [Citado el: 22 de junio de 2012.] http://definicion.de/centralizacion/.
38. **Española, Real Academia de la Lengua.** Real Academia Española. [En línea] [Citado el: 25 de junio de 2012.] http://lema.rae.es/drae/?val=centralizar.
39. [En línea] [Citado el: 5 de junio de 2012.] http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf.

ANEXOS

Anexo 1: Tratamiento de errores.